

Technical Report
840

DARPA Neural Network Study
Final Report
(October 1987 — February 1988)

22 March 1989

Lincoln Laboratory

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

LEXINGTON, MASSACHUSETTS



Prepared for the Tactical Technology Office of the
U.S. Defense Advanced Research Projects Agency (DARPA/TTO)
with the support of the Balanced Technology Initiative Program
under Electronic Systems Division Contract F19628-85-C-0002.

Approved for public release; distribution unlimited.

ADA 20 7580

Part I

INTRODUCTION

FOREWORD

During the summer of 1987, I commissioned a group of technical advisors to define a Defense Advanced Research Projects Agency (DARPA) initiative in neural networks. The group consisted of Steven Andriole, George Mason University; Harris Eisenhardt, DARPA consultant; Lee Giles, Air Force Office of Scientific Research; Alfred Gschwendtner, MIT/Lincoln Laboratory; and Peter Kemmey, DARPA.

It was recommended by the group, initially, that DARPA issue a call for position papers on potential research and application topics. After this solicitation as a Broad Area Announcement (a DARPA vehicle for expressions of interest), and prior to an overwhelming response of almost 300 papers, it was further suggested to me by Al Gschwendtner that DARPA sponsor a national study on neural networks.

The motivations for a national study were manifold. During the summer of 1987, a meeting in San Diego on neural networks attracted almost 2,000 participants and attendees. In fact, numerous meetings were springing up and attracting many enthusiasts. During this same period, the whole field of artificial intelligence was undergoing a serious re-examination, and by some it was perceived that neural networks might be just a re-packaging of old ideas and promises; others perceived that neural networks represented the dawn of a new era in computers. In addition, the various Services of the Department of Defense (DoD) were initiating a variety of neural network projects and, of course, Europe and Japan began touting their own national programs. The United States had no national perspective and it was, therefore, recommended that DARPA undertake a leadership role in evaluating the status and promise of neural networks and possibly establishing a national research program.

Toward that end, MIT/Lincoln Laboratory was directed by DARPA to establish a Terms of Reference to guide the Study and to assist DARPA in the overall management and execution of the effort (this and other details of the Study are discussed in the Introduction to the Neural Network Study Technical Report that follows). The Study was initiated with a two-day symposium at Lincoln Laboratory, where national experts in various aspects of neural network research were invited to present their views.

This symposium emphasized what we had found out in preparing for the Study - - that there are many different and even conflicting views of what neural networks are, what they can do, and how they should be implemented. During the course of the Study, we examined these diverse views and attempted to shed light on the issues.

After participating in this Study, my personal view is that neural networks will provide the next major advance in computing technology. Over the history of computing science, two advances have matured: high-speed numerical processing and knowledge processing (artificial intelligence). Neural networks seem to offer the next necessary ingredient for intelligent machines - namely, knowledge formation and organization.

I would like to thank all of those who participated in the Study's Panels, the Study Panel chairmen, and members of the steering committee for their dedicated efforts in conducting the Study and their hard work in producing this report.

The many pages that follow comprise a thorough review and assessment of the state of neural network research and development, chiefly in the United States, in late 1987 and early 1988; they take into account not only the enthusiastic visions of those committed to the promise of neural networks, but the critical

questioning of those who are willing to acknowledge the limited scope of past and current neural network research efforts. This Technical Report of the DARPA Neural Network Study is, therefore, a valuable sourcebook for anyone interested in neural networks and it is proffered in the spirit of scientific inquiry.

Dr. Jasper Lupo
Defense Advanced Research Projects Agency
Washington, DC
June, 1988

STUDY DIRECTOR'S OVERVIEW

The beginnings of a science of neural networks can be traced back to ancient times, but the modern forms of this subject began with the work of McCulloch and Pitts in 1943. They were the first to show that Boolean operations could be performed using "neural" elements modeled after living neurons. A great deal of work has followed and continues, with acceleration, to this day. Hundreds of workers in the U.S., an equivalent number in Europe, and many hundreds more in Japan and the rest of the world are currently active in the field.

Various agencies of government in the U.S., including DARPA, have begun to fund neural network research at low levels. Industrial companies have begun working on the subject using internal funds.

Workers in many companies have formed spontaneous neural net groups to meet during lunch times and at off hours to discuss and do research on neural networks, not as a part of company business. At many universities, students have formed seminar groups on neural networks to hear invited speakers and to discuss their own research ideas, hoping that they will have the opportunity and the financial support to pursue Ph.D. research in neural nets.

Under the leadership of Stephen Grossberg in the U.S., Teuvo Kohonen in Finland, and Shun-Ichi Amari in Japan, the International Neural Network Society was formed in the Spring of 1987. Their journal *Neural Networks* has just appeared, Vol. 1, No. 1, 1988. The "IEEE First International Conference on Neural Networks" was held during June 21-24, 1987. Hundreds of papers were presented in oral and poster sessions. Approximately 2,000 people attended. An IEEE "Conference on Neural Information Processing Systems - Natural and Synthetic" was held November 8-12, 1987 in Denver, with about 750 attendees. It was an excellent scientific meeting. The IEEE Second International Conference on neural networks will be held during July 1988 in San Diego, and the first Conference of the International Neural Network Society (INNS) will be held in Boston on September 6-10, 1988. Six of the IEEE Societies are co-sponsors of the INNS meeting. Financial support has come from the National Science Foundation, DARPA, and many other governmental agencies. It is anticipated that it will be well attended and will be a fine technical success.

The field of neural networks exists. Its emergence has met with mixed reactions, however. Publicity, venture capital, and hyperbole has accompanied the solid achievements of the field. Among scientists, the presence of hype and extravagant claims casts a dark shadow and makes the work controversial in scientific circles and amongst the world at large.

Why does this field attract hype? I think that answer is this. Neural network scientists are talking about modeling the human brain and its parts, about gaining an understanding of how the brain works. Others in the field are talking about building new and unusual forms of computers having brain-like capability and being constructed of brain-like parts. The ongoing work is good and serious, being done by scientists who are facing problems of profound intellectual interest, and problems of enormous technical difficulty. Some results have been achieved and the field is beginning to show "signs of life." However, work of this type often seems to be sensational and hype unfortunately results. How to evaluate and appraise the work is not easy.

For many years, DARPA has been a great leader in developing the most advanced computer and communications technologies in the U.S. From an historical perspective, it is natural for DARPA to have a

strong interest in neural networks. Involvement is difficult, however, in the presence of hype. The field has to prove its worthiness, and this requires demonstration that neural networks can solve certain important difficult problems in a unique and superior way.

To ascertain and appraise the state of neural network research and applications, this Study was undertaken by DARPA. The Study began on October 8, 1987 and ended on February 26, 1988. Because of the tight time schedule, the Study was primarily addressed to work in the U.S. Hundreds of neural network researchers and practitioners were interviewed by the various panelists, who were flown all over the US to accomplish this task. Logistics, organization, and support were supplied by MIT/Lincoln Laboratory staff and scientists under the able supervision of Al Gschwendtner. Representatives of every school of thought in neural networks were consulted. If we succeeded, this Report will be a fair and balanced representation of neural networks – 1988, in the U.S.A.

Prof. Bernard Widrow, Study Director
Palo Alto, CA
February, 1988

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS	viii
ACKNOWLEDGEMENTS	xi
I INTRODUCTION	
1. NEURAL NETWORK STUDY AND THE TECHNICAL REPORT	1
1.1 Structure of the Study	1
1.2 Structure of the DARPA Neural Network Study Technical Report	9
2. OVERVIEW OF THE STUDY'S FINDINGS AND CONCLUSIONS	13
2.1 Why Neural Networks?	13
2.2 Theoretical Foundations of Neural Networks	16
2.3 Available Implementation Tools	29
2.4 Status of Current Neural Network Applications	35
2.5 Projected Computational Capabilities for Advanced Simulators	45
2.6 General Conclusions of the Neural Network Study	50
APPENDIX A – LIST OF PRESENTATIONS	53
APPENDIX B – NEURAL NETWORK GLOSSARY OF TERMS	65

LIST OF ILLUSTRATIONS

Figure No.		Page
1-1	DARPA Neural Network Study Terms of Reference	2
1-2	DARPA Neural Network Study Symposium Agenda	3
1-3	DARPA Neural Network Study Participants	5
1-4	DARPA Neural Network Study Steering Committee	7
1-5	Neural Network Study Organization	8
1-6	DARPA Neural Network Study Schedule	10
2-1	Clippings of Headlines from 1987 Reports Citing Interest in Neural Networks	14
2-2	Biological Neurons	16
2-3	Artificial Neurons	18
2-4	Single Neuron Learning	20
2-5	Computational Power of Single- and Multi-layer Networks	21
2-6	Massive Parallelism of a Neural Network	22
2-7	History of Neural Network Research	24
2-8	Comparison of Information Processing Approaches	24
2-9	Current Information Processing Approach	26
2-10	Neural Network Approach	27
2-11	Neural Networks for a Variety of Tasks	28
2-12	Neural Network Computation Units	30
2-13	Computational Capabilities of Biological Networks	31
2-14	Examples of Neural Network Simulation Hardware	33
2-15	Computational Capabilities of Neural Network Simulators	34
2-16	Application Survey	36
2-17	Development of Neural Network Applications	37
2-18	Kanji Character Recognizer (Nestor)	38
2-19	Perception (Van Allen)	39

Figure No.		Page
2-20	Sonar Target Discrimination (Sejnowski)	40
2-21	Multi-layer Perceptron Vowel Classifier (Lippmann)	41
2-22	Forklift Robot (Martin Marietta)	43
2-23	Computational Requirements for Current Applications	44
2-24	Projected Computational Capabilities for Advanced Simulators	46
2-25	Direct Implementation Approaches	48
2-26	Implementation Technologies and Potential Applications	49
2-27	Projected Technology Development	51

ACKNOWLEDGEMENTS

This report contains the work of many people, including members of the Study and scientists from the broad field of neural networks, as well as those who gave technical presentations to the Study and the hundreds of researchers who are referenced in the text. Special thanks are due to the panel chairmen and associates – especially the latter, to whom fell the difficult task of compiling the work of the Study into this Report.

Ms. Carol Weizmann served throughout the Study and during the writing of this report as technical editor. Her notes and skill at turning the variety of styles and prose of Study members into a cohesive report were invaluable. Ms. Joyce Michaels carried out the difficult duties of administrative assistant for the Study professionally and with composure during its long and arduous passage. Finally, special thanks are due to Dr. Paul Kolodzy, who not only contributed greatly to the Study by serving as the associate chairman of Panel 3, but also spearheaded the drive to computerize the activities and technical reporting of the Study and thus simplified all of our lives.

A. B. Gschwendtner
MIT/Lincoln Laboratory

1. NEURAL NETWORK STUDY AND THE TECHNICAL REPORT

The Neural Network Study, sponsored by the Tactical Technology Office of the U.S. Defense Advanced Research Projects Agency (DARPA/TTO), was conducted under the auspices of the Massachusetts Institute of Technology's Lincoln Laboratory (MIT/LL) from October, 1987 through February, 1988. Its formal objectives were several:

- To identify potential applications for neural networks in Department of Defense (DoD) systems,
- To determine the current neural network technology base,
- To identify technology requirements, and
- To identify a DoD program plan for the next five years.

1.1 STRUCTURE OF THE STUDY

The Study's Terms of Reference were formalized before its initiation in October 1987 (see Figure 1-1). The Terms mandated the organization and management of an in-depth Study comprised of government, industry, and academic participants, with particular attention to theory, technology, and applications.

The Study began on October 8, 1987 at Lincoln Laboratory with a two-day technical symposium. After the nature and purpose of the Study was explained, symposium participants – who together constituted a significant number of the fledgling neural network community's leading proponents and many of whom served on the Study's various panels or its steering committee – were introduced to the Study's director and offered a thorough technical review of the neural network field. This review was the Study's starting point. (The symposium agenda can be found in Figure 1-2.)

In order to achieve the objectives of the DARPA Neural Network Study, and in accord with the Terms of Reference, five working panels mandated to conduct the activities of the Study were formed, each with a chairman and an MIT/Lincoln Laboratory scientist serving as associate. In addition, a sixth panel was created to produce a program plan; its membership included the chairmen and associates of the five working panels as well as several consultants. The panels included:

- **Panel 1: Intelligent Systems – Status and Expectations** (later informally renamed **Technology Assessment**), which was chaired by Dr. Edward Posner, California Institute of Technology; Dr. Thomas Goblick served as the MIT/Lincoln Laboratory associate.
- **Panel 2: Adaptive Knowledge Processing**, which was headed by Dr. John Pearson of SRI; the MIT/LL associate was Dr. Richard Lippmann.
- **Panel 3: Simulation/Emulation – Tools and Techniques**, which was led by Dr. Andrew Penz, Texas Instruments, with Dr. Paul Kolodzy acting as the MIT/LL associate.

- (1) ORGANIZE AND MANAGE AN IN-DEPTH STUDY OF NEURAL NETWORKS AND THEIR APPLICATION TO DoD SYSTEMS. PARTICIPATION WILL INCLUDE REPRESENTATIVES FROM GOVERNMENT, INDUSTRY AND UNIVERSITIES**
- (2) THE STUDY WILL CONSIST OF A STEERING GROUP, A STUDY DIRECTOR AND FIVE TECHNICAL PANELS. A SIXTH PANEL WILL BE FORMED EARLY IN THE STUDY TO DEVELOP A PROGRAM PLAN FOR DARPA**
- (3) THE FIVE TECHNICAL PANELS WILL BE DESIGNATED AS FOLLOWS:**
 1. INTELLIGENT SYSTEMS — STATUS AND EXPECTATIONS
 2. ADAPTIVE KNOWLEDGE PROCESSING — THEORY
 3. SIMULATION/EMULATION — TOOLS AND TECHNIQUES
 4. SYSTEMS APPLICATIONS
 5. ADVANCED IMPLEMENTATION TECHNOLOGY
- (4) A FINAL STUDY REPORT WILL BE PREPARED WHICH WILL INCLUDE AN EXECUTIVE SUMMARY WITH BRIEFING CHARTS, SUMMARY REPORTS OF THE TECHNICAL PANELS AND A PROGRAM PLAN**

Figure 1-1. DARPA Neural Network Study Terms of Reference.

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

LINCOLN LABORATORY

LEXINGTON, MASSACHUSETTS 02173-0073

DARPA NEURAL NETWORK STUDY SYMPOSIUM

Room A-166

AGENDA

Thursday - 8 October 1987

0900 - 0905	Welcome	W. Morrow MIT/Lincoln Laboratory
0905 - 0915	Study Introduction	J. Lupo DARPA/TTO
0915 - 1015	Theory & Applications of Layered Neural Networks - Past & Future	B. Widrow Stanford University
1015 - 1045	Break	
1045 - 1130	Neural Mechanisms for Processing Visual Information	D. Hubel Harvard Medical School
1130 - 1215	The Interface between Neuroscience and Computer Science	E. Schwartz NYU Brain Research
1215 - 1300	Lunch	
1300 - 1345	Neural Network Architectures for Vision	S. Grossberg Boston University
1345 - 1430	Neural Networks and Combinatorial Optimization	J. Barhen Oak Ridge National Lab
1430 - 1515	Neural Networks for Optimization Problems	D. Tank Bell Laboratories
1515 - 1530	Break	
1530 - 1615	Optical Implementation of Neural Networks	B. Soffer Hughes Research Lab
1615 - 1700	Electronic Circuits for Neuromorphic Systems	J. Raffel MIT/Lincoln Laboratory

END OF FIRST DAY

Friday - 9 October 1987

0900 - 0945	Neurally Inspired Networks	D. Rumelhart Stanford University
0945 - 1030	Analyzing the Hidden Units in Multilayered Neural Networks	T. Sejnowski Johns Hopkins
1030 - 1045	Break	
1045 - 1130	Invariants of Neural Networks for Adaptive Pattern Recognition and Robotics	G. Carpenter Northeastern University
1130 - 1215	Neural Networks in Real World Applications: Biological and Computational Constraints	L. Cooper Brown University
1215 - 1300	Lunch	
1300 - 1345	Massive Parallelism in Nature and Computer Science	J. Feldman Univ. of Rochester
1345 - 1530	Study Organization and Objectives	A. Gschwendtner MIT/Lincoln Laboratory

END OF SYMPOSIUM

Figure 1-2. DARPA Neural Network Study Symposium Agenda.

- **Panel 4: System Applications**, which was chaired by Dr. Jon Leonard of Hughes Aircraft Co.; the MIT/LL associate was Dr. Michael Holz.
- **Panel 5: Advanced Implementation Technology**, which was chaired by Dr. Demetri Psaltis, California Institute of Technology; Dr. Jay Sage served as the MIT/Lincoln Laboratory associate.

The full membership of each of the panels is shown in Figure 1-3.

Dr. Jasper Lupo, of DARPA's Tactical Technology Office, was the Study sponsor. Walter Morrow Jr., Director of Lincoln Laboratory, chaired the Study's Steering Committee, whose membership included: Dr. Robert Fossum, Dr. George Heilmeier, and Dr. Steven Lukasik, ex-directors of DARPA who provided significant insights into the historical role played by DARPA in the development of computer science over the past two decades as well as guidance for the potential use of neural networks as a part of future computer systems; Nobel laureate Professor David Hubel, who has made major contributions to knowledge about vision and represented the important area of biological science within the neural network community; and representatives of industry, universities, the armed services, and the Department of Defense (see Figure 1-4). This committee periodically reviewed the work of the Study's panels and offered comment and direction.

Professor Bernard Widrow of Stanford University served as the Study Director. He was supported by several well-known technical consultants, including: Nobel laureate Professor Leon Cooper of Brown University, who shared the prize for the theory of superconductivity and has since become a leading proponent of neural network research and development; Professor Stephen Grossberg of Boston University, who has contributed greatly to the mathematical foundations of neural networks over the past two decades; and Professor Gail Carpenter of Northeastern University (See Figure 1-5).

The Study's participants maintained a rigorous schedule, meeting almost daily for a period of five months at its own facility in Bedford, MA and at a variety of other sites around the U.S. Members of the Study's panels listened to some 250 formal presentations and conducted several workshops. Among others, they heard from Professor Marvin Minsky of MIT in an AI workshop and from Nobel laureates Professor Gerald Edelman and, before his death, Professor Richard Feynman. Professor Edelman has developed a novel approach to neural networks based on the principle of neuronal group selection from his Nobel Prize winning work in genetics. Professor Feynman, with his inexhaustible interest in and enthusiasm for novel scientific approaches, raised three important questions that were repeated throughout the Study:

1. What good are neural networks?
2. How do you know that?
3. What other techniques are there to achieve the same objectives and how well do they work?

Former DARPA director Dr. Robert Cooper reviewed the history of the Strategic Computing effort and offered some sound advice concerning how the Neural Network Study should proceed. (See Figure 1-6.)

**PANEL 1
INTELLIGENT SYSTEMS —
STATUS & EXPECTATIONS**

Edward Posner, JPL — Chairman
Thomas Goblick, MIT/LL — Associate

John Daugman, Harvard University
Bart Kosko, Verac
A.F. Lawrence, Hughes Aircraft
Tomaso Poggio, MIT
Oliver Selfridge, GTE Laboratories
David Waltz, Thinking Machines, Inc.
Allen Waxman, Boston University

**PANEL 3
SIMULATION/EMULATION
TOOLS & TECHNIQUES**

Andrew Penz, TI — Chairman
Paul Kolodzy, MIT/LL — Associate

Michael Cohen, Boston University
Charles Elbaum, Brown University
Knut Kongelbeck, Hughes Aircraft
Martin Fong, SRI
Robert Hecht-Nielsen, HNC
Mike Myers, TRW
Douglas Palmer, HNC
Mark Watson, SAIC

**PANEL 2
ADAPTIVE KNOWLEDGE PROCESSING**

John Pearson, SRI — Chairman
Richard Lippmann, MIT/LL — Associate

Richard Andersen, MIT
Yaser Abu-Mostafa, CALTECH
Andrew Barto, U. Mass./Amherst
Jerome Feldman, Rochester University
Edward Finn, Georgetown University
Michael Kuperstein, Wellesley College
Ennio Mingolla, Boston University
John Moody, Yale Computer Science
Eric Schwartz, NYU Brain Research
Richard Sutton, GTE Laboratory
Ronald Williams, Northeastern

**PANEL 4
SYSTEMS APPLICATION**

Jon Leonard, Hughes — Chairman
Michael Holz, MIT/LL — Associate

Patrick Castelaz, Hughes Aircraft
Mark Coy, US ARMY/ETDL
William Dress, Oak Ridge National Lab
Charles Elbaum, NESTOR
Lynn Garn, Ctr. for Night Vision & EO
Edward Gliatti, Wright Patterson AFB
Ted Hoff, Consultant
William Miceli, ONR
Doyce Satterfield, USASDC
Charles Wagner, Wright Patterson AFB
Charles Woods, RADC

Figure 1-3. DARPA Neural Network Study Participants.

**PANEL 5
ADVANCED IMPLEMENTATION TECHNOLOGY**

**Demetri Psaltis, CALTECH — Chairman
Jay Sage, MIT/Lincoln Laboratory — Associate**

**Joshua Alspector, Bell Communications
Dana Anderson, University of Colorado
Dean Collins, Texas Instruments
Arthur Fisher, Naval Research Lab
Hans Graf, AT&T Bell Laboratories
Ted Hoff, Consultant
B. Keith Jenkins, USC
Greg Nash, Hughes Research Lab
Anil Thakoor, Jet Propulsion Lab
Bernard Soffer, Hughes Research Lab**

**PANEL 6
PROGRAM PLAN**

**Bernard Widrow, Stanford University
Al Gschwendtner, MIT/Lincoln Laboratory**

**Yaser Abu-Mostafa, CALTECH
Gail Carpenter, Northeastern University
Leon Cooper, Brown University
Thomas Goblick, MIT/Lincoln Lab
Steven Grossberg, Boston University
Ted Hoff, Consultant
Michael Holz, MIT/Lincoln Lab
John Hopfield, CALTECH
Paul Kolodzy, MIT/Lincoln Lab
Jon Leonard, Hughes Aircraft**

**Richard Lippmann, MIT/Lincoln Lab
William Miceli, ONR
John Pearson, SRI, David Sarnoff Ctr.
Andrew Penz, Texas Instruments
Ed Posner, Jet Propulsion Lab
Demetri Psaltis, CALTECH
David Rumelhart, Stanford University
Jay Sage, MIT/Lincoln Lab**

Figure 1-3. Continued.

W. MORROW, JR. MIT/LINCOLN LABORATORY, CHAIRMAN		
S. ANDRIOLE	GEORGE MASON UNIVERSITY	MIT/LINCOLN LABORATORY
E. BIZZI	MIT/BRAIN AND COGNITIVE SCIENCE	IBM
P. BLANKENSHIP	MIT/LINCOLN LABORATORY	ASN/RE&S
H. EISENHARDT	DARPA CONSULTANT	DARPA/ISTO
C. FIELDS	DARPA/DEPUTY DIRECTOR	NSA
R. FOSSUM	SOUTHERN METHODIST UNIVERSITY	ASSIST. SECRETARY OF THE ARMY
L. GILES	AFOSR	USA CONCEPTS/ANALYSIS AGENCY
A. GSCHWENDTNER	MIT/LINCOLN LABORATORY	AIR FORCE CHIEF SCIENTIST
G. HEILMEIER	TEXAS INSTRUMENTS	NAVAL RESEARCH LABORATORY
D. COLLINS	TEXAS INSTRUMENTS	CENTER FOR NIGHT VISION & EO
D. HUBEL	HARVARD MEDICAL SCHOOL	HUGHES AIRCRAFT CORPORATION
H. KOTTLER	MIT/LINCOLN LABORATORY	MARTIN MARIETTA CORPORATION
S. LUKASIK	NORTHROP CORPORATION	ONR
J. LUPO	DARPA/TTO	
	A. McWHORTER	
	S. KIRKPATRICK	
	R. RUMPF	
	J. SCHWARTZ	
	R. SCOTT	
	J.R. SCULLEY	
	T. SHOOK	
	H. SORENSON	
	H. SZU	
	P. TRAVESKY	
	T. WONG	
	S. ZEIBERG	
	S. ZORNETZER	

Figure 1-4. DARPA Neural Network Study Steering Committee.

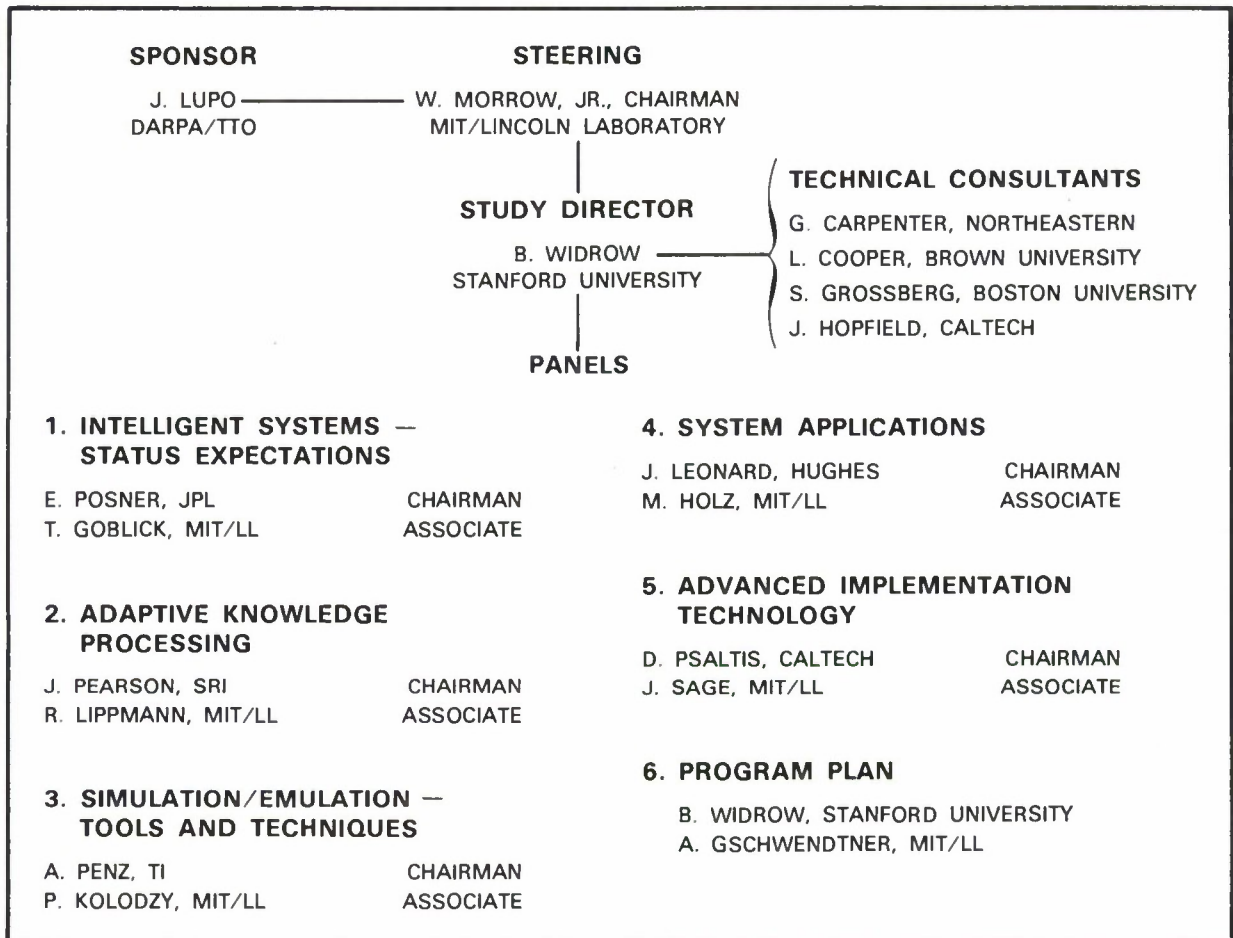


Figure 1-5. Neural Network Study Organization.

A complete list of all presentations made to all the panels can be found in *Appendix A: DARPA Neural Network Study List of Presentations*.

1.2 STRUCTURE OF THE DARPA NEURAL NETWORK STUDY TECHNICAL REPORT

The DARPA Neural Network Study Technical Report was prepared under the direction of Dr. Lupo of DARPA; Dr. Widrow, Study Director; and Mr. Gschwendtner, Associate Study Director. Its Technical Editor was Ms. Carol Weiszmann.

Parts II-VI of this document were prepared by the individual Study panel members under the supervision of their chairmen; the MIT/Lincoln Laboratory associates also served as editors.

- **Part II: Adaptive Knowledge Processing**, edited by Dr. Richard Lippmann, presents Panel 2's review of the theoretical underpinnings of the neural network field, covering neurobiology, mathematical theory, new parallel computer architectures, learning theory, and neural network algorithms.
- **Part III: Assessment of Neural Network Technology**, edited by Dr. Thomas Gobllick, reports on Panel 1's efforts to (a) put neural networks in the context of information processing technology and (b) compare neural networks with other approaches to information processing, including signal processing, communication and information theory, adaptive control systems, pattern recognition/classification, artificial intelligence, computer science, and optimization theory.
- **Part IV: System Applications**, edited by Dr. Michael Holz, contains Panel 4's survey of some 77 applications of neural networks encountered during the Study, as well as a more in-depth description of 11 of them.
- **Part V: Simulation/Emulation Tools and Techniques**, edited by Dr. Paul Kolodzy, describes Panel 3's assessment of the presently available hardware and software for simulation and emulation of neural networks.
- **Part VI: Advanced Implementation Technology**, edited by Dr. Jay Sage, discusses why implementation – as opposed to simulation – is an important matter in neural network research, articulates philosophical issues relating to advanced implementations of neural networks in technologies available now and in the foreseeable future, summarizes current implementation work, and presents Panel 5's conclusions.

Because the body of this Technical Report delineates the findings of the Neural Network Study's five separate panels, the reader will find in it an assortment of perspectives and opinions about neural network research efforts and their implications. Although the Study's panels functioned independently of each other, and although they each approached the topic of neural networks from distinctly different points of

	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	
OCTOBER 5						
OCTOBER 12	X	S	S	S	SYMPOSIUM	
OCTOBER 19	S	AI WORKSHOP	FEYNMAN/CALTECH	S	S	
OCTOBER 26	S	L. COOPER/BROWN	PANEL PREPARATION	STEERING/STUDY CTR.	S	
NOVEMBER 2	S	S	S	S	IMPLEMENTATION WORKSHOP	
NOVEMBER 9		IEEE CONFERENCE ON NEUROMORPHIC SYSTEMS IN DENVER, CO				
NOVEMBER 16	S	APPLICATION WORKSHOP				
NOVEMBER 23	S	S	X	X	S	
NOVEMBER 30		STEERING/STUDY CTR.	S	S	S	
DECEMBER 7		PANEL PRESENTATIONS IN SOUTHERN CALIFORNIA				
DECEMBER 14	R. COOPER/S-COMP	S	S	S	S	
DECEMBER 21	X	X	X	X	X	
DECEMBER 28	X	X	X	X	X	
JANUARY 4	X	APPLICATION WORKSHOP	SIMULATION WORKSHOP	THEORY WORKSHOP		
JANUARY 11	S	S	PANEL PREPARATION	STEERING/WASHINGTON	S	
JANUARY 18	S	VISION WORKSHOP				
JANUARY 25	S	S	PANEL PREPARATION	APPLICATION WORKSHOP		
FEBRUARY 1	S	EDELMAN/ROCKEFELLER UNIV.	S	STEERING/ENDICOTT	HUBEL/HARVARD	
FEBRUARY 8		REPORT PREPARATION				
FEBRUARY 15		REPORT PREPARATION				
FEBRUARY 22		STEERING PREPARATION				STEERING STUDY CTR.

Figure 1-6. DARPA Neural Network Study Schedule.

view and with different mandates, their conclusions – articulated at the end of each Part of this Technical Report – are noteworthy for their consistency and agreement.

In addition to this Technical Report from the Neural Network Study participants, a separate Executive Summary has been published; this document offers a synopsis of the Study's findings and articulates technical recommendations for further DARPA involvement in neural network research and development.

2. OVERVIEW OF THE STUDY'S FINDINGS AND CONCLUSIONS

A brief review of the Study's efforts and conclusions are presented below for those seeking a somewhat more cohesive albeit highly synoptic picture. Also, because the field of neural networks is new and its technical terminology unfamiliar to many, a glossary of neural network terms is included in *Appendix B: Neural Network Glossary of Terms*. The questions answered on the pages that follow include:

- Why did DARPA choose to conduct a technical study of neural networks?
- What did the Study determine to be the theoretical foundations of neural networks?
- What tools did the Study conclude are presently available to implement neural networks?
- What neural network applications did the Study find? How many? Are these applications well-developed or rudimentary?
- What advanced implementation technologies will be required for neural network development over the next five years?
- What are the Study's conclusions?

2.1 WHY NEURAL NETWORKS

Interest in neural networks as an alternative – that is, a non-Von Neumann – type of computing has been building for several years. Some have envisioned neural networks as an alternative to artificial intelligence (AI) – indeed, as a way to attack problems AI had been unable to solve. Others believe this new-found faith in neural networks to be naive. Between these two perspectives ranges a variety of views about the efficacy of neural networks.

By the summer of 1987, neural networks were attracting so much attention that some 2,000 people attended a summer meeting on the subject conducted by the Institute of Electrical and Electronics Engineers (IEEE), at which hundreds of papers were presented. Another IEEE conference followed in November, 1987; its 750 attendees considered it a scientific success. Another IEEE-sponsored conference on neural networks is scheduled for July, 1988.

On the commercial side, entrepreneurial spirit has enlivened the neural network community, and several start-up enterprises – some with significant venture capital backing – have emerged in the last year or so. They have identified commercial opportunities in a number of industries – including defense-oriented ones – and are particularly focused on offering systems and software which encourage users to educate themselves about and experiment with neural network ideas and concepts. Clippings of headlines from 1987 reports citing interest in neural networks are shown in Figure 2-1.

As the headlines in Figure 2-1 reveal, interest in neural networks has not been limited to the United States. Both Japan and the European nations have made commitments to neural network research: the

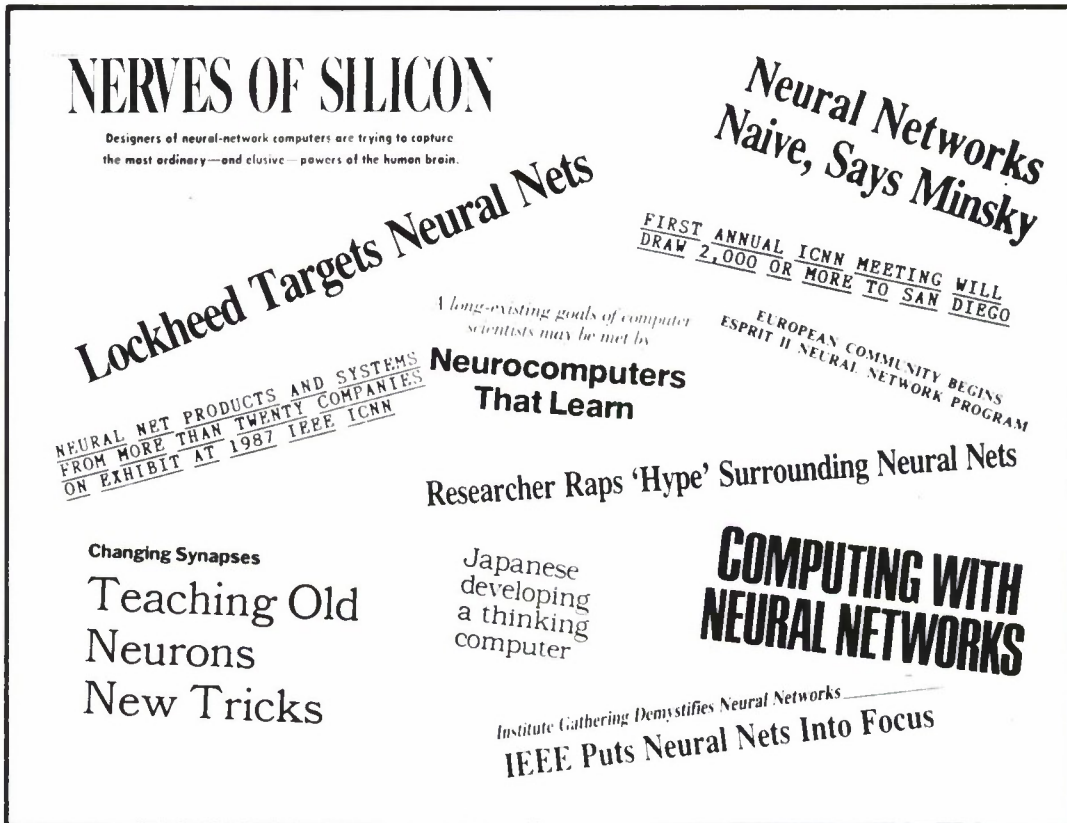


Figure 2-1. Clippings of Headlines from 1987 Reports Citing Interest in Neural Networks.

former has initiated a government-sponsored program, called "The Human Frontiers," to look at the biological origins of neural networks, and firms such as Fujitsu began developing "thinking computers" especially for robotic applications; the European countries, meanwhile, have begun a neural network-oriented program called Esprit II. Moreover, under the leadership of Stephen Grossberg in the U.S., Teuvo Kohonen in Finland, and Shun-Ichi Amari in Japan, the International Neural Network Society (INNS) was formed in the spring of 1987. The first issue of its journal, *Neural Networks*, appeared early in 1988 and its first conference is scheduled for the latter part of 1988; notably, six IEEE societies are co-sponsors of the INNS meeting, and financial support has come from the National Science Foundation, DARPA, and a number of other government agencies.

Some of this interest in neural networks is, perhaps, a result of the hyperbole which almost inevitably accompanies the use of phrases like "thinking computers" or "neurocomputers that learn." Hyperbole's natural habitat is, after all, the territory between achievement and promise: the greater the potential of a new and mostly unexplored avenue of research, the greater the human tendency to imagine it opening pathways to both utopia and dystopia.

Nevertheless, DARPA's Tactical Technology Office concluded that if neural networks are only *partially* as useful and powerful as suggested even by those whose views are considered conservative, then this alternative form of computing and its implications for defense systems require examination.

The Neural Network Study was undertaken to conduct this examination, which regards a neural network as a computational structure modeled on biological processes. Biological systems can easily solve problems that are very difficult for conventional computers to solve; a pigeon, for example, boasts an image processing system which is far superior to any capability of modern computer systems. (For a detailed and specific definition of 'neural network,' see *Appendix B: Neural Network Glossary of Terms and Part II: Adaptive Knowledge Processing, Chapters 1 and 2* of this Report.)

The Study's focus was the two key features which, it is widely believed, distinguish neural networks from any other sort of computing developed thus far:

- **Neural networks are adaptive, or trainable.** Neural networks are not so much programmed as they are *trained with data* – thus many believe that the use of neural networks can relieve today's computer programmers of a significant portion of their present programming load. Moreover, neural networks are said to improve with experience – the more data they are fed, the more accurate or complete their response.
- **Neural networks are naturally massively parallel.** This suggests they should be able to make decisions at high-speed and be fault tolerant.

The Study has sought to determine (a) to what extent neural networks are in fact adaptable/trainable and massively parallel, and (b) the implications of neural network capabilities, as they have actually been used thus far and as they may potentially be exploited in the future.

2.2 THEORETICAL FOUNDATIONS OF NEURAL NETWORKS

A neural network is a computational structure modeled on biological processes. At its most fundamental level, interest in neural networks is prompted by two facts: (a) the nervous system function of even a “lesser” animal can easily solve problems that are very difficult for conventional computers, including the best computers now available, and (b) the ability to model biological nervous system function using man-made machines increases understanding of that biological function.

2.2.1 A Biological Inspiration

Figure 2-2 illustrates the sort of biological neuron which has influenced the development of “artificial,” or computational, neural networks. (The neural structure shown here is generic; a lowly leech, for instance – as well as other aquatic creatures – has a neural cell structure identical to that of a human being, though the manner in which a leech’s neurons are connected to each other is much simpler than man’s.) Figure 2-2 shows two neurons in synaptic contact:

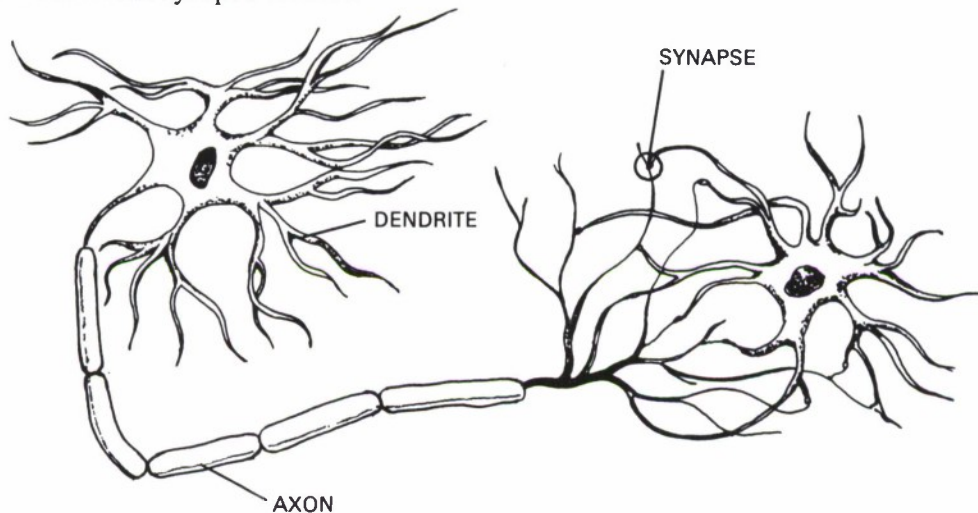


Figure 2-2. *Biological Neurons.*

- The *soma*, or nerve cell, which is the large round central body of the neuron, is anywhere from five to 100 microns in diameter;
- The *axon* is attached to the soma and is electrically active, producing the pulse which is emitted by the neuron;
- The electrically passive *dendrites* receive inputs from other neurons by means of a specialized contact – this is the *synapse*, which occurs where the dendrites of two different nerve cells meet.

The synapse is the tissue connecting neurons and it is capable of changing a dendrite's local potential in a positive or negative direction, depending on the pulse it transmits. Note that these transmissions occur in very large numbers, but, since they are chemical, they occur fairly slowly.

The human cerebral cortex, for instance, is comprised of approximately 100 billion (10^{11}) neurons with each having roughly 1,000 dendrites that form some 100,000 billion (10^{14}) synapses; given that this system operates at about 100 Hz, it functions at some 10,000 billion (10^{16}) interconnections per second. It weighs approximately three pounds, covers about 0.15 square meters, and is about two millimeters thick. This capability is clearly beyond anything which can be reconstructed or modeled; but it is, perhaps, possible to understand how the brain performs information processing, and it is hoped that this *understanding* can be modeled and ultimately implemented in hardware.

(For a more detailed description of the role of neurobiology plays in neural networks – including an overview of brain physiology as well as discussions of the biological foundation of neural networks and recent results in neurobiology – see *Part II, Chapter 13: Neurobiology and Neural Networks* of this Report.)

2.2.2 The Modest Analogy

“Artificial” neurons, as illustrated in Figure 2-3, are analogous to their biological inspirers. Figure 2-3 presents the simplest artificial neuron configuration. Here neurons become *processing elements*, the axons and dendrites become wires, and the synapses become variable resistors carrying *weighted* inputs that represent data or the sums of weights of still other processing elements.

These inputs – which are voltages that are proportional to weights that have been established – are summed across the resistor. The resistor is connected to an operational amplifier on which has been set a threshold so that when the sum of these voltages reaches a pre-set threshold, the neuron – or processing element – will fire. Figure 2-3 assumes this processing element to be a hard-limiting device: that is, when the sum of the voltages is below threshold, the device will have a -1.0 output and not fire; when the sum reaches threshold, its output will be +1.0 and the device will fire.

Processing elements can interact in many ways by virtue of the manner in which they are *interconnected*. Prosaic as it is, Figure 2-3 actually suggests a variety of possibilities:

- Processing elements which *feed forward* only; processing elements which have a *feedback* loop.
- Processing elements that are *fully connected* to all other processing elements; processing elements that are *sparsely connected*, linked only to a few others.

The nature and number of these feedback loops and connections depend on the model, or architecture, used to construct the neural network. The design of a neural network's feedback loops has implications for the nature of its adaptivity/trainability, while the design of a network's interconnections has implications for its parallelism.

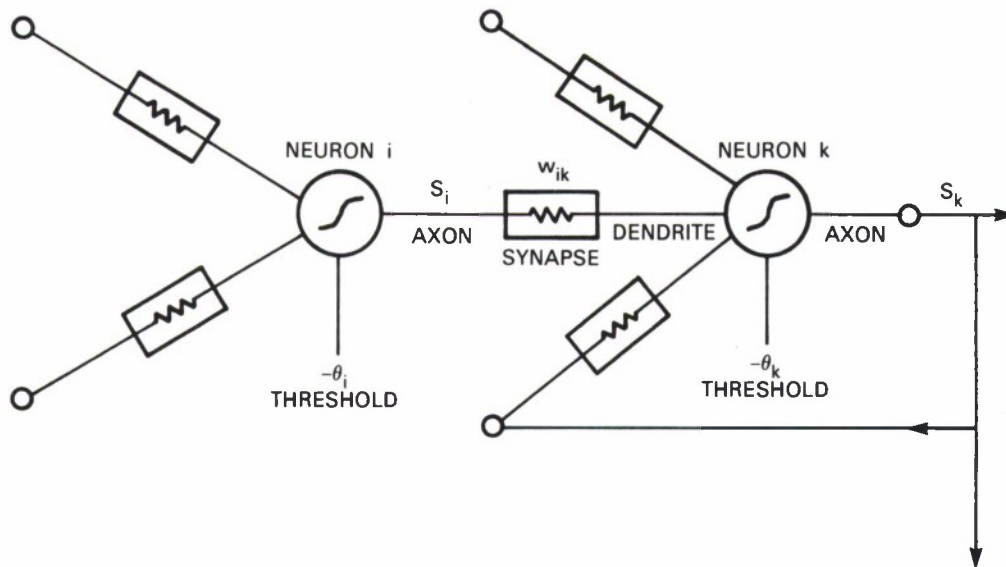


Figure 2-3. Artificial Neurons.

2.2.3 How A Neural Network Learns

A key neural network feature – **trainability** or **adaptivity** – has been demonstrated to the Study. Figures 2-4 and 2-5 illustrate the learning capabilities of a neural network, though in an extremely simplified way.

There are many techniques – generally articulated as algorithms – used to train neural networks; they fall into three basic categories:

- **Supervised training** requires the presence of an external teacher and labeling of the data used to train the network. The teacher knows the correct response wanted from the network and inputs an error signal when the network produces an incorrect response. The error signal “teaches” the network the correct response in a process that is explained below, and after a succession of learning trials the network consistently produces the correct response.
- **Unsupervised training** uses unlabeled training data and requires no external teacher; data is presented to the network, which forms internal clusters that compress the input data into classification categories.
- **Self-supervised training** is used by certain kinds of neural networks. The networks monitor performance internally, requiring no external teacher; an error signal is generated by the system and fed back to itself, and the correct response is produced after a number of iterations.

Figure 2-4 illustrates in a simplistic way how *supervised* neural network training works. Here, a single neuron is being trained to turn on a light when the sum of two input voltages is greater than 0.5 and turn

the light off when the sum is less than 0.5. The neuron will be trained by adjusting the weights W_1 and W_2 , corresponding to input voltages X_1 and X_2 , and a threshold, θ_1 . The sum of the inputs and the output state are shown on the right of Figure 2-4.

The supervised training process shown in Figure 2-4 works, through the course of several training trials, as follows:

- In the first trial, the inputs add up to less than 0.5, so the light should be off. However, the weight settings and threshold are such that the light is on, so a teacher (supervisor) adjusts the weights and threshold to turn off the light.
- The next trial's inputs sum to greater than 0.5, so the light should be on. But the light is actually off because the teacher overcorrected the weight and threshold adjustments on the previous trial. The weights and threshold are therefore again readjusted by the teacher to turn the light on.
- The third inputs' sum is negative and the light turns off, which is correct – so no adjustments are made.
- The next trial results in inputs greater than 0.5 – and the weights and threshold are now correctly adjusted so that the light turns on.

This exercise can be summarized by noting that this simple neural network has learned to identify a line in two-dimensional space – as shown in the lower left of Figure 2-4 – given by $X_1 + X_2 = 0.5$. Correspondingly, with *three* inputs the network could be trained to identify a plane in *three-dimensional* space; further inputs could allow a neural network to identify the equation of a hyperplane in a higher-dimensional space. This geometrical analogy can be extended to many *layers of processing elements* in a neural network, as shown in Figure 2-5.

Here, again, the single-layer neural network – composed of three processing elements – provides a half-plane solution, so it can separate regions A and B in a two-dimensional space. If a *second layer* of processing elements is added, each neural layer can classify a half-plane – and this amounts to a solution of the “exclusive OR” problem. The addition of a third processing element in the second layer allows for dividing a region into three half-planes – which provides a convex region. If a *third layer* of processing elements is added, two convex regions can be joined – thus arbitrary region formation is possible with a three-layer neural network.

Since geometrical regions are the equivalent of classification regions, a neural network can be trained (a) to search for and identify the many primitives that define a complex input space, and (b) to group these primitives so that classifications or identifications can be made automatically.

2.2.4 Massive Parallelism in Neural Networks

As noted in Figure 2-6, a network of $64 \cdot 64$ elements that are fully interconnected can undertake millions of simple parallel operations aimed at the dissection of a problem. Figure 2-6 shows a neural

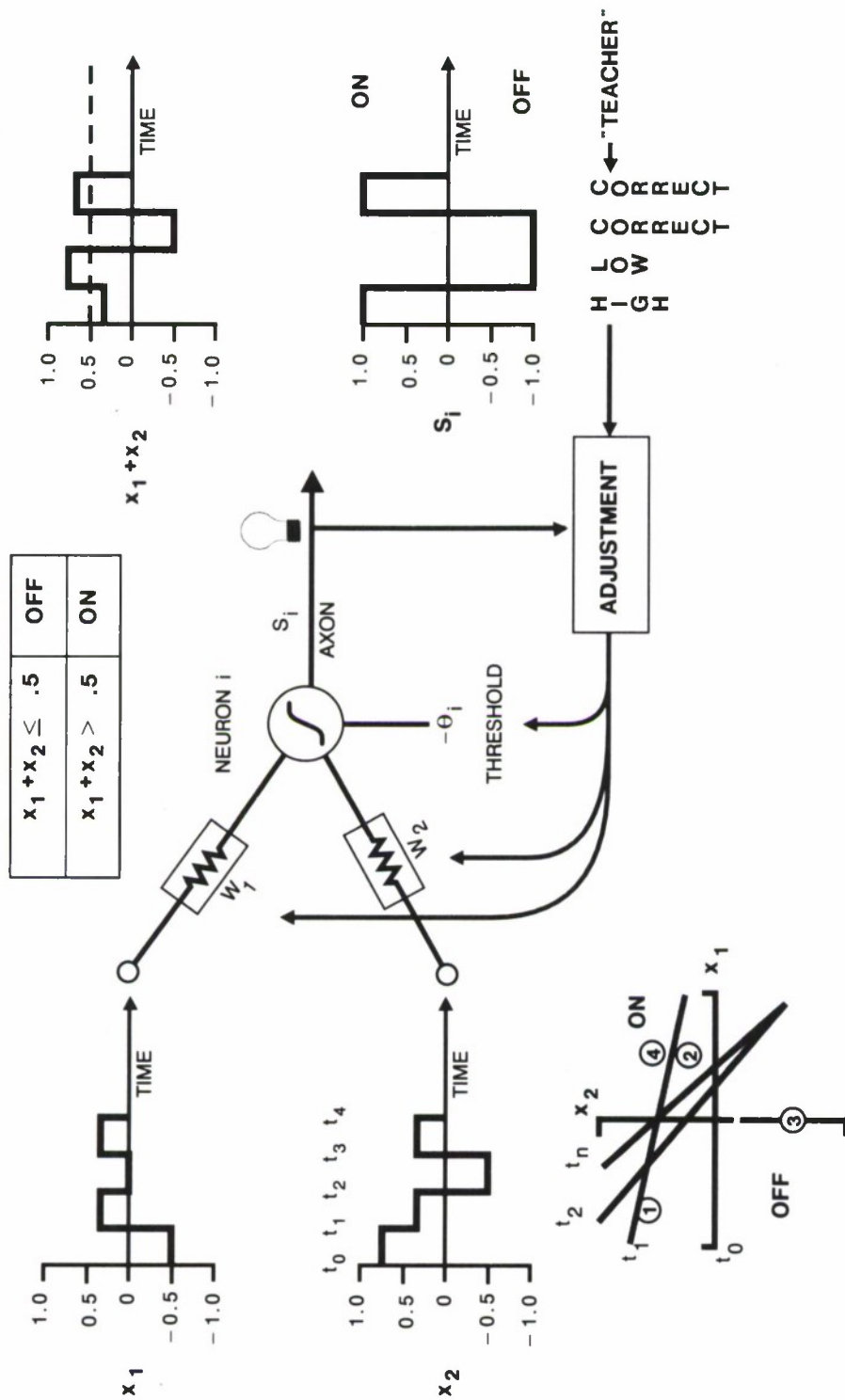
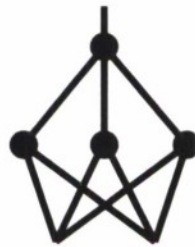


Figure 2-4. Single Neuron Learning.

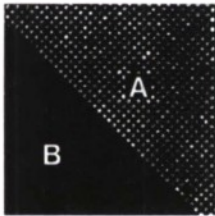
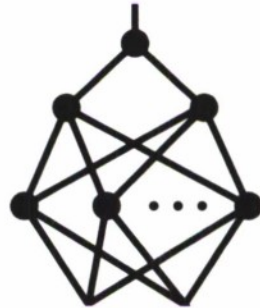
ONE-LAYER



TWO-LAYERS

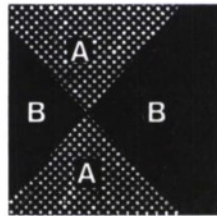


THREE-LAYERS



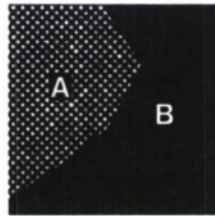
1 HALF PLANE

HALF PLANE

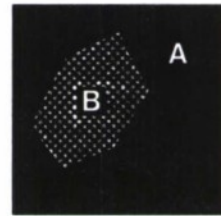


2 HALF PLANES

TYPICALLY CONVEX



3 HALF PLANES



TWO REGIONS

ARBITRARY

Figure 2-5. Computational Power of Single- and Multi-layer Networks.

network with layers M and N; each layer has 4,096 processing elements laid out in a 64 · 64 grid. If these two layers are fully interconnected – that is, if every processing element is linked to every other processing element – then this apparently simple structure sorts through no less than 16 million interconnects.

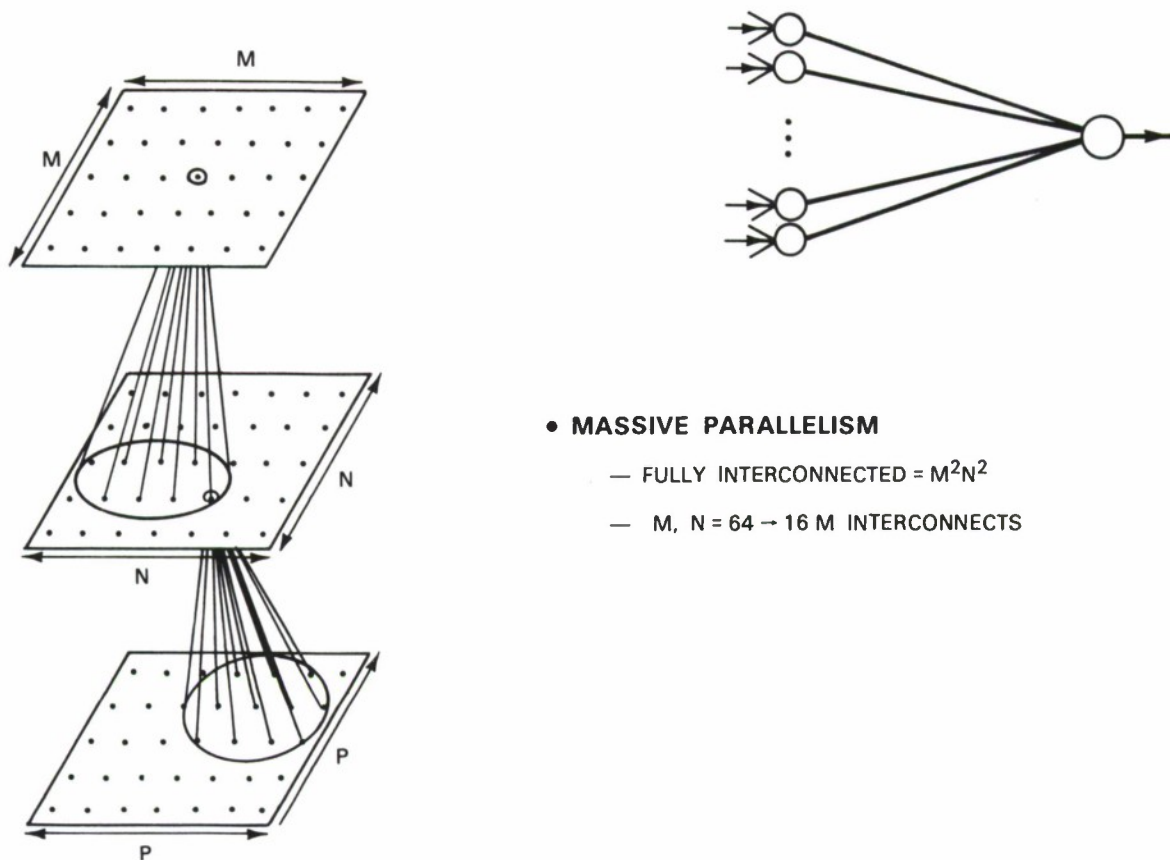


Figure 2-6. Massive Parallelism of a Neural Network.

It is this pyramiding of many simple operations to automatically provide complex classifications that is the core of the neural network promise.

(Part II, Chapter 3: Tasks Neural Networks Perform and Representative Models of this Report offers a more detailed look at tasks at which neural networks are effective as well as illustrative neural network models and a review of important training and performance issues.)

2.2.5 The Importance of Surviving the Single-layer Perceptron

Neural network research is not new – it is, rather, newly revived from an obscurity and even disrepute which is now understood to have been undeserved.

Work in neural networks, which began nearly 50 years ago, falls into three eras, as described in Figure 2-7. The early efforts, in which basic thinking about neural networks was accomplished by McCulloch and Pitts, Hebb, Rosenblatt, and others, took place in the 20 years before 1960.

Then a schism in the AI research community occurred involving Rosenblatt's single-layer **perceptron**. Minsky and Papert of MIT pointed out that the perceptron could not solve the "exclusive OR" class of problems, for reasons already explained, whereupon it and neural network research in general were largely abandoned by DARPA in favor of work in the apparently more promising realm of symbolic processing.

(For a lengthier discussion of single- and multi-layer perceptrons, see *Part II, Chapter 4: Single- and Multi-layer Perceptrons* of this Report.)

This loss of support forced neural network researchers into the "wilderness years" of 1960-1980, which – despite a lack of money or support – saw dedicated people develop a variety of mathematical theories which laid the foundations for a resurgence of interest and effort in neural networks that has been underway since the early 1980s. This resurgence was also fueled by other crucial factors:

- Understanding of the deficiencies of the single-layer perceptron and the extension of theoretical work into multi-layer systems;
- The revolution in computer technology, which produced powerful and comparatively inexpensive computing devices and diagnostic tools that enabled further work on neural network mathematical theories;
- Simultaneous breakthroughs in the understanding of neurobiological processes.

2.2.6 Neural Networks and Other Information Processing Approaches

It is helpful to an understanding of neural networks and their implications to place them in the larger context of information processing approaches, for neural networks do not oppose other approaches as much as they extend them.

Figure 2-8 delineates the major thrusts in information processing in a historical sense, beginning with system and information theory, which is really about *database processing*. This was followed by the development artificial intelligence techniques for *knowledgebase processing*. It is certainly worth noting that knowledgebase processing has not by any means eclipsed database processing; they are, in fact, quite complementary.

Neural networks have been touted as effective in tackling problems in, among other areas, machine vision, robotics, and speech understanding. But it is highly unlikely that neural networks will replace either knowledgebase or database processing. They will, the Study expects, find a niche within the whole scheme of information processing. The nature of the neural network niche becomes more apparent when neural networks as an information processing approach is compared to database and knowledgebase processing.

EARLY WORK 1940-1960 FUNDAMENTAL CONCEPTS	McCULLOCH & PITTS HEBB FARLEY & CLARK RONSENBLATT STEINBUCH, TAYLOR	— BOOLEAN LOGIC — SYNAPTIC LEARNING RULE — FIRST SIMULATION — PERCEPTRON — ASSOCIATIVE MEMORIES
TRANSITION 1960-1980 THEORETICAL FOUNDATIONS	WIDROW & HOFF ALBUS ANDERSON VON DER MALSBURG FUKUSHIMA GROSSBERG	— LMS ALGORITHM — CEREBELLUM MODEL (CMAC) — LINEAR ASSOCIATIVE MEMORY — COMPETITIVE LEARNING — NEOCOGNITRON — ART, BCS
RESURGENCE 1980- THEORY BIOLOGY COMPUTERS	KOHONEN FELDMAN & BALLARD HOPFIELD REILLY, COOPER et al. HINTON & SEJNOWSKI RUMELHART et al. RUMELHART & McCLELLAND EDELMAN, REEKE	— FEATURE MAP — CONNECTIONIST MODELS — ASSOCIATIVE MEMORY THEORY — RCE — BOLTZMANN MACHINE — BACK PROPAGATION — PDP BOOKS — DARWIN III

Figure 2-7. History of Neural Network Research.

APPROACH	METHOD	KNOWLEDGE-ACQUISITION	FORM OF IMPLEMENTATION
SYSTEM & INFORMATION THEORY	MODEL DATA, NOISE, PHYSICAL CONSTRAINTS	ANALYZE MODELS TO FIND OPTIMAL ALGORITHM	HARDWARE IMPLEMENTATION OF ALGORITHM
AI EXPERT SYSTEMS	EMULATE HUMAN EXPERT PROBLEM SOLVING	OBSERVE HUMAN EXPERTS	COMPUTER PROGRAM
TRAINABLE NEURAL NETS	DESIGN ARCHITECTURE WITH ADAPTIVE ELEMENTS	TRAIN SYSTEM WITH EXAMPLES	COMPUTER SIMULATION OR NN HARDWARE

Figure 2-8. Comparison of Information Processing Approaches.

The expert system developer is interested in emulating a human expert's way of solving a specific set of problems, so he/she observes and analyzes a particular human expert(s), and then models that "intelligence," or expertise, in a computer program which explicates the rules (heuristic and otherwise) used by the human expert.

More generically, the system and information theorist models data, analyzes those models seeking the optimal algorithm, and then implements that algorithm in hardware.

If, for instance, the object of a hypothetical vision system is to find a cat in a backyard (as depicted in Figure 2-9), current information processing techniques would require that the data containing the visual information be extracted from a digitizing preprocessor and entered into a computer; this data would be subject to a rather human-intensive analysis, followed by another human-intensive period of algorithm development. The algorithm would be implemented on yet another computer which would then be able to find the cat. The same system could *not* be used if, perhaps, the goal is now to recognize a mouse; instead, the entire data-gathering, data analysis, algorithm development, and design implementation process must be repeated.

(A more extended discussion of "traditional" information processing, and particularly knowledge-based systems, may be found in *Part III: Assessment of Neural Network Technology, Chapter 2: Methodology* of this Report.)

Neural network designers claim, by contrast, to place the "intelligence" of the network in its architecture and adaptation rules, which are optimized not to a single problem or application, but to an entire *class* of problems. The network is trained, either with supervision or in unsupervised mode, with examples. The neural network is then implemented either by simulating it on conventional computer hardware or in special-purpose neural network hardware.

Thus, a hypothetical neural network implementation of the system looking for the cat in the backyard (Figure 2-10) may not even require the digitization of data; it can, perhaps, be entered directly from a sensor into the network in analog form. The neural network might then be trained by voice command to recognize the cat. Unlike its conventional information processing counterpart, the *same neural network* can also be trained to recognize a mouse.

This comparison suggests that neural networks can save substantial amounts of time and human effort which, in conventional information processing approaches, are now devoted to data analysis and algorithm development. But the Study resists such conclusions, for at least two reasons:

- Too little is presently known about training time requirements as neural networks are scaled up for large, real-world problems. If this training time is extended and requires human supervision, it could rival the time and effort demanded by conventional approaches.
- It is possible that, at least for the next several years, the time and effort required to develop neural network architectures might rival that of conventional algorithm development.

(*Part III, Chapter 3: Comparison of Neural Networks With Other Technologies* of this Report offers a detailed comparison of neural networks and other computational approaches in handling several types of

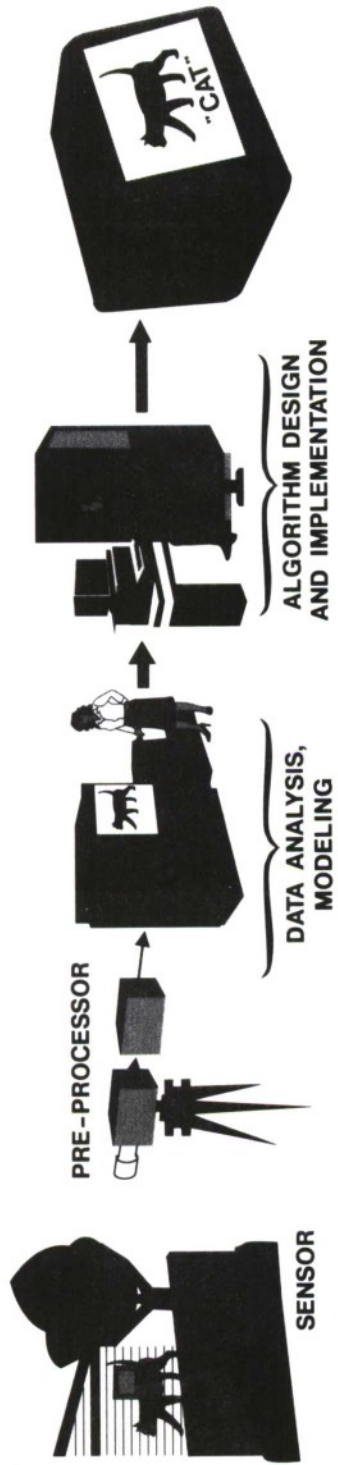


Figure 2-9. Current Information Processing Approach.

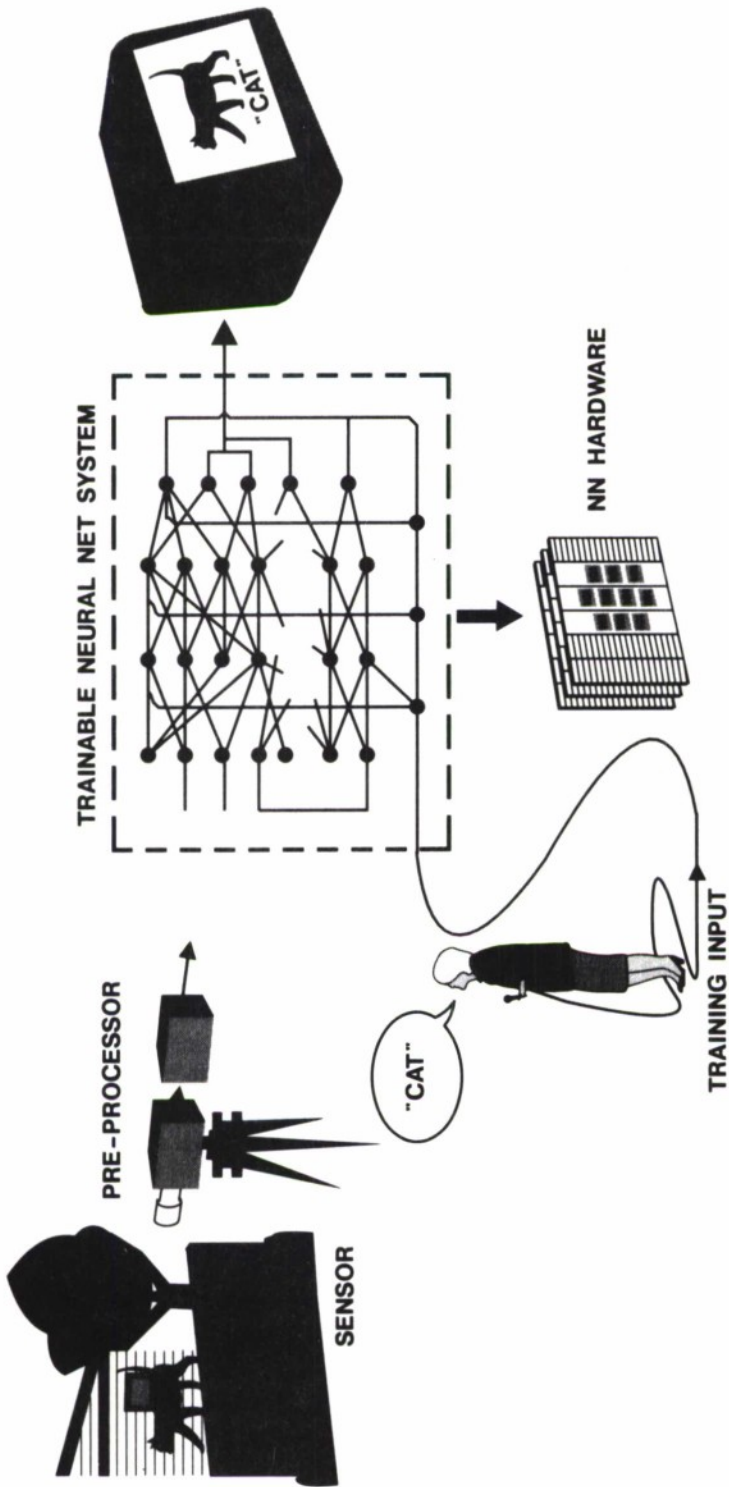


Figure 2-10. Neural Network Approach.

problems, including the traveling salesman problem, associative memories, pattern classification applications, computational maps, signal processing, speech applications, and machine vision.)

That is not to say that substantial work has not been done on neural network architectures, though this early work on theoretical models has tended to concentrate on some problem types more than others. Pattern classification has seen perhaps the greatest amount of neural network effort to date; but other types of problems – including machine vision, speech recognition, robotics, signal processing, and optimization – are also being taken on by neural network researchers, who have produced a variety of models to deal with them, as Figure 2-11 makes clear. The field of neural networks is in fact very rich; it's not just for perceptrons anymore.

(Detailed descriptions of the many types of neural network models and architectures, as well as the problems they address, can be found in *Part II, Chapters 4-10* of this Report).

PATTERN CLASSIFICATION	SPEECH RECOGNITION	MACHINE VISION	ROBOTICS	SIGNAL PROCESSING	OPTIMIZATION/COMPUTATION
ADALINE (LMS)	MARTIN SPEECH PREPROCESSOR	BCS/FCS	CMAC CEREBELLUM MODEL	ADALINE	CELLULAR AUTOMATA
ART	MASKING FIELD	DARWIN II	COMPUTATIONAL MAPS	MULTI-LAYER PERCEPTRON	HOPFIELD NET
BOLTZMANN MACHINE	SILICON COCHLEA	MARKOV RANDON FIELD	DARWIN III		
COMPETITIVE LEARNING	SYNAPTIC TRIAD	NEOCOGNITRON	MULTI-LAYER PERCEPTRON		
FEATURE MAP	VITERBI NET	PARAMETER NET			
MULTI-LAYER PERCEPTRON (Back Propagation)		SILICON RETINA			
PERCEPTRON					
REDUCED COLOUMB ENERGY (RCE)					

101450-15

Figure 2-11. Neural Networks for a Variety of Tasks.

2.2.7 Conclusions Concerning Theoretical Foundations of Neural Networks

From its examination of the theoretical foundations of neural networks, the Study has determined that:

- **Neural networks offer important new approaches to information processing** because of their adaptivity and ability to learn as well as their massive parallelism.
- **Neural network research has matured greatly since the perceptron of the 1950s.** This maturation has three sources: (a) advancement in mathematical theories, (b)

development of new computer tools, and (c) increased understanding of neurobiology. There is bridge forming between the biologically-oriented neural modelers and the artificial neural network modelers, and substantial progress will result from this alliance.

- **There is a need for research that focuses on training, scaling, and performance criteria.** Mathematical theories about training neural networks exist now for networks with only modest numbers of processing elements, but researchers are interested in scaling up to neural networks containing millions of processing elements. So the mathematical foundations concerning how the robustness of large, interconnected neural networks varies with signal, noise ratios, etc., need to be explored. Performance criteria are required to indicate how and where neural networks are most effective, and those conclusions need to be proven employing realistic databases addressing important problems.

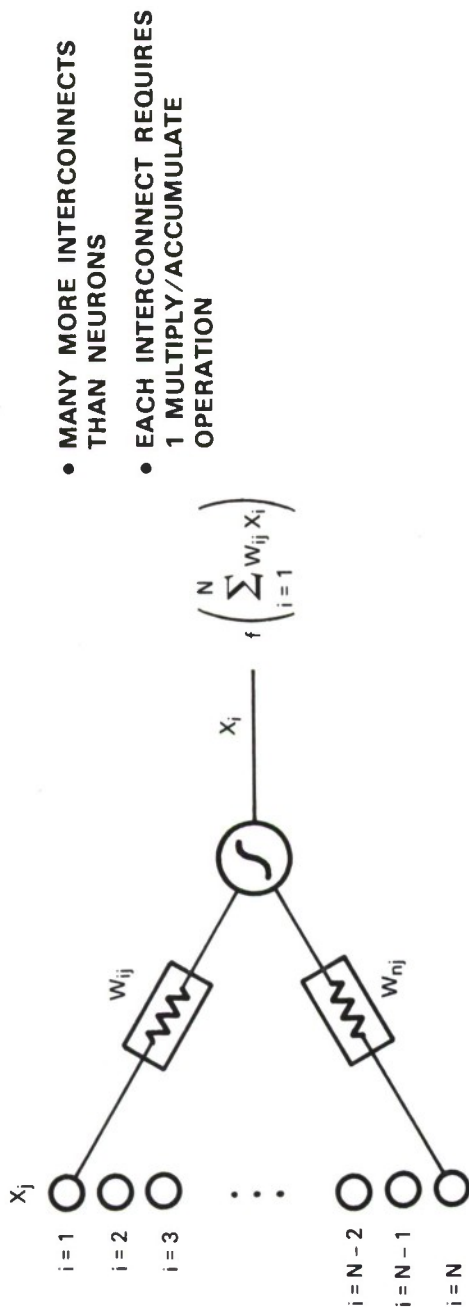
(The conclusions of Study's Technology Assessment Panel (Panel 1) and the Adaptive Knowledge Processing Panel (Panel 2) are delineated in this Report in *Part III, Chapter 4: Conclusions Concerning the Assessment of Neural Network Technology* and *Part II, Chapter 14: Conclusions Concerning Adaptive Knowledge Processing*, respectively.)

2.3 AVAILABLE IMPLEMENTATION TOOLS

The field of neural networks, as might be expected, has developed its own computational vernacular, some understanding of which is necessary to appreciate neural network simulation and implementation requirements. Some of the key concepts and terms (see Figure 2-12) are these:

- A typical neural network contains many more interconnects than neurons, or processing elements.
- Each interconnect requires one multiply/accumulate operation for summing.
- While digital computers are normally assessed in terms of storage or memory (where the unit of measure is words) and speed (instructions-per-second or floating-point-operations-per-second), the neural network vernacular defines **storage** as *the value of the input weights* and measures it in terms of *interconnects*; neural network **speed** is described in terms of *interconnects-per-second* within a layer or between layers. (This way of conceiving neural network storage is important only in the case of network simulation; in the case of implementation in special-purpose hardware, storage would be handled by resistive networks and would be defined differently.)

(A more detailed discussion of the computational requirements of neural networks can be found in *Part V: Simulation/Emulation Tools and Techniques, Chapter 2: Algorithm and Solution Requirements and Chapter 3: Application Computational, Requirements* of this Report.)



	STORAGE (memory)	SPEED (operations/s)
DIGITAL COMPUTER	WORDS	IPS, FLOPS
NEURAL NETWORK	INTERCONNECTS	INTERCONNECTS/s

Figure 2-12. Neural Network Computation Units.

To understand both this neural network computational vernacular and its implications for neural network tools and applications, the Study has used the terms *interconnects* and *interconnects-per-second* to chart a set of coordinates, placing interconnects on the abscissa and interconnects-per-second on the ordinate. The environment defined by this chart will reappear throughout the remainder of this summary to describe the computational capabilities of present and future neural network systems and applications.

The Study introduces this environment to chart the computational capabilities of certain biological systems, as can be seen in Figure 2-13. Significantly, man – with the staggering capacity of 10^{14} interconnects and 10^{16} interconnects per second – is beyond the range of the chart for reasons which will shortly be apparent. Within the range of Figure 2-13 are such creatures as the worm, the fly, and the bee. “Lowly” as a fly or a bee may seem, neural network researchers would be very pleased indeed to replicate their capabilities with a machine.

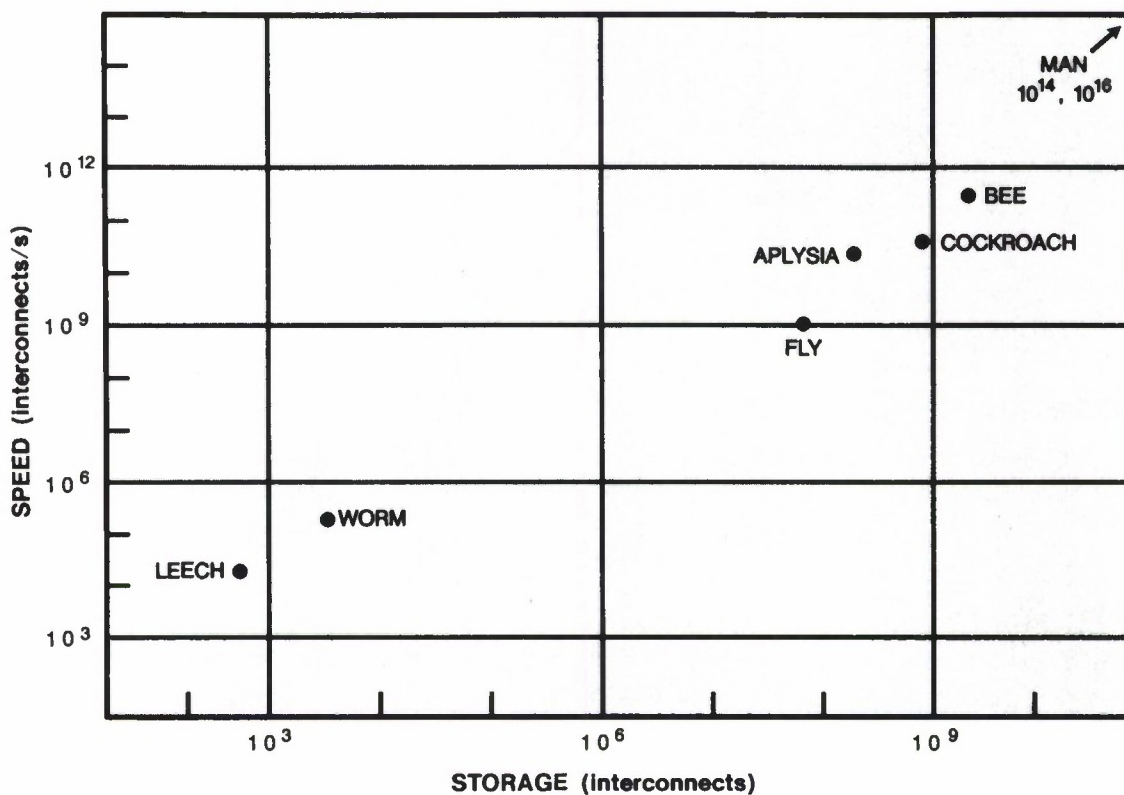


Figure 2-13. Computational Capabilities of Biological Networks.

2.3.1 A Simulated World

That will be difficult with the tools presently available. Figure 2-14 delineates the variety of tools to which today’s neural network researchers have access. They range from the low-priced microprocessor-based workstations, through attached processors and bus-oriented processors, to the more expensive

massively-parallel machines (including the Connection Machine developed by DARPA), and finally to supercomputers at the very high end.

Generally, the micro/minicomputers provide a very modest interconnects-per-second capability, though in some cases storage capacity is substantial. The speeds of these devices are limiting; neural network models take a very long time to run on them. Attached processors improve this situation somewhat, since they can boost interconnects-per-second into the millions from the hundreds of thousands.

Bus-oriented processors, in some cases, raise by an order of magnitude the number of interconnect-per-second available, but storage is not equivalently greater. The MX-1/16 shown in Figure 2-14 is actually an experimental architecture that is only partially proved but should be able to deliver around 120 million interconnects per second, along with 50 million interconnects for storage – this is quite good for meeting today's neural network simulation needs.

The massively parallel systems feature no better speed (interconnects-per-second) than the MX-1, and there remain gaps between their speed and storage capabilities. Supercomputers, meanwhile, do not offer significantly more capability than systems which cost far less.

In addition, the programming necessary to run neural network models on these massively parallel machines is very complex – it is, in fact, a problem which limits the extent to which these systems can be used to conduct neural network research. Moreover, the architectural limitations of these systems prevent researchers from stacking up several of them to significantly boost their storage or speed.

Thus the system most useful to neural network research is one configured like the still-experimental MX-1, not only because of its storage and speed – neural network efforts in fact will soon require even more than this – but because of the need for user-friendly systems that allow researchers to test a variety of neural network ideas against a variety of databases in timely fashion.

(Part V, Chapter 3: Existing Hardware and Software of this Report conducts a lengthier discussion of presently available hardware and software tools for neural network simulation. In addition, detailed descriptions of some neural network simulation hardware may be found in this Report in Part V, Appendix A: Parallel Processing Hardware.)

Figure 2-15 charts the presently available simulation tools onto the same set of interconnect and interconnect-per-second coordinates upon which the fly and bee were situated in Figure 2-13, with an additional distinction between those systems priced less than \$100,000 – and therefore more widely available as well as much easier to program – and those costing more than \$100,000 (the asterisk at the outer edge of the present computing capability shown in Figure 2-15 represents the experimental MX-1/16).

2.3.2 Conclusions Concerning Today's Neural Network Tools

Upon reviewing the implementation tools presently available to the neural network research community, it has become clear to the Study that:

- **The computational units of neural networks are best understood in terms of interconnects (storage) and interconnects-per-second (speed).**

WORKSTATIONS		INTERCONNECTS	INTERCONNECTS/s
MICRO/MINI COMPUTERS	PC/AT	100K	25K
	SUN 3	250K	250K
	VAX	32M	100K
	SYMBOLICS	10M	35K
ATTACHED PROCESSORS	ANZA	500K	45K
	DELTA 1 *	1M	10M
	TRANSPUTER *	2M	3M
BUS ORIENTED PROCESSORS	MARK III, V	1M	500K
	ODYSSEY	256K	20M
	MX-1/16 *	50M	120M
MASSIVELY PARALLEL	CM-2 (64K)	64M	13M
	WARP (10)	320K	10M
	BUTTERFLY (64)	60M	8M
SUPERCOMPUTERS	CRAY XMP 1-2	2M	50M

* PROJECTED PERFORMANCE

Figure 2-14. Examples of Neural Network Simulation Hardware.

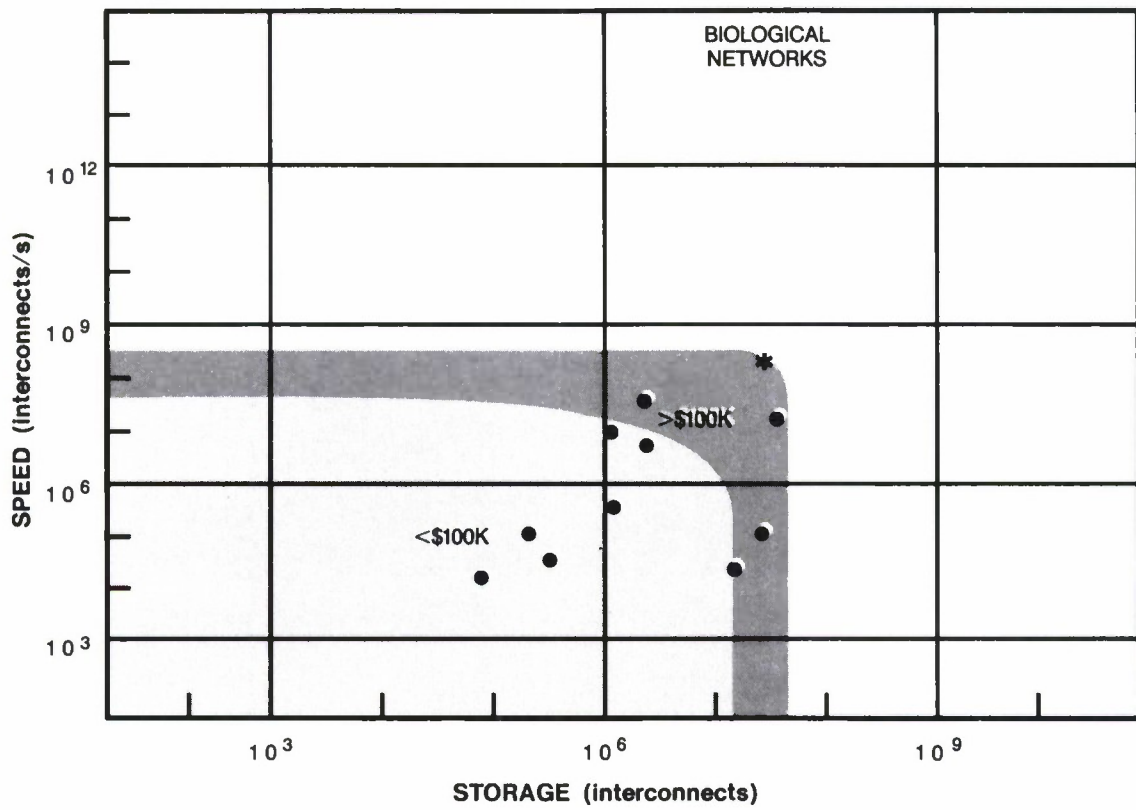


Figure 2-15. Computational Capabilities of Neural Network Simulators.

102520-3

- **Today's computer tools – with simulation capabilities of roughly 10^7 interconnects per second – fall far short of the computational capabilities of even modest biological networks;** a fly, for instance, computes at the rate of 10^9 interconnects per second.
- Although the computer revolution of the last 20 years has played a critical role in the revival of neural network research by making it possible to undertake much work in mathematical theory, **the limits of today's computing devices – inadequacies in storage, speed, and user flexibility – have restricted present neural network efforts.**

(The conclusions of the Study's Simulation/Emulation Tools and Techniques Panel (Panel 3) are articulated in *Part V, Chapter 6: Conclusions Concerning Neural Network Simulation/Emulation Tools* of this Report.)

2.4 STATUS OF CURRENT NEURAL NETWORK APPLICATIONS

The Study heard from researchers who had developed 77 applications in an assortment of areas, including vision, signal processing, classification, robotics, speech, and sensor fusion (see Figure 2-16). Many were redundant, others were comparatively uninteresting. Of the 77 applications reviewed by the study, 11 which were considered typical of the state of the neural network applications art were selected for more detailed review, and several of those are described briefly below.

(*Part IV: System Applications, Chapter 2: Summary of System Applications* of this Report contains a more detailed summary of the neural network applications presented to the Study, and descriptions of the 11 highlighted applications, written by the researchers who developed them, comprise *Part IV, Appendices A-K*. Also, a complete list of application contributors as well as brief, one-page summaries of all of the applications presented can be found in *Part IV, Appendix L: Presentations to the Applications Panel* of this Report.)

Figure 2-17 displays the history of the 11 applications reviewed by the Study, which is in large measure typical of the present state of the neural network art. Clearly, while much time between 1960 and now has been devoted to modeling applications, relatively few fielded applications have emerged; in only a few cases were models even demonstrated. Of the 11 applications chosen for closer review by the Study, just four of them have been fielded:

- The Risk Analysis system, developed by Nestor Inc. using the reduced Coulomb energy (RCE) neural network model, handles the functions of a loan officer in a financial institution. The system is trained in ongoing fashion using both the successes and failures of loan officers as well as ancillary information about the characteristics of people who repay their loans and those who default. It has been shown to be as effective as the best human loan officer in spotting bad risks.

VISION	31	— IMAGE CLASSIFIER — TARGET RECOGNIZER
SPEECH	6	— NET TALK — WORD RECOGNIZER
CLASSIFICATION	14	— RISK ANALYSIS
SIGNAL PROCESSING	17	— SONAR CLASSIFIER — TARGET TRACKING — ADAPTIVE CHANNEL EQUALIZER — PROCESS MONITOR
ROBOTICS	8	— FORKLIFT
SENSOR FUSION	2	— IMAGE FUSION
	78	

Figure 2-16. Application Survey.

- The Process Monitor is a single-layer neural network being installed by GTE in a fluorescent lightbulb manufacturing facility. The network monitors the activities of the production line, using inputs from a variety of sensors to infer when the line is operating correctly, and shuts the line down if something is wrong.
- The Word Recognizer is a small application of limited capability – its vocabulary is 100 words – built by Intel Corp.; it has enjoyed commercial success.
- The Adaptive Channel Equalizer, which is perhaps the most commercially successful of all neural network applications to date, is a single-neuron device used now in virtually all long-distance telephone systems to stabilize voice signals.

The Study found that there has *not* been extensive neural network application development generating available products that accomplish some useful task. Of the 77 application efforts described to the Study, just the four described above have thus far resulted in fielded systems. What's more, all of these fielded systems are operating on personal computers, and none are using any special-purpose neural network hardware. Overall, because of the general lack of support for neural network research, neural network applications remain limited. But the potential of neural networks begins to become evident with a somewhat closer look at various application efforts.

2.4.1 Application: The Kanji Character Recognizer

Figure 2-18 illustrates Nestor Inc.'s Kanji Character Recognizer, which, in this depiction, is translating handwritten Kanji characters for "neural network" into English. This system can accurately recognize approximately 2,500 handwritten Kanji characters better than 92% of the time. This compares favorably

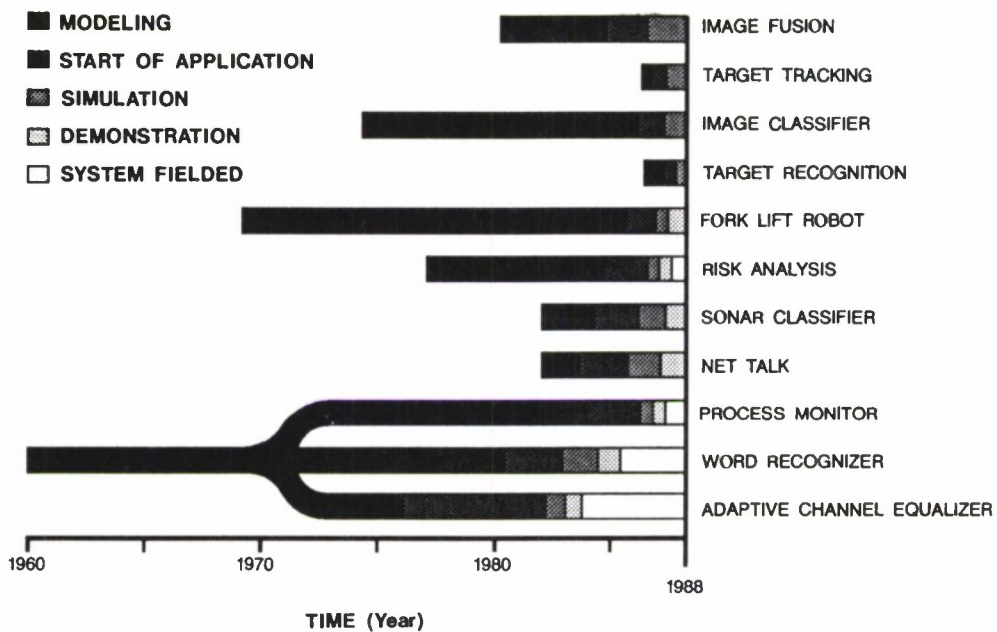


Figure 2-17. Development of Neural Network Applications.

with the average Japanese reader, who is capable of recognizing between 1,800 and 3,000 characters; an intellectual has a grasp of perhaps 10,000 characters.

One useful feature of this system, thanks to the fact that it's a neural network, is that upon training it is capable of recognizing *any* alphabet – e.g., Cyrillic, Hebrew, etc.

The Kanji Character Recognizer is based on Nestor Inc.'s NLS (Nestor Learning System), a set of neural network modules based on the reduced Coulomb energy (RCE) architecture. Each of the NLS three-layer neural network modules is defined to process a different feature subspace of inputs; the RCE architecture allows each network to describe a class-separating mapping that can support any required degree of nonlinearity between pattern class territories. A controller module synthesizes the outputs of the individual neural networks and directs their training.

The Nestor Learning System is a software simulation which operates on an IBM PC/AT personal computer or workstations from Sun Microsystems or Apollo Computer.

2.4.2 Application: Perception

Figure 2-19 shows the results of a neural network now under development that is based on Grossberg's boundary contour system (BCS), a neural network model incorporating neurobiological insights about perception.

Many algorithms for machine vision that are based on symbolic processing techniques are designed to deal with just one type of information – such as boundary, disparity, curvature, shading, spatial frequency – using different mathematical schemes to analyze each of information. Often, then, other types of signals

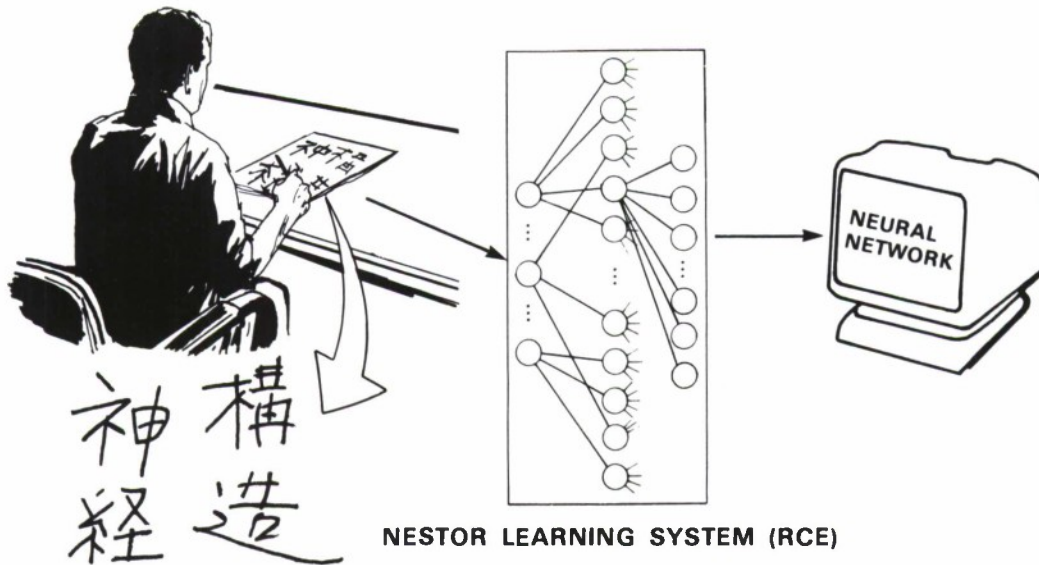


Figure 2-18. Kanji Character Recognizer (Nestor).

101450-24

act as contaminants – noise – rather than as cooperative sources of ambiguity-reducing information. The boundary contour system, by contrast, clarifies scenic data about boundaries, textures, shading, depth, multiple spatial scales, and motion and cooperatively synthesizes this data in real time into a coherently fused representation that is more informative than a representation derived from any one type of scenic data taken in isolation.

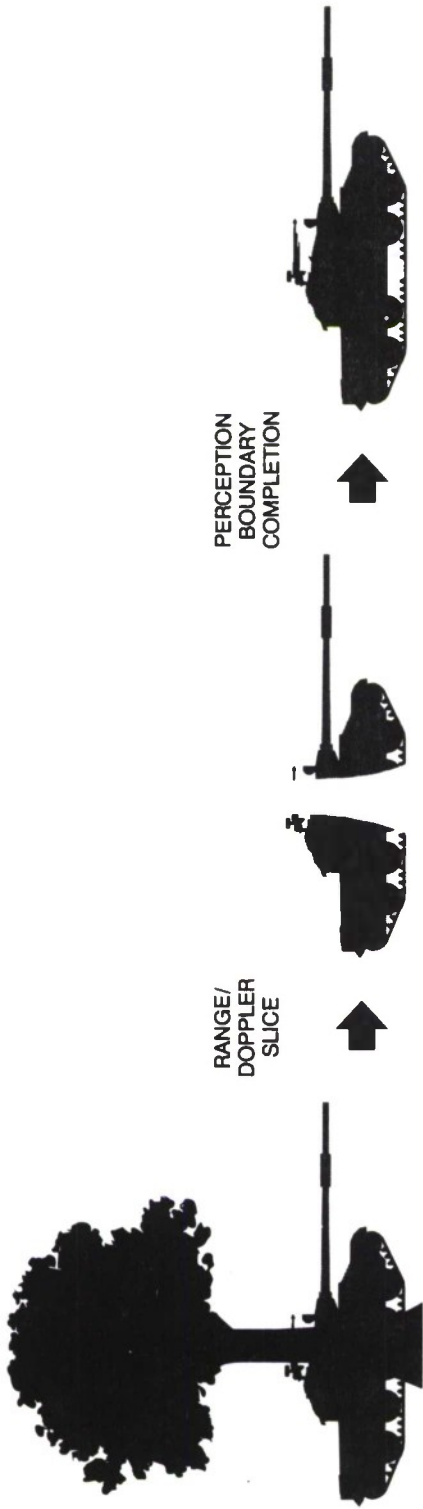
Figure 2-19 illustrates work presently underway using a BCS neural network to process multi-dimensional image data from laser radar sensors. Of course, a human being who sees a tank partially hidden by a tree nevertheless understands that he/she is observing a whole, single tank.

But conventional computer systems do not perform this task well, so a BCS neural network has been used here to automatically complete and fill in the boundaries of a tank which were obliterated in an actual laser radar image by two trees and other noise, as shown in the range silhouette in the lower left of Figure 2-19. Although the neural network operates slowly, the segmented image of the tank is ready for further processing.

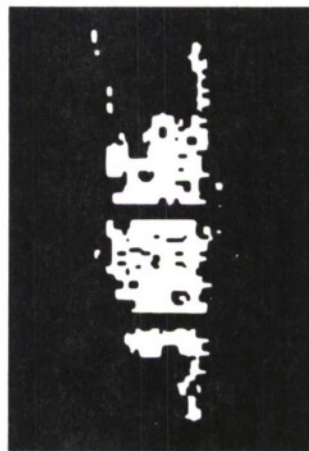
This work is being done on a Digital Equipment Corp. VAX system; the neural network contains 400,000 neurons and 17 million interconnects.

2.4.3 Application: Sonar Target Discrimination

Figure 2-20 describes a neural network developed by Sejnowski that remotely detects undersea mines in shallow waters using active sonar returns. Although it is often difficult for both humans and conventional classification systems to distinguish between mines and clutter on the sea floor, this three-layer neural network based on the backpropagation learning model successfully discriminated between sonar returns of an undersea rock and cylinder.



BOUNDARY CONTOUR SYSTEM (BCS) - LASER RADAR TANK IMAGE



RANGE SILHOUETTE



BOUNDARY COMPLETION

- 4×10^5 NEURONS, 17×10^6 INTERCONNECTS

Figure 2-19. Perception (Van Allen)

The network consisted of 60 input processing elements, or units, in the first layer whose states were “clamped” to the amplitude of the preprocessed sonar signal in the power spectral envelope; a hidden second layer of between 0 and 24 units; and a third output layer of two units – one for cylinder and one for rock. Network training involved approximately 30,000 trials using 104 patterns. Upon testing with another 104 patterns, the network accurately distinguished between rocks and cylinders 90% of the time, which compares favorably with the performances of both human operators and conventional classifiers.

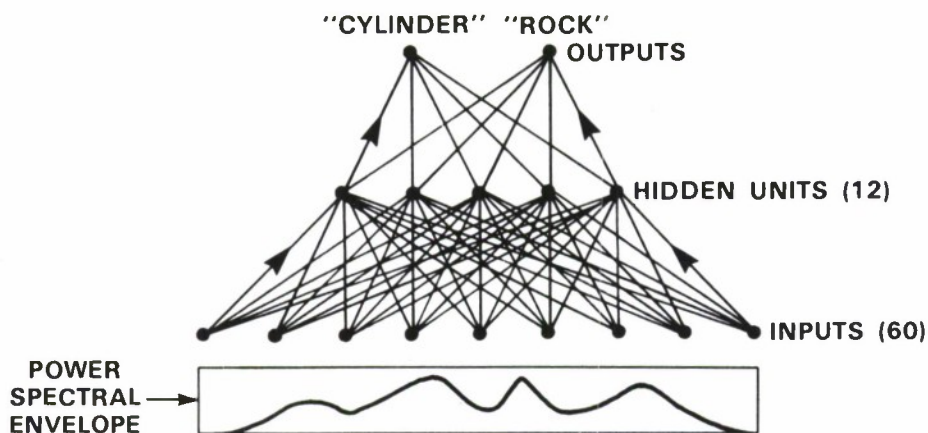


Figure 2-20. Sonar Target Discrimination (Sejnowski).

2.4.4 Application: Multi-layer Perceptron Vowel Classifier

Figure 2-21 indicates how a multi-layer perceptron neural network developed by Lippmann has been used as a vowel classifier.

Conventional speech recognition systems do not perform well when their task is to recognize continuous speech from more than one talker because of several factors, including: (a) the variability and overlap of information in the acoustic signal, (b) the need for high computational rates (a human-like system must match inputs to 50,000 words in real time), and (c) the multiplicity of analyses – phonetic, phonemic, syntactic, and pragmatic – that must be performed.

The input for the network shown in Figure 2-21 consisted of the first and second formants (i.e., the regions of concentration of energy which combine to make up the frequency spectrum of a spoken sound) from roughly 330 tokens of ten different vowels spoken in context by 67 men, women, and children. Half the data was used for training the network; the other half was used to test it.

The network was trained using backpropagation with 50,000 trials. The network’s error rate proved to be the same as the best conventional classifier, and the decision regions it formulated were the same as those which were hand-drawn by a human expert.

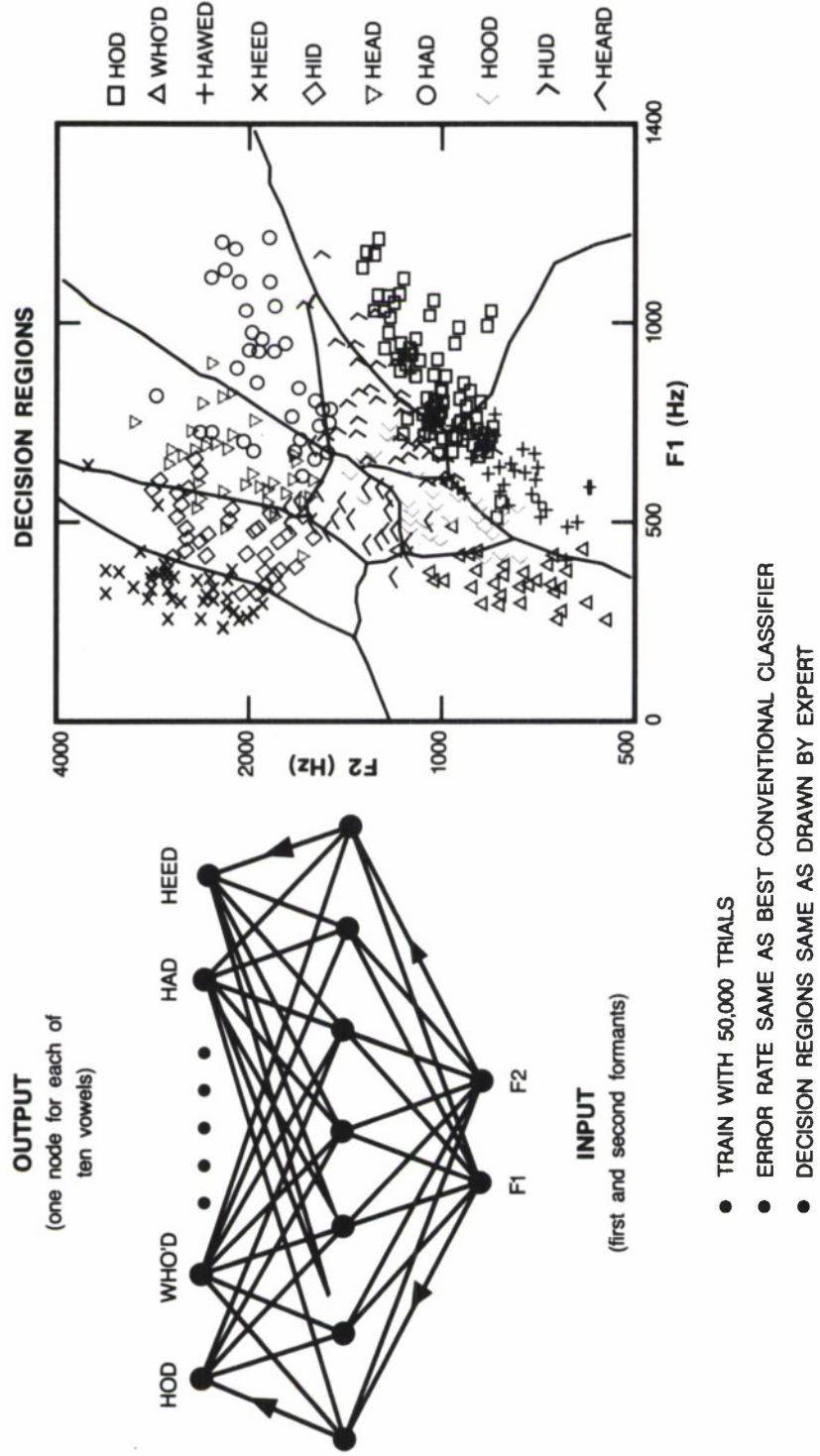


Figure 2-21. Multi-layer Perceptron Vowel Classifier (Lippmann).

2.4.5 Application: Forklift Robot

Figure 2-22 illustrates the use at Martin Marietta of a neural network based on the CMAC model (cerebellar model articulation controller) to operate an industrial robot.

The CMAC is, basically, a trainable function generator. Using a table look-up method to implement complex nonlinear functions implicitly rather than by the mathematical solution of equations, CMAC can realize any smooth and continuous function using simple learning procedures and can operate in real time on conventional digital computers. The system described in Figure 2-22 was implemented on a Motorola 68020.

Guided by the neural network, the robot, which has a forklift end effector equipped with a series of infrared proximity sensors, is supposed to insert the forklift into pallets placed irregularly on a conveyor. This is a difficult problem for conventional robotic controllers because the relationship between a particular pattern of sensor readings and the appropriate direction of forklift movement is very complicated.

The network was trained by a human operator who grasped the forklift and positioned it into the pallet; during this learning procedure, the network's weights are adjusted so that the sensor inputs generate a trajectory command that's similar to the command generated by the human operator. This network has learned to acquire the pallet from any arbitrary starting point with as few as seven teaching points.

2.4.6 Conclusions Concerning Current Neural Network Applications

Although neural network application efforts are diverse, they share one factor, which can be seen in Figure 2-23. Here, the computational requirements of the current applications reviewed by the Study are charted along the same coordinates of interconnects (storage) and interconnects-per-second (speed) which were depicted for biological organisms in Figure 2-13 and for neural network simulators in Figure 2-15. Notably, these applications occupy only the small dark space at the lower left corner of Figure 2-23 – a space well within the bounds of present simulation capabilities.

(A discussion of the computational requirements of various types of neural network applications – including signal processing, robotics, speech, pattern recognition (vision), and some of the applications presented to the Study's Systems Applications Panel – may be found in *Part V, Chapter 4: Application Computational Requirements* of this Report.)

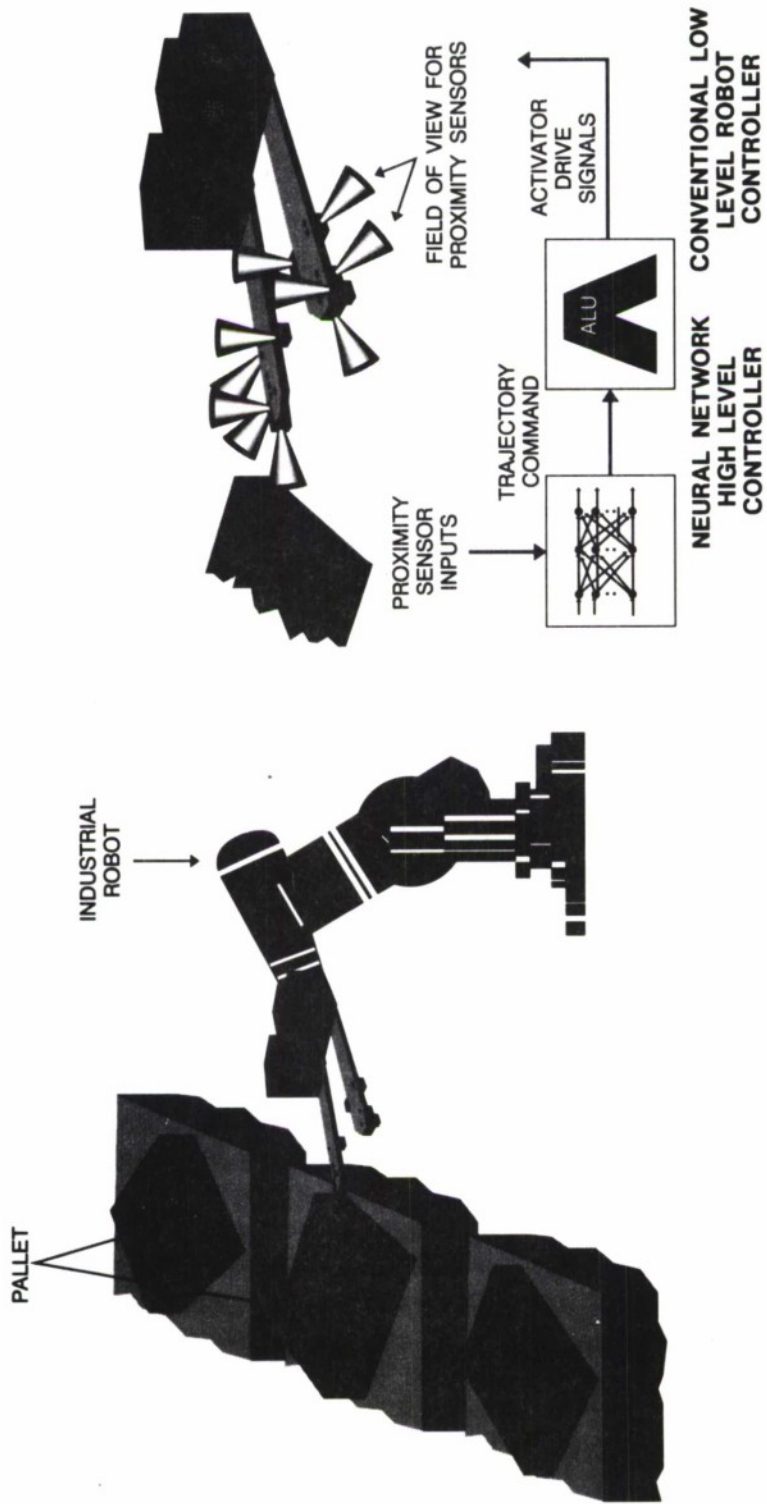


Figure 2-22. Forklift robot (Martin Marietta).

From its examination of current neural network application efforts, the Study offers these specific conclusions:

- Most of today's neural network applications have been accomplished – in the absence of funding for this work – through simulation on low-cost personal computers. Not surprisingly, **current application requirements are consistent with available simulation speed and storage requirements.**
- Despite these limitations, **there have been significant demonstrations of such key neural network capabilities as adaptivity and learning.**
- **Interest in application research and development runs high in the neural network community.**

(Part IV, Chapter 3: *Conclusions Concerning Neural Network System Applications* of this Report contains the conclusions drawn by the Study's System Applications Panel (Panel 4).)

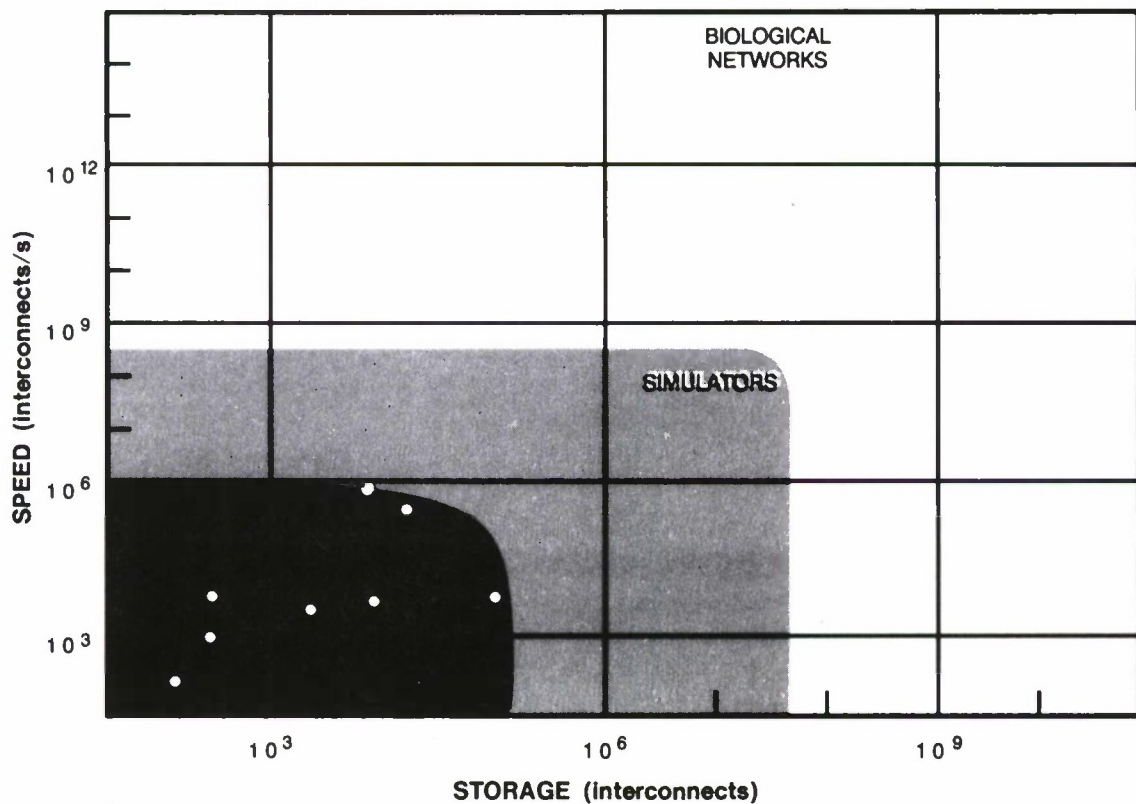


Figure 2-23. Computational Requirements for Current Applications.

2.5 PROJECTED COMPUTATIONAL CAPABILITIES FOR ADVANCED SIMULATORS

With even modest levels of funding and motivation, the gap between actual and potential exploitation of current hardware capabilities will be eliminated rather quickly. But it is certainly evident from Figure 2-13 and Figure 2-15 that neural network researchers face a much greater and more difficult gap: the one between the present hardware state-of-the-art and the capabilities required to implement neural networks that rival the computational storage and speed of, say, a bee.

From the perspective of researchers, this gap is not merely about storage and speed and their inevitable corollary – cost. The gap between what is now available and what researchers need is also filled with other issues – especially the need for greater ease of programming, better information display, and more flexible systems. All of these matters need to be addressed so researchers can move beyond simulations of neural networks to their actual implementation.

2.5.1 The Limits of Neural Network Simulation

Figure 2-24 indicates the present state of hardware affairs in the context of interconnects (storage) and interconnects-per-second (speed): simulators stand at the outer edge of today's capabilities – but, although they exist, for the most part they are not very user-friendly for research, statistical work, or development of large databases. However, as Figure 2-24 also shows, several technologies are poised to push hardware capabilities beyond their present speed and storage limits:

- *Gallium arsenide (GaAs)* and special-purpose *charge-coupled devices (CCDs)* will push up the ceiling on interconnects-per-second.
- Continued developments in *random-access memory (RAM)* technology as well as the lesser-developed *three-dimensional (3-D) chip* technology are expected to expand current storage capacities.
- *Multiprocessing*, meanwhile, will allow the boundaries of simulation to be moved outward up to an order of magnitude.

(*Part V, Chapter 5: Considerations for Future Simulations* as well as *Part VI: Advanced Implementation Technology, Chapter 1: Overview* and *Part VI, Chapter 2: General and Philosophical Issues* of this Report offer expanded discussions of present and future technologies for neural network simulation.)

2.5.2 Technologies and Tools for Implementing Neural Networks

Certainly, if neural networks are to offer solutions to important problems, those solutions must be implemented in a form that exploits the physical advantages offered by neural networks: the high throughput that results from massive parallelism (realtime operation), small size, and low power consumption. In the near term, smaller neural networks may be digitally implemented using conventional integrated circuit techniques. But in the longer term, **implementation of neural networks will have to rely on new technologies.**

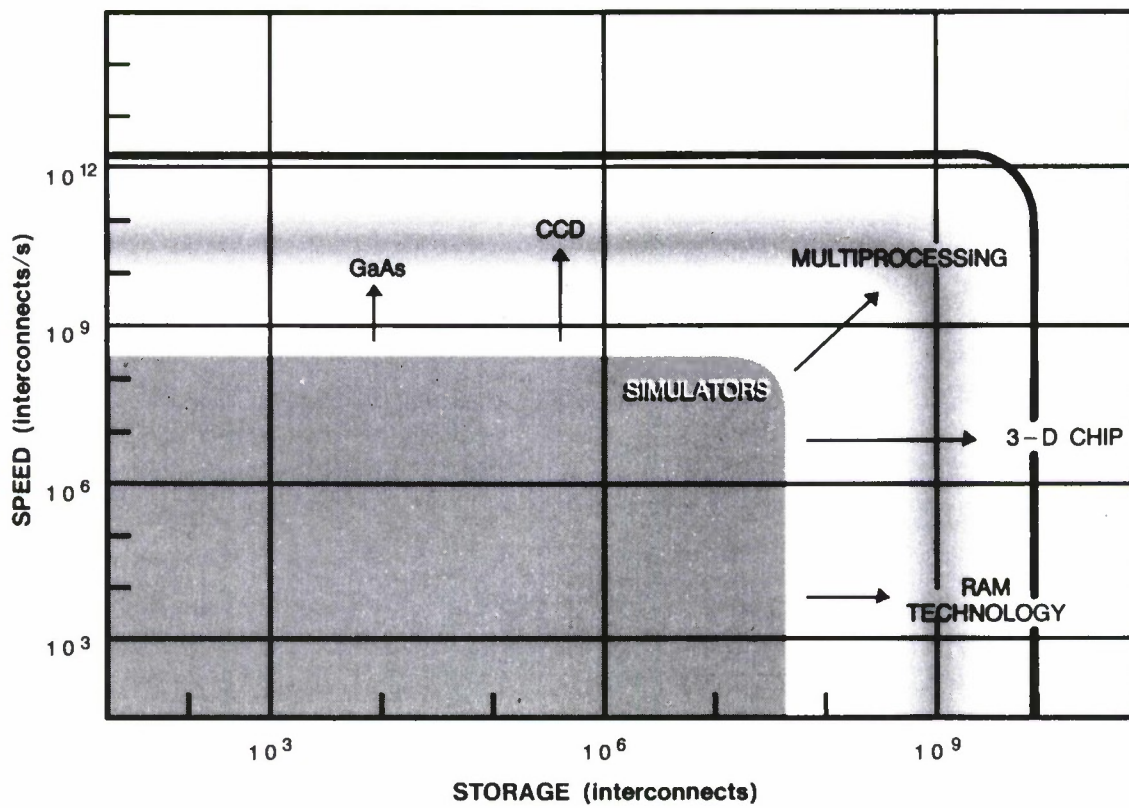


Figure 2-24. Projected Computational Capabilities for Advanced Simulators.

102520-1

The alternatives – as illustrated in Figure 2-25 – for direct implementation of neural networks include:

- *Direct VLSI/VHSIC* (very large scale integration/very high speed integrated circuits), which is a mature technology limited to a low density of interconnects due its two-dimensional nature. All the weights in a neural network implemented in direct VLSI/VHSIC would have to be stored in memory, which would consume a lot of capacity. Figure 2-25 shows a Bell Laboratory chip for vision preprocessing which has been implemented directly in VLSI.
- *Analog VLSI*, which is still a developing technology but promises near-term results and – although it, too, is two-dimensional – offers a high density of interconnects because the weights can be implemented in resistors, thereby obviating the need for additional memory.
- *Optical* technology, which is less developed and longer-term than the silicon-based approaches and limited in the types of neural networks that it can implement, but which offers a very high density of interconnects due to its three-dimensional nature. Here information is stored holographically; another optical system attempts to “read” the information in the first system – comparisons between the stored information and that which the second system attempts to recognize are a form of an associative memory.

(The rationale for these new technologies necessary for neural network implementation, as well as a more detailed description of the technologies themselves, is explicated in *Part VI, Chapter 2: General and Philosophical Issues* of this Report.)

There are, of course, variations on these ideas and combinations of these technologies which are being considered, including layering two-dimensional chips to achieve three-dimensionality and the combination of optical and VLSI devices to create three-dimensional structures.

A variety of experimental chips are now being developed; these are plotted in Figure 2-26 along the storage and speed coordinates and against both current applications and current simulators.

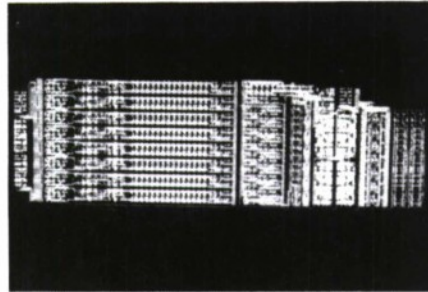
The California Institute of Technology’s (CIT) silicon retina, developed by Mead, is indicated at the lower left of Figure 2-26; this work will be extended to array of $128 \cdot 128$ in the near term, bringing its capabilities to roughly 10^4 interconnects and 10^8 interconnects per second. Lincoln Laboratory (LL) is developing several analog and hybrid analog-digital chips; the Jet Propulsion Laboratory (JPL) and AT&T (ATT) are working on analog and digital chips.

With single chips, speed can be increased by several orders of magnitude, approaching 10^{12} in the next few years. These chips could be stacked up – as many as a thousand, or several thousand, of them might be linked – to bring their collective storage capability to around 10^8 or 10^9 .

In addition, Northrop Corp. (NC), has developed an optical neural network that is projected to perform at better than 10^8 interconnects and 10^9 interconnects per second – through the use of an improved crystal.

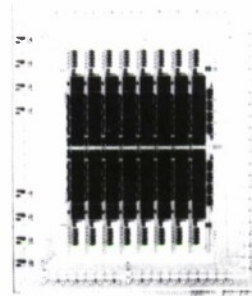
- **DIGITAL VLSI/VHSIC**

- MATURE TECHNOLOGY
- LOW SYNAPSE DENSITY — 2D



- **ANALOG VLSI**

- DEVELOPING — NEAR TERM
- HIGH SYNAPSE DENSITY — 2D



- **OPTICAL**

- DEVELOPING — LONG TERM
- VERY HIGH SYNAPSE DENSITY — 3D

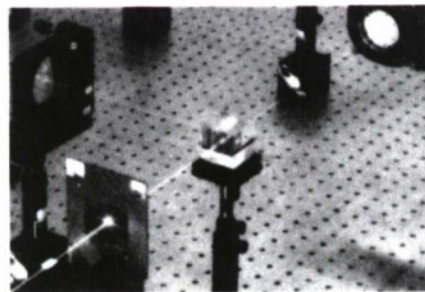


Figure 2-25. Direct Implementation Approaches.

Thus, as Figure 2-26 makes clear, the computational environment defined approximately by 10^{10} interconnects and 10^{12} interconnects per second will be filled over the next five or six years with proposed technology developments.

These new chips are needed not only to build neural network simulators, but they may become very valuable individually in neural network-based military applications.

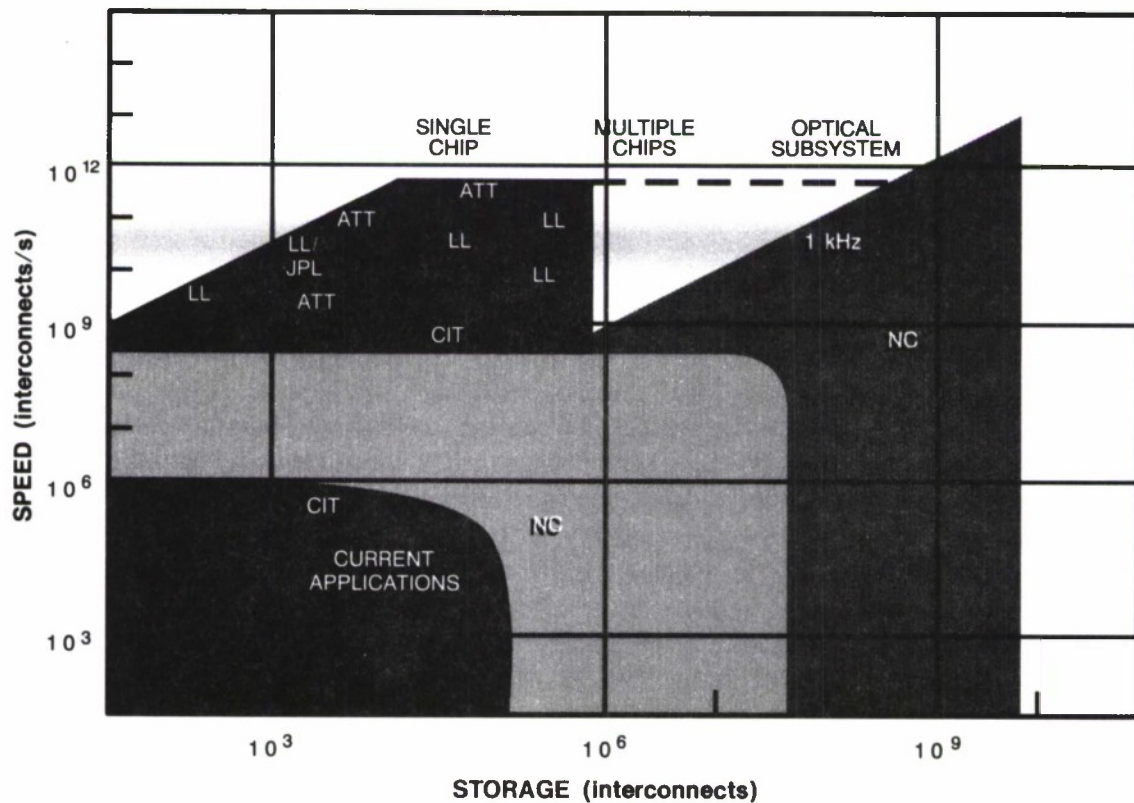


Figure 2-26. Implementation Technologies and Potential Applications.

(Part VI, Chapter 3: Survey of Implementation Efforts of this Report offers both a taxonomy of neural network implementation and a summary of present implementation experiments. Part VI, Appendix A: Description of Implementation Projects, meanwhile, offers brief descriptions of current efforts to implement neural networks in hardware.)

2.5.3 Conclusions Concerning Projected Technology Development

The Study has concluded that technological development in neural network storage and speed requirements will establish the sort of trend line depicted in Figure 2-27.

Above the timeline in its middle, Figure 2-27 deals with speed – that is, interconnects-per-second. Several technological thrusts are underway here:

- The digital signal processing (DSP) chip. DSP experimentation has reached the commercial marketplace – Hecht-Nielsen Neurocomputer Corp., for instance, is using one in its machine. Combining several DSP chips – developers of the MX-1 multiprocessor use 16 – will result, the Study expects, in useful simulators emerging over the next few years.
- Advanced versions of the Cray supercomputer, of course, will be available – but the Study has found that it cannot be expanded for neural network development work.
- Gallium arsenide (GaAs) is expected to be viable for neural network use in the next five years. Here again, a number of gallium arsenide chips can, the Study anticipates, be combined to boost interconnects-per-second into the 10^{10} region. The Study also believes that ideas for a charge-coupled device (CCD) chip will see fruition in about three years, as will an analog-digital hybrid chip.
- Optics, which is the least mature of all the technologies that are viable for neural network research, stands at the far end of the timeline.

If these capabilities are to have any meaning for neural network development, there must be concomitant progress in storage capacity. Fortunately, the Study predicts there will be. The lower half of Figure 2-27 describes the Study's projections of storage technology development:

- The one-megabyte dynamic RAM (DRAM) is widely available, and, hopefully, it will be followed soon by the 16-megabyte version.
- Both wafer-scale and analog storage is expected in three and four years, respectively.
- Optical storage, while under development, remains a little uncertain at the present time.

What Figure 2-27 makes clear is that **technological developments in interconnects and interconnects-per-second could essentially keep up with each other over the next five-plus years**. The Study therefore has concluded that **neural network research and development will not face major hardware bottlenecks through this period if a balanced technology development program is pursued**.

(The conclusions of the Study's Advanced Implementation Technology Panel (Panel 5) can be found in *Part VI, Chapter 4: Conclusions Concerning Advanced Implementation Technology* of this Report.)

2.6 GENERAL CONCLUSIONS OF THE NEURAL NETWORK STUDY

After its examination of the theoretical foundations of neural networks, the simulation and implementation tools currently available as well as those becoming available over the next five or six years, and the status of current neural network applications, the Study has concluded that:

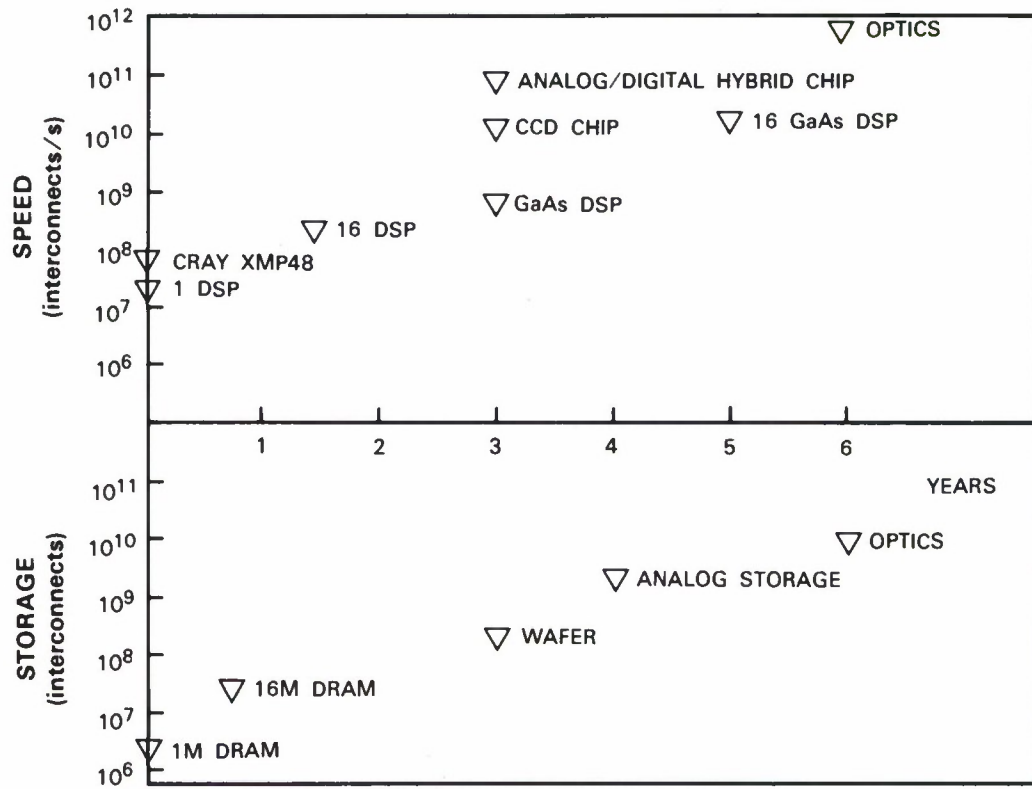


Figure 2-27. Projected Technology Development.

- **Neural networks offer important new computational structures. Their real strength is derived from their ability to self-adapt and learn. *If neural networks realize their full potential, they can be used for machine vision, speech recognition, signal processing, robotics, and other applications – without the need for application-specific software.***
- **Neural network research has matured greatly since the perceptron of the 1950s, *thanks to the development of advanced mathematical theories and new computer tools, and also to a better understanding of neurobiology.* It is time for DARPA to re-examine neural network capabilities.**
- **There have been significant demonstrations of neural network capabilities in vision, speech, signal processing, and robotics, *perhaps not to the scale which might be desired – due to a lack of funding of research – but the variety of problems addressed by neural networks is impressive.***
- **Hardware capabilities are limiting the development of important neural network applications. It is clear that if researchers are not provided with improved simulation and implementation capabilities, the field of neural networks will once again drift off into the wilderness.**

APPENDIX A LIST OF PRESENTATIONS

Yaser Abu-Mostafa, California Institute of Technology
12-09-1987: **Practical Results From NN Theory**

A. Agranat, California Institute of Technology
12-14-1987: **Semiparallel Microelectronic Implementation of NN Models Using CCD Technology**

J. Albus, National Bureau of Standards
10-22-1987: **Robotics**

Joshua Alspecter, Bell Communications Research
10-16-1987: **Electronic Learning Networks**

Charles Anderson, Jet Propulsion Laboratory
12-10-1987: **Strategy Of Biological Systems**
01-20-1988: **Biological Systems From An Engineering Point of View**

David Andes, Naval Weapons Center
11-11-1987: **Thoughts On The Fuze Problem**
11-12-1987: **Analog Neural Network Using Floating Gate Technology**

Bernard Angeniol, Thomson CSF
01-06-1988: **Self-Organizing Maps And The Esprit Program**

Michael Arbib, University of Southern California
12-07-1987: **Neural Network Simulator**

Jacob Barhen, Oak Ridge National Laboratory
10-08-1987: **Neural Networks, Combinatorial Optimization And Asynchronous Computation**
12-07-1987: **Concepts Of Asynchronous Processing**
12-10-1987: **Applications Of Neural Networks In Robotics**

Eric Baum, Jet Propulsion Laboratory
12-10-1987: **Neural Nets And Optimization Problems**

Guy Blelloch, Massachusetts Institute of Technology
11-03-1987: **Implementation Of Back Propagation On The Connection Machine**

Anselm Blumer, Tufts University
02-08-1988: **Theoretical Developments On Machine Learning**

J. Bower, California Institute of Technology
12-07-1987: **Biological Simulations Of NN On Hypercube**
12-10-1987: **Introduction Of NN Technology Into Applications**

Michael Brill, SAIC
12-17-1987: **Retinal Photoreceptor Model**

Heinrich Buelthoff, Mass. Institute of Technology
01-21-1988: **Motion Detection In Fly And Machine Vision**

Michael Cain, Booz-Allen and Hamilton, Inc.
11-17-1987: **Sensor Data Fusion**

Gail Carpenter, Center for Adaptive Systems
10-09-1987: **Invariants Of Neural Networks For Adaptive Pattern Recognition And Robotics**
12-22-1987: **ART-II**

Patrick Castelaz, Hughes Aircraft
11-18-1987: **Neural Network Technology And Applications**

Tony Chan, UCLA
12-09-1987: **Discussion Of Parallel Multi-Grid Methods**

C-C. Chen, Simmons College
12-17-1987: **Modern Data Storage And Retrieval Systems**

Alice Chiang, MIT/Lincoln Laboratory
12-02-1987: **CCD Neural Net Processor And CCD Retina**

Avis Cohen, Cornell University
01-05-1988: **Central Pattern Generator For Locomotion Modeled As A System Of Coupled Limit-Cycle Oscillators**

Dean Collins, Texas Instruments
01-05-1988: **Adaptive Radar Processing**

D. Coon, University of Pittsburgh
11-09-1987: **New Artificial Network Technology**

Leon Cooper, Brown University
10-09-1987: **Neural Networks In Real World Applications: Biological And Computational Constraints**

Robert Cooper, Atlantic Aerospace Corporation
12-14-1987: **Strategic Computer Program Planning**
01-14-1988: **Strategic Computer Program Planning**

Jack Cowan, University of Chicago
02-09-1988: **Neural Net Survey; Eye-Brain Connections**

Oliver Curme, Battery Ventures
01-06-1988: **Venture Capitol Financing Of A Neural Net Company**

John Daugman, Harvard University; Psychology Depart.
01-19-1988: **Biological Early Vision Mechanisms**

Robert Dawes, Martingale Research Corporation
11-11-1987: **On The Simulation Of Neural Networks And Other Dynamical Systems**

Gerald Edelman, Neural Sciences Institute
02-02-1988: **Neural Darwinism**

Mitchell Eggers, MIT/Lincoln Laboratory
01-05-1988: **Biological Learning/Adaption For Pattern Processing**

Richard Elsley, Rockwell Science Center
12-09-1987: **Hopfield Net Application To Planning And Robotics**

Robert Farber, Los Alamos National Laboratory
11-09-1987: **Large Neural Network Simulations On Crays**

Nabil Farhat, University of Pennsylvania

12-14-1987: **An Optical Stochastic Neural Network**

01-22-1988: **Radar Target Recognition From Partial Information Based on Models Of Neural Networks**

Jerome Feldman, Rochester University

10-09-1987: **Massive Parallelism In Nature And Computer Science**

John Fiala, National Bureau of Standards

11-02-1987: **Robotics At NBS And Application Of Cerebella Model Arithematic Computer (CMAC)**

George Fucik, TRW

11-18-1987: **BM/C3**

Lynn Garn, Center for Night Vision and EO

01-20-1988: **Application Of Neural Networks To Target Classification**

Jack Gelfand, SRI International

11-19-1987: **A Multidisciplinary Study Of Integrated Adaptive Systems**

A.P. Georgopoulos, Johns Hopkins University

01-07-1988: **Coding Of Movement Direction By Neuronal Populations**

Nigel Goddard, California Institute of Technology

11-12-1987: **ANN Simulation**

R. Goodman, California Institute of Technology

12-10-1987: **Some Considerations In Applying Neural Nets To Decision Systems And Expert Systems**

Paul Gorman, Allied Signal

01-22-1988: **Learned Classification Of Sonar Target Using A Massively Parallel Network**

Hans Graf, ATandT Bell Laboratories

10-16-1987: **Electronic Neural Network Integrated Circuits**

Guenter Gross, North Texas State University

01-05-1988: **Importance Of Investigating Mammalian Neural Network Parameters For Neural Modeling**

Stephen Grossberg, Boston University

10-09-1987: **Neural Networks And Combinatorial Optimization**

01-19-1988: **A Neural Network For Multiple Scale Filling In Of Fuse 3-D Visual Representations**

Norberto Grzywacz, Massachusetts Institute of Technology

01-05-1988: **Motion Measurement In Invertebrate**

C. Guest, UCSD

12-14-1987: **Optical Neurocomputing at UCSD**

Allon Guez, Drexel University

10-14-1987: **Adaptive Robotic Control**

Todd Gutschow, HNC

12-08-1987: **Simulators And Applications Of NN**

Daniel Hammerstrom, University of Oregon

12-07-1987: **Neural Net Simulation And Evaluation**

Robert Hecht-Nielsen, HNC

11-09-1987: **Perspectives On Neural Network Research**

11-10-1987: **Military Applications Of Neural Networks**

Neville Hogan, Massachusetts Institute of Technology

12-18-1987: **Tool-Use In Biological And Artificial Systems**

Frank Hoppenstaedt, Michigan State University

01-05-1988: **Phase-Locking Of Neuronal Circuits**

David Hubel, Harvard Medical School

10-08-1987: **Neural Mechanisms For Processing Visual Information**

Larry Jackel, Bell Laboratories

11-11-1987: **The Bell Labs Perspective On Funding For Neural Networks**

Von Jennings, Martin Marietta

01-22-1988: **Trainable And Adaptable Neural Nets For Robot Control**

K. Johnson, University of Colorado

11-11-1987: **Optical Implementation Of Neural Networks**

Leonard Johnson, Yale University

12-16-1987: **Simulations Of Differential Equations On A Parallel Machine**

Chuck C. Jorguesen, Thomson CSF

01-06-1988: **Robot Navigation**

Alex Jourjine, Analog Intelligence Corporation

10-30-1987: **Random Code Neural Networks**

Matthew Kabrisky, AFIT/ENG

12-18-1987: **Processing Of Laser Radar Data With Neural Nets**

J. Katz, Jet Propulsion Laboratory

12-14-1987: **Optoelectronic Neural Networks**

Knut Kongelbeck, Hughes Aircraft

10-19-1987: **Exploratory Applications Of Neural Networks**

F. J. Kub, Naval Research Laboratory

12-03-1987: **Programmable Analog Synapses; Learning Implementations**

Robert Kuczewski, TRW

12-08-1987: **Mark III, IV, And V Neurocomputers From TRW**

S-J. Kung, Princeton

11-09-1987: **Systolic Neural Network Processor**

Michael Kuperstein, Wellesley College

10-14-1987: **Neural Networks For Adaptive Hand Eye Coordination**

12-18-1987: **Neural Dynamics Of Adaptive Sensory Motor Control**

Alan Lapedes, Los Alamos National Laboratory
11-09-1987: **Large Neural Network Simulations On Cray**

James Leonard, AFWAL
11-18-1987: **Multifunctional CO2 Laser Radar For Automatic Target Recognition**

Ralph Linsker, IBM
01-07-1988: **Towards An Organizing Principle For A Perceptual Network**

James Mann, MIT/Lincoln Laboratory
12-02-1987: **Self-Organizing Kohonen Network**

Christie Marrian, Naval Research Laboratory
12-03-1987: **Electronic 'Neural' Net Algorithm For Solving Ill Posed Problems**

Orin Marvel, Hughes Aircraft
11-17-1987: **Future Command And Control Systems**

Bimal Mathur, Rockwell
12-09-1987: **Vision Chip Development At Rockwell**

D. Maxwell, Levco
12-08-1987: **Transputer Board For High Speed Simulation**

Harold McBeth, General Dynamics
12-09-1987: **Sensor Target Classification With A Neural Net**

Murali Menon, MIT/Lincoln Laboratory
12-17-1987: **Target Classification Using A Neural Net**

W.T. Miller, University of New Hampshire
11-02-1987: **Learning Algorithms For Robotic Control**

Michael Miller, Washington University
01-19-1988: **Entropy, Markov Random Fields, And Constrained Optimization**

Ennio Mingolla, Boston University

01-07-1988: **A Neural Network Architecture For Pre-Attentive Visual Segmentation**

Marvin Minsky, Mass. Institute of Technology

10-20-1987: **Progress in Neural Networks?**

Alan Moopenn, Jet Propulsion Laboratory

12-07-1987: **Development Of A NN Simulator/Emulator**

A. Murray, University of Edinburgh

11-12-1987: **Bit Serial VLSI Implementations**

Michael Myers, TRW

12-09-1987: **Neural Net Applications At TRW**

Krishnan Natarajan, MIT/Sloan School of Management

11-02-1987: **Review Of Neural Network Study For Merrill Pickard**

William O'Neill, Rochester University

01-07-1988: **Computational Maps For Echo Location In Bat Cortex**

J. Michael Oyster, Hughes

01-06-1988: **Adaption Edge Detection For ATR; A Guideline For Demonstrating the Advantage Of NN For Intelligent Sensor Processors**

John Pearson, SRI International

11-19-1987: **A Multidisciplinary Study Of Integrated Adaptive Systems**

12-18-1987: **Applications Of Computational Maps**

Andre Pellionisz, NYU Medical Center

10-14-1987: **Tensor Geometry Connecting Neural Science And Robotics**

Clif Penn, Texas Instruments

11-11-1987: **Digital Image Processor Approach To Neural Net Simulation**

Leonid Perlovsky, Nichols Research

11-17-1987: **Multiple Sensor Fusion For Discrimination**

Fernando Pineda, Johns Hopkins University
01-22-1988: **Applications And Implementation Of Dynamical Adaptive Systems**

Tomaso Poggio, Mass. Institute of Technology
10-20-1987: **Computer Vision vs. AI And NN**
01-20-1988: **Computer Vision And Neural Networks**

Edward C. Posner, Jet Propulsion Laboratory
12-10-1987: **Application Areas For NNs**

Jack Raffel, MIT/Lincoln Laboratory
10-09-1987: **Electronic Circuits For Neuromorphic Systems**
12-02-1987: **Wafer Scale Neuromorphic Systems Design**

Todd Reed, University of Minnesota
11-04-1987: **Segmentation Of Textured Images And Gestald Organization Using Spatial/Spatial Frequency Represent**

Ronald Rivest, Massachusetts Institute of Technology
02-05-1988: **Theoretical Developments On Machine Learning**

Steven Rogers, AFIT
12-18-1987: **Processing Of Laser Radar Data With Neural Nets**

David Rumelhart, Stanford University
10-09-1987: **Neurally Inspired Networks**

Howard Rumsey, Consultant
12-10-1987: **Associative Memories**

Thomas Ryan, SAIC
11-10-1987: **GINNI: A Neural Network Simulator**
12-17-1987: **Artificial Adaptive Neural Network Systems**

Doyce Satterfield, USASDC / Strategic Defense Command
11-17-1987: **BM/C3 The Paramount Strategic Defense Problem**

Walter Schneider, University of Pittsburgh
01-20-1988: **Models For Focus Of Attention In Human Vision**

Eric Schwartz, NYU Brain Research
10-08-1987: **The Interface Between Neuroscience And Computer Science**
01-07-1988: **Computational Mapping**

Terrence Sejnowski, Johns Hopkins University
10-09-1987: **Analyzing The Hidden Units In Multilayered Neural Networks**
01-05-1988: **Computing Shapes From Shading With Neural Networks**

Gordon Shaw, Univ. of California at Irvine
01-21-1988: **Spatial-Temporal Coding In Dynamic Models Of Brain Function And Potential Applications**

David Skapura, Ford Aerospace and Communications Corp.
12-11-1987: **Practical Expert Systems, Neural Networks, And Frame Based AI**

Charles Sodini, Massachusetts Institute of Technology
12-03-1987: **Silicon Implementation Issues For Vision Processing; The MIT Database Accelerator: A Novel Content Addressable Memory**

Bernard Soffer, Hughes Research Laboratory
10-08-1987: **Optical Implementation Of Neural Networks**

David Spencer, MIT/Lincoln Laboratory
10-30-1987: **AI In Air Traffic Control**

Hal Stoll, Northrop Research and Technology Center
11-19-1987: **DARPA ATR Optical Processors**

William Stoner, SAIC
12-17-1987: **Neocognitron Evaluation**

Ted Sullivan, Princeton
01-07-1988: **Computational Maps For Auditory Space Localization In The Owl's Brain Stem**

Richard Sutton, GTE Laboratory

01-06-1988: **Connectionist Learning For Computer Integrated Manufacturing**

Harold Szu, Naval Research Laboratory

10-15-1987: **The Sixth Generation Computers And Super Technology**

A. Tanguay,

12-14-1987: **Fundamental Limits of Photorefractive Materials and Devices**

David Tank, Bell Laboratories

10-08-1987: **Neural Networks For Optimization Problems**

M. Fernando Tenorio, Purdue University

11-12-1987: **Simulations And Emulations**

Anil Thakoor, Jet Propulsion Laboratory

11-12-1987: **New Technology For Analog Synapses Based On Ion Transport**

Tomasio Toffoli, Massachusetts Institute of Technology

11-04-1987: **Cellular Automata As Neural Networks: Pattern Recognition And Tracking**

David Touretzky, Carnegie Mellon University

11-11-1987: **Back Propagation On The Warp Supercomputer**

K. Wagner, University of Arizona

11-11-1987: **Optical Implementation Of Neural Networks**

Deborah Walters, SUNY/Buffalo

01-19-1988: **Features, Representations And Computation For Neural Networks**

David Waltz, Thinking Machines Corporation

10-20-1987: **Connectionist And 'Neural Net' Models**

Allen Waxman, Boston University

10-20-1987: **Mobile Robots vs. Neural Navigators**

01-19-1988: **Motion Computation In Vision**

Harry Wechsler, University of Minnesota

11-04-1987: **Distributed Processing For Invariant Recognition And Data Fusion**

Fred Weingard, Booz-Allen and Hamilton, Inc.

11-18-1987: **Laser Radar Signal Interpretation**

James White, Honeywell Research Center

11-17-1987: **Application Of Neural Networks**

Harper Whitehouse, Naval Ocean Systems Center

10-27-1987: **Advances In Technology For Signal Processing**

Bernard Widrow, Stanford University

10-08-1987: **Theory And Applications Of Layered Neural Networks - Past And Future**

George Works, SAIC

12-08-1987: **Development Of Anspec Software And Delta-1 Hardware For NN Simulation**

G. Rakuljic Yarviv, Stanford University

12-14-1987: **Photorefractive Materials and Applications in Phase Conjugate Optics**

Ben Yuhaz, Johns Hopkins University

01-22-1988: **Mapping Visions To Phonemes For Visual Speech Recognition**

APPENDIX B

NEURAL NETWORK GLOSSARY OF TERMS

ACAMs (Associative Content Addressable Memories) Neural networks whose associative memory is content-addressable. I.e., when such a network is stimulated with some fragment of an associated memory or pattern, the network will respond with the entire memory or pattern. Thus the network can be addressed using a partial pattern or memory rather than just its location. See also "Associative Memory."

Adaline (Adaptive Linear Neuron) A member of a family of trainable pattern-classifiers which distinguishes between patterns on the basis of linear discriminate functions. See also "Perceptron," "Multi-layer Perceptron," and "Single-layer Perceptron."

Adaptive Resonance Theory (ART) A type of neural network model trained without supervision which is used in pattern classification problems. ART actually describes two classes of neural networks (ART I and ART II) that form categories for input data with the coarseness of the categories determined by the value of a user-selectable parameter (called the vigilance parameter).

Associative Memory A memory which allows retrieval of information by presentation of inexact or incomplete stored memory keys. Such a memory is tolerant of partial or partially erroneous information. The Hamming Network is an example of an associative memory. Also called 'content-addressable memory' and 'associative learning.' See also "ACAMs (Associative Content Addressable Memories)."

Axon In biological systems, a nerve process attached to the soma which, unlike the dendrite, is electrically active and can serve as the final output channel of the neuron. An axon acts as a nonlinear threshold device that produces a rapid voltage increase or decrease, called an 'action potential.' An action potential is the pulse emitted by a neuron. See also "Dendrite," "Neuron," "Neurotransmitters," "Soma," and "Synapse."

Backpropagation A learning algorithm for updating weights in a multi-layer, feedforward, mapping neural network that minimizes mean squared mapping error.

Bidirectional Associative Memory A type of neural network model which is a hetero-associative version of the Hopfield Network.

Boltzmann Machine A type of supervised neural network learning algorithm in which network states are determined by "simulated annealing." Boltzmann machines use a noise process to find the global minimum of a cost function.

Boundary Contour System (BCS) A type of neural network algorithm used in image segmentation problems.

Cellular Automata A mathematical formalization for parallel processes. Specifically, a cellular automaton is a graph whose nodes are finite-state machines (thus the underlying graph, or 'space,' of a given cellular automaton is considered to be fixed; it cannot be altered by any of its nodes). The operation of a cellular automaton is determined by information passed between those nodes that are connected (in most cases, the interconnections between nodes pass information bidirectionally).

Cerebellum In biological systems, the part of the brain which researchers believe controls voluntary movements. The cerebellum is present in all vertebrates and, in comparison with the cerebrum, has a notably regular and simple architecture.

Cerebral Neocortex In biological systems, the major part of the cerebrum, containing most of the brain's neurons.

Cerebrum In biological systems, the brain.

CMAC (Cerebellar Model Articulated Controller) A type of neural network model which adaptively forms complex nonlinear maps and is typically used in motor control problems and defined by a redundant but direct, one-to-one, feedforward connection topology.

Cochlea Chip An analog VLSI circuit modeled after the biological cochlea (a part of the mammalian ear).

Competitive Learning An unsupervised learning algorithm in which groups of processing elements in a neural network compete among themselves to respond to a set of stimulus input patterns. The winner within each group is the one whose connections make it respond most strongly to the pattern; the winner then adjusts its connections slightly toward the pattern that it won.

Computational Maps Two-dimensional arrays (often stacked along a third dimension) of locally interconnected processing elements that represent variables or objects by the position and pattern of activity on their surfaces. Computational maps exhibit properties of topological self-organization, self-optimization, and fault tolerance.

Connectionism This term is based on an assumption shared by most massively-parallel computational formalisms: that only a small number of bits of information can be sent from one processor to another. Hence, an important conventional computer mechanism – i.e., passing complex symbolic structures – cannot be used directly. So the burden of computation is put on the connection structure of the network. 'Connectionist' systems have become largely synonymous with neural networks.

Connectivity Neural networks exhibit several kinds of patterns of connectivity between their processing elements, depending on the neural network model being used.

Processing elements, or nodes, may be fully connected, locally connected to neighboring nodes, or sparsely connected to a few distant nodes. In addition, networks

may be layered and the processing elements, or nodes, in these layers linked by means of feedback or feedforward connections.

See also “Feedback,” “Feedforward,” “Full Connectivity,” “Local Connectivity,” “Nearest Neighbor Connectivity,” “Sparse Connectivity,” and “Neural Network.”

Convergence Procedure In some neural networks, after a finite number of presentations to the network of given stimulus- response patterns, the values of the network’s weights approach the set of values representing whatever computation or classification that is embodied in the stimulus-response patterns.

Crosstalk The overlap of input patterns in a neural network, which can result when a network does not have enough processing elements to allow one element or a group of elements to be reserved exclusively for every possible input pattern.

Darwin III Automaton A type of neural network model using self-supervised training; this model is a complex simulated automaton that learns to follow a moving target and touch the target with a multi-jointed arm. It is an instantiation of a developing theory of brain function called ‘neural darwinism.’

Dendrites In biological systems, the long, irregularly- shaped nerve processes attached to the soma which either (a) receive inputs from other neurons via specialized contacts called synapses or (b) connect other dendrites to synaptic outputs. Dendrites are electrically passive; the geometry of their tree- like shapes can have major effect on the time course and final potential of any synaptic activation. See also “Axon,” “Neuron,” “Neurotransmitters,” “Soma,” “Synapse.”

Distributed Representation Each entity or concept is represented by a pattern of activity distributed over many processing elements, and each processing element is involved in representing many different entities or concepts. As opposed to local or unary representation. See also “Grandmother Cells” and “Local Representation.”

Dryware Idiom for man-made systems which perform information processing, control, or emulation of human intelligence; used in contrast to ‘wetware.’ See also “Wetware.”

Excitation See “Neurotransmitters” and “Weight.”

Fan-in The number of processing elements that either excite or inhibit a given unit.

Fan-out The number of processing elements directly excited or inhibited by a given unit.

Feedback Characterized by multi-layer neural networks with recursive connections that iterate over many cycles to produce an output. An example of a feedback neural network is the Hopfield Network. Contrasted with ‘feedforward.’ See also “Connectivity” and “Feedforward.”

Feedforward Characterized by multi-layer neural networks whose connections exclusively feed inputs from lower to higher layers; in contrast to a feedback network, a feed-forward network operates only until its inputs propagate to its output layer. An example of a feedforward neural network is the multi-layer perceptron. See also “Connectivity” and “Feedback.”

Fixed Weight See “Weight.”

Formal Neurons Simplified artificial representations of biological networks of neurons based on a proof that all those processes which can be described with a finite number of symbolic expressions – e.g., simple arithmetic, recursive application of logical rules, etc. – can be embodied as simple logical switches (published by W.S. McCulloch and W.H. Pitts in 1943). Also called McCulloch-Pitts networks.

Full Connectivity All processing elements, or nodes, in a neural network are connected to all other processing elements, or nodes; also ‘fully connected.’ In contrast to local and sparse connectivity. See also “Connectivity,” “Local Connectivity,” “Nearest Neighbor Connectivity,” and “Sparse Connectivity.”

Graceful Degradation In neural networks, the notion that no single processing element or neuron is essential to the network’s operation; network performance gradually deteriorates as more and more processing elements are destroyed, but there is no single critical point at which performance breaks down.

Grandmother Cells In neural networks, processing elements which store memory on a one-to-one basis. See also “Distributed Representation,” “Local Representation.”

Hamming Network A neural network algorithm, based on the Hopfield Network, which is used in pattern classification problems. The feedforward Hamming Network is notable for requiring fewer connections than the Hopfield Network. Also called the ‘unary Model.’

Hebbian Learning A learning algorithm in which the repeated excitation of the interconnection between two processing elements causes the strength, or weight, of that interconnection to increase.

Hidden Units Those processing elements in multi-layer neural network architectures which are neither the input layer nor the output layer but are located in between these and allow the network to undertake more complex problem-solving (i.e., nonlinear mapping) than networks with no hidden units. Also called ‘hidden layers of processing elements.’

Hippocampus In biological systems, the part of the brain, located in the temporal lobe of the cerebrum, thought to be the site of formation of short-term or working memories, especially those related to spatial aspects of a biological organism’s environment. Like the cerebellum, the hippocampus has a strikingly regular structure.

- Hopfield Network** A type of neural network model characterized by full connectivity, feedback, and unsupervised training which is used in pattern classification and optimization problems.
- Indium Bump Bonding** A technology under development for connecting non-silicon infrared detector arrays to silicon integrated circuit preamplifiers and signal processors. Bump bonding is being proposed as a means of achieving massively parallel interconnections of neural networks.
- Inhibition** See “Neurotransmitters” and “Weight.”
- Input** See “Processing Element” and “Weight.”
- Interconnect** The links or information channels between a neural network’s processing elements. The pattern of these interconnections must be appropriate to the neural network’s application. See also “Connectivity,” “Neural Network,” “Processing Element,” and “Weight.”
- Kohonen Self-organizing Feature Map** A type of neural network learning algorithm which does not require explicit tutoring of input-output correlations and spontaneously self-organizes upon presentation of input information patterns. It is used in optimization and pattern classification problems.
- Kurogi Spatiotemporal Pattern Recognition Model** A neurobiological neural network model trained with supervision which attempts to model and predict internal neuron dynamics.
- Layers** The processing elements of a neural network are arranged into layers (also called ‘slabs’). Although the weights of each of the processing elements in a layer may vary, all processing elements in a layer have the same transfer function. See also “Connectivity.”
- Learning Algorithms** In neural networks, the equations which modify some of the weights of processing elements in response to input signals and values supplied by the transfer function; the learning algorithm(s) employed in a neural network allow the processing elements’ responses to input signals to change over time.
- LMS (Least Mean Square) Algorithm** A modification to the perceptron convergence procedure which can form the least mean squared solution in certain problems. This solution, used the Adaline, minimizes the mean squared error between the desired output of a perceptron-like network and the actual output. See also “Adaline.”
- Local Connectivity** The processing elements, or nodes, in one layer of a multi-layer neural network are connected only to the corresponding nodes in other layers. In contrast to full and sparse connectivity. See also “Connectivity,” “Full Connectivity,” “Nearest Neighbor Connectivity,” and “Sparse Connectivity.”

Local Representation In neural networks, the use of one processing element to represent each entity or concept. Also called unary representation; as opposed to distributed representation. See also “Distributed Representation” and “Grandmother Cells.”

Markov Random Field Network A type of neural network algorithm used in optimization problems and closely related to cellular automata.

Marr Neocortex Model A neural network model trained with supervision which attempts to determine the function of biological cell layers in the neocortex.

Multi-layer Perceptron A multi-layer feedforward neural network that is fully connected and which is typically trained by the backpropagation learning algorithm.

Nearest Neighbor Connectivity A type of local connectivity in which a neural network’s processing elements, or nodes, are connected to those processing elements, or nodes, which are physically contiguous. See also “Connectivity,” “Full Connectivity,” “Local Connectivity,” and “Sparse Connectivity.”

Neocognitron A type of neural network model used in pattern classification problems. The neocognitron model combines an unsupervised learning algorithm with a multi-layer architecture designed to provide pattern recognition with tolerance to positional shifts, geometric distortion, and scale variation.

Neural Network An information processing system which operates on inputs to extract information and produces outputs corresponding to the extracted information. Also called ‘artificial neural networks,’ ‘connectionist models,’ ‘parallel distributed processing models,’ ‘neuromorphic systems.’

Specifically, a neural network is a system composed of many simple processors – fully, locally, or sparsely connected – whose function is determined by the connection topology and strengths. This system is capable of a high-level function, such as adaptation or learning with or without supervision, as well as lower-level functions, such as vision and speech pre-processing. The function of the simple processor and the structure of the connections are inspired by biological nervous systems.

The key attributes of neural networks are (a) massive parallelism, which results in high-speed decisions and potential fault tolerance, and (b) adaptivity, which means neural networks can be trained rather than programmed in the classical way, and their performance may improve with experience.

A neural network is described by either an algorithm (which specifies the functional transformation from inputs to outputs) and/or an implementation (the physical realization of the processing mechanism that runs the algorithm).

See also “Connectivity,” “Processing Element,” and “Weight.”

Neurodynamics The study of the generation and propagation of synchronized neural activity in biological systems.

Neuron The nerve cells in biological systems. These can be categorized into two broad types – local processing ‘interneuron’ cells and output cells. Each neuron has an inside and an outside separated by a plasma membrane. The inside of the cell and the fluid surrounding the cell have different concentrations of charged ions which create a potential difference across the membrane; these differential ion concentrations provide the electrical energy for all nerve cells (just like a battery). See also “Axon,” “Dendrite,” “Neurotransmitters,” “Processing Element,” “Soma,” and “Synapse.”

Neurotransmitters In biological systems, specialized molecules that act across synapses and which open up neural membrane channels that permit ionic currents (i.e., action potentials) to act. Neurotransmitters and currents either depolarize the membrane, resulting in neural excitation, or hyperpolarize it, resulting in neural inhibition. Some 50 different neurotransmitters have been identified so far; some appear to play an important role in determining patterns of neural interconnections. See also “Axon,” “Dendrite,” “Neuron,” “Soma,” and “Synapse.”

Output See “Processing Element” and “Weight.”

Pattern Classifiers Mappings that define partitionings of feature space into regions corresponding to class membership.

Perceptron A member of a family of trainable pattern-classifiers which distinguishes between patterns on the basis of linear discriminate functions. See also “Adaline,” “LMS Algorithm,” “Multi-layer Perceptron,” and “Single-layer Perceptron.”

Processing Element The simple processors (also called ‘neurons’ after their biological inspiration, or, simply, ‘units’) that are the essential units of which a neural network is comprised.

Every processing element, which is endowed with only a small amount of local memory, receives one or more inputs from other processing elements or from external sources; these inputs are then modified by some weighted value specific to each input according to a learning algorithm. The sum of the products of the different weights times their individual inputs is then computed by the processing element. The processing element generates a single output signal that depends on these input sums. This single output signal can be fanned out to some number of other processing elements or be used as output from the network.

See also “Connectivity,” “Neural Network,” and “Weight.”

Receptive Fields In some multi-layer neural networks, a processing element in the hidden layer(s) may receive input from a group of neighboring units, called the receptive field.

Reduced Coulomb Energy (RCE) Network A type of neural network model used in general classification problems and characterized by a sparse, feedforward connection topology and supervised training.

Relaxation In neural networks, the notion that computation proceeds by iteratively seeking to satisfy a large number of weak restraints; thus connections represent constraints on the co-occurrence of pairs of processing elements. The network settles into a solution rather than calculating one.

Retina Chip A type of neural network model used as a VLSI vision front-end and characterized by local connectivity.

Self-organization The autonomous modification of the dynamics of a complete neural network via learning in some or all of its processing elements to achieve a specified result. See also “Self-supervised Training,” “Supervised Training,” and “Unsupervised Training.”

Self-supervised Training A means of training adaptive neural networks; self-supervision is used by automata which require internal error feedback to perform some specific task. For example, automata which learn to track a moving spot by controlling simulated eye muscles can generate an error signal based on the distance between the position of the spot on a simulated retina and the center or fovea of the retina. See also “Self-organization,” “Supervised Training,” and “Unsupervised Training.”

Simulated Annealing A stochastic computational technique derived from statistical mechanics for finding near globally-minimum-cost solutions to large optimization problems.

Single-layer Perceptron A type of neural network algorithm used in pattern classification problems and trained with supervision. The single-layer perceptron generated much interest when it was initially developed in the 1950s by Rosenblatt because of its ability to learn to recognize simple patterns. Connection weights and the thresholds in a perceptron can be fixed or adapted using a number of different algorithms. See also “Adaline” and “Perceptron.”

Site Function In a neural network, a processing element’s inputs are connected to specific sites. A processing element may have more than one “input site.” Each site has an associated site function which carries out local computation based on the input values at the site.

Soma In biological systems, the large, round central body of a neuron which contains the genetic and metabolic machinery necessary to keep the neuron alive. See also “Axon,” “Dendrite,” “Neuron,” “Neurotransmitters,” and “Synapse.”

Sparse Connectivity The processing elements, or nodes, in a neural network are connected to only a few distant other processing elements, or nodes. In contrast to full and local connectivity. See also “Connectivity,” “Full Connectivity,” “Local Connectivity,” and “Nearest Neighbor Connectivity.”

Stochastic A process involving a randomly determined sequence of observations, each of which is considered as a sample of one element from a probability distribution. Stochastic variation implies randomness as opposed to a fixed rule or relation in passing from one observation to the next in order.

Supervised Training A means of training adaptive neural networks which requires labeled training data and an external teacher. The teacher knows the desired correct response and provides an error signal when an error is made by the network. This is sometimes called 'reinforcement learning' or 'learning with a critic' when the teacher only indicates whether a response was correct or incorrect but does not provide detailed error information. Also called 'hetero-associative learning.' As opposed to unsupervised training, self-organization, and auto-associative learning. See also "Self-organization," "Self-supervised Training," and "Unsupervised Training."

Synapse In biological systems, the tissues connecting neurons. Synapses are the specialized contacts on a neuron which are the termination point for axons from other neurons. Synapses make contact with the dendrites from other neurons and are capable of changing a dendrite's local potential in a positive or negative direction. See also "Axon," "Dendrite," "Neuron," "Neurotransmitters," and "Soma."

Transfer Function The differential (or difference) equations which determine each processing element's operation. These equations describe how the output signal evolves in time as a function of the input signals.

Unsupervised Training A means of training adaptive neural networks which requires unlabeled training data and no external teacher. Data is presented to the network and internal categories or clusters are formed which compress the amount of input data that must be processed at higher levels without losing important information. This clustering task is sometimes called 'vector quantization.' See also "Self-organization," "Self-supervised Training," and "Supervised Training."

Value Unit Coding In neural network data representation, the encoding of the value of a variable as the location of an active processing element, or node. Such nodes are often arranged in an orderly fashion to form a topographic map of some external variable.

Variable Unit Coding In neural network data representation, the encoding of the value of a variable as the amplitude of the output of a processing element, or node.

Vector Quantization See "Unsupervised Training."

Viterbi Network A neural network architecture which implements a temporal decoding algorithm used for nonlinear-analog-processing-based speech recognition.

Weight A processing element (or neuron or unit) need not treat all inputs uniformly. Processing elements receive inputs by means of interconnects (also called 'connections' or 'links'); each of these connections has an associated weight which signifies its

strength. The weights are combined into a simple value in order to update the processing element's potential.

The processing element communicates with the rest of the network by transmitting this simple value to all the other processing elements to which it is connected. This output value is often described in terms of a level of excitation (which is quite different from the binary on-off typical of digital systems). A processing element's potential reflects the amount of excitation it has been receiving from other processing elements.

There is a separate processing element dedicated to each possible value of each of the parameters defined in the system (a neural network with pre-defined parameters is considered fixed). Since a consistent state in the network requires that there should be only one active value for each parameter, the network has mutually inhibitory connections among the competing values for each parameter. Thus the network itself embodies mutual constraints and its behavior is characterized by states where a coalition of mutually reinforcing processing elements becomes stable and suppresses its rivals. In a neural network there are, therefore, many active processing elements forming one or more coalitions. Any given processing element will participate in several coalitions and need not have a simple response pattern.

See also "Connectivity," "Processing Element," and "Neural Network."

Wetware Idiom for biological systems which exhibit a capacity to perform information processing functions, control functions, or intelligence; used in contrast to dryware. See also "Dryware."

Part II

**ADAPTIVE KNOWLEDGE
PROCESSING**

Edited by Richard P. Lippmann

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS	vii
LIST OF TABLES	ix
II ADAPTIVE KNOWLEDGE PROCESSING	
1. OVERVIEW	1
1.1 Introduction	1
1.2 Neurobiology	1
1.3 Mathematical Theory	1
1.4 Conclusions	3
2. BACKGROUND	5
2.1 Introduction	5
2.2 The Panel's Charter	5
2.3 Defining 'Neural Network'	5
2.4 Training Procedures	7
2.5 Topology	7
2.6 Data Representation	8
2.7 Recent Interest and Potential	8
3. TASKS NEURAL NETWORKS PERFORM AND REPRESENTATIVE MODELS	11
3.1 Introduction	11
3.2 Tasks That Neural Networks Perform	11
3.3 Illustrative Neural Network Models	13
3.4 Important Training and Performance Issues	18

4. SINGLE- AND MULTI-LAYER PERCEPTRONS	21
4.1 Introduction	21
4.2 Single-layer Perceptrons	21
4.3 Multi-layer Perceptrons and Backpropagation	24
5. ASSOCIATIVE MEMORIES	27
5.1 Introduction	27
5.2 The Unary or Hamming Network	27
5.3 The Hopfield Model	29
5.4 Sparsely-distributed Feedforward Models	30
6. CLASSIFICATION AND CLUSTERING MODELS	33
6.1 Introduction	33
6.2 A Taxonomy of Classifiers	33
6.3 Supervised Classifiers	35
6.4 Unsupervised Classifiers	38
7. RECURRENT NETWORKS	43
7.1 Introduction	43
7.2 Computational Energy	43
7.3 The Boltzmann Machine	44
7.4 High-order Networks	44
7.5 Cellular Automata	44
7.6 Selecting or Enhancing the Maximum Value	45
7.7 Stability	45

8. VISION	47
8.1 Introduction	47
8.2 Early and High-Level Vision	48
8.3 Physiology and Machine Vision	48
8.4 Survey of Representative Work	49
9. SPEECH RECOGNITION	55
9.1 Introduction	55
9.2 Physiological Preprocessors	57
9.3 Calculating Distance Scores to Exemplar Patterns	58
9.4 Classifying Words and Phonemes without Time Alignment	58
9.5 Neural Networks for Time Alignment	61
10. ROBOTICS	67
10.1 Introduction	67
10.2 Trajectory Control	67
10.3 Arm/Camera Control	69
11. NEURAL NETWORKS AND OTHER DISCIPLINES	73
11.1 introduction	73
11.2 Higher-Level AI Problems	73
11.3 Cognitive Science	74
11.4 Classical Conditioning	74
12. TOWARD A THEORY OF NEURAL NETWORKS	77
12.1 Introduction	77
12.2 Capability	78

12.3	Memory Capacity	79
12.4	Distribution-free Learning	79
12.5	Learning and Using Internal World Models	80
12.6	NP Complete Complexity Results	81
12.7	Data Representation	82
12.8	Network Design	82
12.9	Genetic Algorithms	83
13.	NEUROBIOLOGY AND NEURAL NETWORKS	85
13.1	Introduction	85
13.2	Overview of Brain Physiology	85
13.3	Biological Foundation of Neural Networks	89
13.4	Recent Results in Neurobiology	89
14.	CONCLUSIONS CONCERNING ADAPTIVE KNOWLEDGE PROCESSING	95
	REFERENCES	98

LIST OF ILLUSTRATIONS

Figure No.		Page
2-1	Example of a Neural Network and a Nodal Processing Element	6
2-2	Biological Neurons and a Small Biological Neural Network	6
3-1	Seven Tasks that Neural Networks Can Perform	12
3-2	Some Neural Network Models Arranged by Degree of Biological Inspiration and the Type of Learning the Network Supports	14
3-3	Some Neural Network Models Arranged by Task Each Network Performs	15
4-1	A Single-layer Perceptron and Half-plane Decision Regions Formed During Training Using the Perceptron Convergence Procedure	21
4-2	Capabilities of Single- and Multi-layer Perceptrons. Smooth closed contours bound input distributions for two classes labeled A and B. Shading denotes decision regions for class A	23
4-3	A Circular Decision Region (Shaded Area) for Class A Formed by a Two-layer Perceptron After 200 Trials Using Backpropagation	25
5-1	General Feedforward Associative Memory	28
5-2	The Fully-connected Hopfield Outer Product Memory	29
6-1	A Taxonomy of Neural Network Classification and Clustering Models	34
6-2	A Reduced Coulomb Energy (RCE) Classifier and the Decision Region for Class A (Shaded Area) Formed by Five Prototype Nodes	35
6-3	A Hierarchical Feature Map Classifier Which is Trained with Combined Supervised/Unsupervised Training and the Decision Region for Class A (Shaded Region) Formed by this Classifier	36
6-4	The Feature Map Network Uses an Architecture Made of Nodes Arranged in a Two-dimensional Grid (Top) and Requires Neighborhoods (Bottom) to be Defined Around Nodes. The neighborhoods shrink in size over time	40
6-5	Connection Weights in a Feature Map Network Vary Over Time to Approximate the Probability Density Function of the Input Patterns. Line intersections represent weights from two inputs to 100 nodes and lines connect weight values for nearest-neighbor nodes	41
8-1	Multi-layer Perceptron (Top) Which Produced Receptive Fields (Bottom) Similar to Those Measured by Neurobiologists in the Posterior Parietal Cortex of Monkeys	50
9-1	Block diagram of an Isolated Word Recognizer	57

Figure No.		Page
9-2	Decision Regions Formed by a Two-layer Perceptron Using Backpropagation Training and Vowel Formant Data	59
9-3	Time delay Neural Network Used to Classify the Voiced Stops “B,D,G”	60
9-4	Neural Network Called a Viterbi Network that Implements a Viterbi Decoder as Used in HMM Speech Recognizers	61
9-5	A Synaptic Triad that can be Used to Recognize Two-component Pattern Sequences	63
9-6	A Model Called a Masking Field can be Used to Detect Pattern Sequences	64
10-1	The CMAC Cerebellum Model Forms Nonlinear Input/Output Mappings Useful for Robotic Control	68
10-2	A Block Diagram of the Darwin III Simulated Automaton	71
13-1	Three Types of Neuron-to-Neuron Connections Called Synapses	86
13-2	A Hierarchy of Important Processing Stages in the Visual Cortex of the Macaque Monkey. The majority of connections are reciprocal and inputs from the eye enter V1	88
13-3	Range Cell and Frequency/Intensity Topological Maps Measured in the Auditory Cortex of the Bat. Neurons in these maps are responsive to specific characteristics of the returned bat sonar signal	91

LIST OF TABLES

Table No.		Page
3-1	Six Representative Neural Network Models	16

1. OVERVIEW

1.1 INTRODUCTION

Neural networks are systems made of many simple processing elements operating in parallel whose function is determined primarily by the pattern of connectivity. These systems are capable of high-level functions, such as adaptation or learning, and/or lower level functions, such as data preprocessing for visual or auditory inputs. Neural networks are inspired both by biological nervous systems and mathematical theories of learning, information processing, and control.

1.2 NEUROBIOLOGY

The design of neural networks draws heavily on developments in the field of neurobiology. Recent neurobiological studies have demonstrated neural plasticity with training in the somatosensory (sense of touch) cortex of mammalian brains [186] and in the hippocampus (an area of the brain required to form long-term memories) [24,164,243]. Studies have also been successful in determining the function of sensory areas in the brain responsible for early vision and hearing. Little is known of the functioning of high-level brain structures such as the neocortex. Neurobiological studies have also led to a detailed understanding of some complex neural structures, such as those in the bat used to process sonar return signals [251] and those in the owl used to localize prey via auditory inputs [136,137,146].

Some neural network models are biologically unrealistic and use overly simplified neurons. Other more realistic and complex models are being developed and explored as simulation and analysis techniques improve and as results from neurobiological studies are disseminated. For example, recent neurobiological research has demonstrated that many types of map-like data representations are used in biological brains [17,137]. Some of these map-like representations are now being used in neural network models [17,148,268]. Other work has demonstrated that complex types of temporal processing and nonlinear operations are performed in neurons. Complex neuron models are also being used in neural network models [48,67,150,215,225,257].

Finally, biological studies have demonstrated that there are multiple distinct neural structures in the brain which interact to perform various sensory functions. Neural network researchers have begun to explore more complex multi-module structures that share this characteristic of biological brains [114,148,215].

1.3 MATHEMATICAL THEORY

Mathematical theory in the field of neural networks builds on results from many other fields of science – including pattern classification theory, information theory, nonlinear system theory, optimization theory, machine learning theory, game theory, automata theory, population dynamics, and evolution theory. Neural network researchers are, however, establishing a unique and strong theoretical foundation by exploring

the limitations and capabilities of specific networks and algorithms and by exploring the theoretical limitations of learning in different contexts. This research has been surprisingly successful in a number of areas. These include:

- **New Parallel Computer Architectures.** The approach to achieving realtime response using fine-grain parallelism has led to large supercomputers, such as the Connection Machine [102], and analog VLSI chips for hearing and vision preprocessing modeled after the cochlea and retina [183,185]. This approach has also led to parallel analog architectures to implement some of the most promising algorithms in the fields of vision and hearing [140,158,207] as well as almost all traditional classifiers [142,157]. New neural network classifiers often require less memory and simpler computations during training than more conventional approaches and provide faster responses by using fine-grain parallelism.
- **Learning Theory.** The emphasis on learning and adaptation has led to a long string of neurally inspired adaptive classification algorithms beginning with the *perceptron convergence procedure* and the *least mean square (LMS)* algorithm and extending to the more recent *backpropagation*, *feature map*, and *reduced Coulomb energy (RCE)* algorithms. These algorithms have driven theoretical advances in pattern classification and machine learning theory. Recent work is exploring three important areas. One consists of building hierarchical networks which use combined unsupervised and supervised learning. Such networks have been suggested by many researchers [96,100,114,264] and recent work has demonstrated rapid formation of complex mappings with little supervised training [96,114].

Recent work is also exploring an area called “distribution-free learning” which is associated with the work of Valiant [127,265]. This work focuses on determining those types of input/output mappings that are learnable. Here a mapping is considered learnable if the time it takes to learn it grows less than exponentially in the number of inputs. This work indicates that specific types of data representations and network structures are necessary to provide rapid learning for classification and decision problems.

Other empirical and theoretical work involves examining the properties of robot-like automata that learn by interacting with objects in a simulated environment [215,221,256]. Some of these automata build internal models of the environment over time and can then use these models to plan future actions. Although the necessity of building internal models of the world has often been noted [15,247,255], it is only recently that inroads have been made in this area. This work is opening up an important area of learning where internal models of a limited environment are built through interaction and do not have to be pre-programmed into the network. Learning algorithms using this approach could solve one of the most important problems in the field of artificial intelligence – that of obtaining and using common sense knowledge.

- **New Parallel Neural Network Algorithms.** A focus on fine-grain parallelism has led to the development of new algorithms for speech, vision, and robotics applications. These new algorithms either provide better performance, faster response, or are easier to implement in parallel architectures than more conventional algorithms. New highly parallel algorithms represent one of the major approaches in the vision field. Neural network algorithms are also beginning to be explored in the fields of speech recognition and robotics.
- **Explaining and Modeling Neurobiology.** Developments in computational neural networks have the side effect of helping to model neurobiological systems. This is typified by the recent use of backpropagation to generate receptive fields similar to those of posterior parietal neurons [283]. As neurobiological techniques become more advanced and the response of many neurons can be measured simultaneously, neural network models will become more and more important in analyzing data and suggesting directions for further study.

1.4 CONCLUSIONS

Further basic research is necessary before important new neural network applications should be expected. This research should focus on basic issues of learning, data representation, and network structure in the application areas of vision, speech, and robotics.

Overall, this is a fascinating research field. It has good potential for providing realtime vision and speech recognition and for developing more powerful automatic learning techniques and capable learning automata. Although this field has received little funding, its potential has been demonstrated by recent empirical results with multi-module networks and learning automata, by recent theoretical results, and by serious interest expressed by not only students, but also by seasoned researchers and Nobel laureates.

2. BACKGROUND

2.1 INTRODUCTION

As noted in the beginning of *Chapter 1: Overview*, neural networks are systems made of many simple processing elements operating in parallel; their function is determined primarily by the processing elements' pattern of connectivity. Despite the simplicity of their components, these systems are capable of high-level functions, such as adaptation or learning, as well as lower level functions, such as data preprocessing for visual or auditory inputs; they are inspired by biological nervous systems.

Adaptive neural networks can be trained using *supervised, unsupervised, or self-supervised training*. Supervised training requires labeled training data and a teacher who provides error information. Unsupervised training forms internal clusters automatically with unlabeled data. Self-supervised training uses internal monitoring and error correction to improve performance without an external teacher.

Neural networks are characterized by their topology and the internal data representations used to represent external variables. Many representations, such as topological maps, are motivated by representations observed by neurobiologists.

The recent resurgence of interest in neural networks is due to many factors, including new algorithms and models, new neurobiological information, the availability of computers for simulations, new interest in parallel algorithms, and continuing interest in modeling brain function.

2.2 THE PANEL'S CHARTER

The Neural Network Study's *Adaptive Knowledge Processing Panel* (Panel 2) was mandated to gather together those mathematical and neurobiological findings that form a theoretical framework for the field of neural networks. Important models, learning algorithms, mathematical proofs and derivations were reviewed by the Panel, as were recent developments in neurobiology.

This chapter introduces neural network models and a taxonomy used to classify these models. The following chapters review important models as well as developments in neurobiology and cognitive science and other related fields. They also provide an assessment of the theoretical basis of the neural network field and recommendations for further work.

2.3 DEFINING 'NEURAL NETWORK'

It is worth repeating again: a neural network is a system composed of many simple processing elements operating in parallel whose function is determined by network structure, connection strengths, and the processing performed at computing elements or nodes. As noted already, neural networks can perform high-level tasks, such as adaptation or learning, or low-level tasks, such as preprocessing sensory input data for vision tasks or speech recognition. Neural network architectures are inspired by the architecture

of biological nervous systems, which use many simple processing elements operating in parallel to obtain high computation rates. Figure 2-1 illustrates this definition.

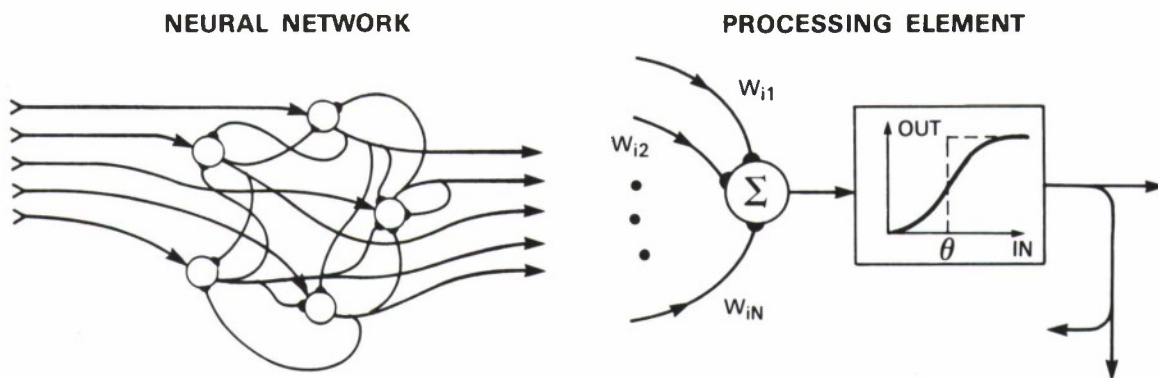


Figure 2-1. Example of a Neural Network and a Nodal Processing Element.

A small interconnected neural network is presented on the left side of this figure and one simple type of processing element or node is presented on the right side. This particular node sums all inputs and passes them through a nonlinearity. Nodes are almost always nonlinear, typically analog, and may be slow compared to modern digital circuitry. Nodes may also include temporal integration and other types of time dependencies and also mathematical operations more complex than summation.

Architectures and processing elements used in neural network models are simplified versions of those observed in biological nervous systems. Figure 2-2 illustrates a number of different types of biological neurons and a small biological neural network.

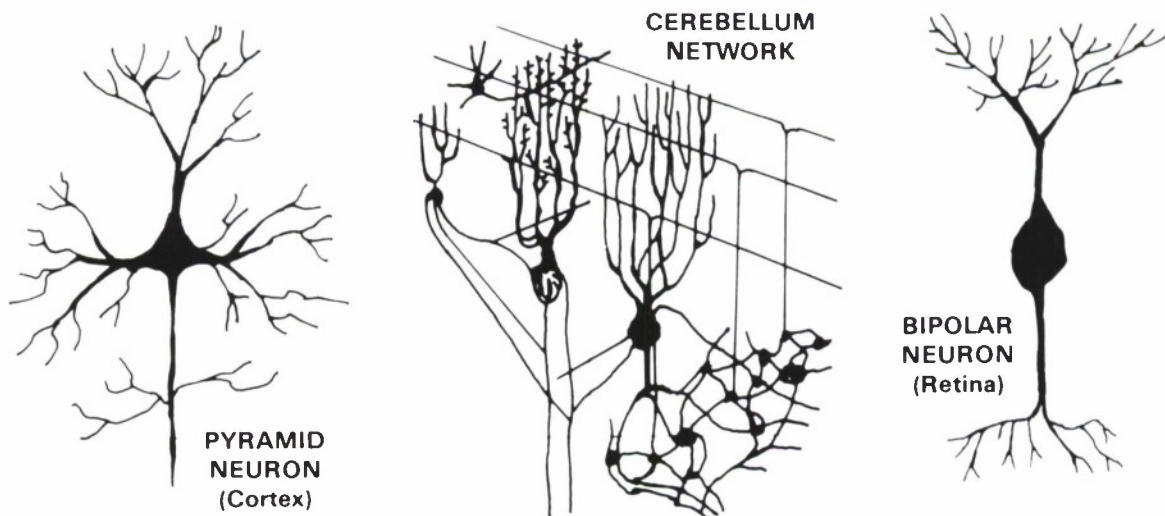


Figure 2-2. Biological Neurons and a Small Biological Neural Network.

As can be seen, biological networks contain many types of neurons, many sub-networks, and rich connectivity. Characteristics of biological neural networks that artificial neural network models hope to provide include:

- Fault tolerance to loss of a small number of computational elements,
- Insensitivity to small variations between computational elements,
- The need for primarily local connectivity and local learning rules,
- Realtime response, and
- Parallelism.

2.4 TRAINING PROCEDURES

Adaptive neural networks can be trained using using three types of training procedures:

- **Supervised training**, which requires labeled training data and an external teacher. The teacher knows the desired correct response and provides a feedback error signal after each trial. This is sometimes called *reinforcement learning* or *learning with a critic* when the teacher only indicates whether a response was correct or incorrect and does not provide detailed error information.
- **Unsupervised training**, sometimes called *self-organization*, uses unlabeled training data and requires no external teacher. Data is presented and internal categories or clusters are formed which compress the amount of input data that must be processed at higher levels without losing important information. Clustering is an important component of many pattern classification procedures [55,98]. It is sometimes called *vector quantization* when used to convert analog inputs into a binary form suitable for transmission or storage [168].
- **Self-supervised training** is used by automata which monitor performance internally and require no external teacher. For example, automata which learn to track a moving spot by controlling simulated eye muscles can generate an error signal based on the distance between the position of the spot on a simulated retina and the center or fovea of the retina. Self-supervision is sometimes called *learning-by-doing* or *learning by experimentation*.

2.5 TOPOLOGY

Patterns of connectivity between nodes vary across neural network models. Nodes may be only **locally connected** to neighbors, **fully connected** to all other nodes, or **sparsely connected** to a few distant nodes.

In addition, networks may be layered with exclusively **feedforward** connections from lower to higher layers as in multi-layer perceptrons [225] or provided with recurrent **feedback** connections as in fully-connected Hopfield networks [109].

Networks with recurrent connections must iterate over many cycles to produce an output, while feedforward networks must operate only until the inputs propagate to the output layer. Feedback networks may be unstable for certain conditions, while feedforward networks have guaranteed stability. An unstable network will have outputs that oscillate, that vary chaotically over time, or that lock up to fixed values.

2.6 DATA REPRESENTATION

Past work on artificial intelligence and neurobiology has demonstrated the importance of data representation. This importance has been substantiated by research with neural network models which can support many different types of internal data representations.

Distributed representations, where the value of an internal variable is represented by outputs of many nodes over a wide region, are used, as are **local representations**, where the value is represented by outputs of only a few nodes [226].

In addition to this dichotomy, variables can be represented using value unit and variable unit coding [17]. **Variable unit coding** encodes the value of a variable as the amplitude of the output of a node. **Value unit coding** encodes the value of a variable as the location of an active node. Such nodes are often arranged in an orderly fashion to form a topographic map of some external variable such as frequency of a tone (tonotopic map). In the extreme, each node represents one possible value of the variable. More typically, coarse coding, coarse-fine, or interpolation codings [67,226,18,268] are used where a few nodes are used to represent a variable value with high precision. These nodes have overlapping response regions as are found in biological networks.

2.7 RECENT INTEREST AND POTENTIAL

Artificial neural network models have been studied for many years in the hope of achieving human-like performance in the fields of speech recognition, machine vision, and robotics. The recent resurgence of interest in neural networks is due to many factors, including:

- The development of new training algorithms [4,105,225] and network design procedures [112,258].
- The demonstration of the ability of relatively simple networks to perform human-like tasks, such as associative recall [109,111] and forming text-to-phoneme rules [232]. Also the demonstration that neural networks can solve complex combinatorial problems, such as the traveling salesman problem [112,258].
- A renewed interest in adaptive algorithms and the desire for systems that self-organize and learn from examples to reduce programming time.

- The demonstration that parallel analog neural networks can be implemented in VLSI at high densities [182,184,183].
- The realization that human-like tasks, such as visual object recognition and speech recognition, are extremely difficult and will require massive parallelism for realtime response.
- Interest in new parallel computer architectures resulting in computers such as the Connection Machine with 64,000 parallel processors [102].
- Developing experience with different data representations suitable for parallel processing, and the demonstration of the importance of data representation from the fields of artificial intelligence and neurobiology.
- The relatively recent widespread availability of affordable, powerful computers for simulation studies.
- A continuing interest in building systems that have some of the parallelism, adaptivity, and fault tolerance of biological brains.
- The presumed failure of more conventional approaches to problems in the fields of vision, speech, and robotics.

Although much of the current interest was inspired by work on a few relatively simple models [109, 112,232], the Adaptive Knowledge Processing Panel found that a solid theoretical foundation is being built, complex multi-module neural network structures are being developed, and researchers from many disparate fields are cooperating to explore and develop this new field. Work that illustrates the potential of this new field was reported at the recent IEEE Conference on “Neural Information Systems – Natural and Synthetic ” held in Denver in November, 1987. In the remainder of this Part, the Panel reviews some of this new work, which has led to greater theoretical understanding and design principles for complex systems.

3. TASKS NEURAL NETWORKS PERFORM AND REPRESENTATIVE MODELS

3.1 INTRODUCTION

Neural network models can perform a wide variety of tasks useful for speech, vision, and robotics problems. The primary tasks performed include:

- Pattern classification,
- Self-organization or clustering,
- Associative memory storage and access,
- Vision and speech preprocessing,
- Computational tasks such as those required to solve combinatorial optimization problems,
- Nonlinear input/output mapping such as is required in robotics control,
- Sensory fusion of inputs from multiple sensors, and
- Recognition of time-varying patterns as is required for speech recognition.

More than 30 different models have been developed. Important models include the *Hopfield network*, *single- and multi-layer perceptrons*, the *cerebellar model articulated controller (CMAC) network*, the *feature map network*, *Darwin III*, and the *silicon retina chip*. Many other important networks and multi-module systems are being explored.

3.2 TASKS THAT NEURAL NETWORKS PERFORM

The field of neural networks includes many different models designed to address a wide range of problems in the primary application areas of *speech*, *vision*, and *robotics*.

Two separate classes of models have been explored. **Neurobiological models** closely model some aspect of biological brain function and/or animal or human behavior. **Computational models** perform some technically important function. Neurobiological models are judged by how well they summarize existing neurobiological and psychophysical data and on the accuracy of predictions. Computational models are judged primarily by their potential impact and by their performance and implementation efficiency. The Neural Network Study and the efforts of the Adaptive Knowledge Processing Panel have focused on computational models due to their potential technical importance.

A recent thorough review of neurobiological models can be found in [42], and the importance of these models is discussed in *Chapter 14: Neurobiology and Neural Networks* of this Part of the Neural Network Study Technical Report.

Researchers developing computational models tend to focus on one or more of the following common problem areas:

- Designing neural network architectures to implement important conventional architectures.
- Developing new highly parallel neural network algorithms.
- Analyzing algorithms using simulation and theory.
- Implementing algorithms in hardware.

Most researchers focus on neural networks that perform those seven major tasks illustrated graphically in Figure 3-1. These tasks include:

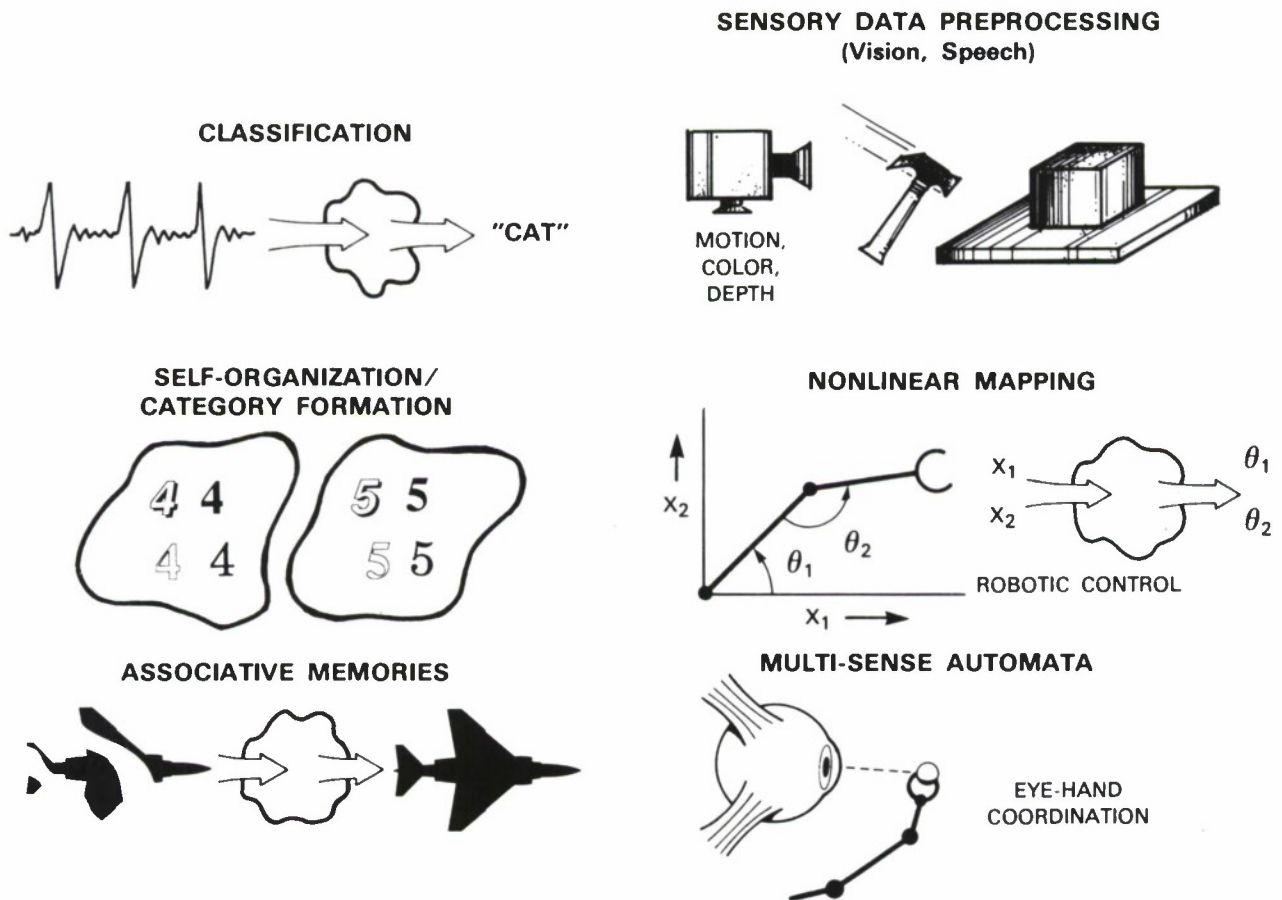


Figure 3-1. Seven Tasks that Neural Networks Can Perform.

- **Classification.** Classifiers are trained with supervision using labeled training data to partition input patterns into a pre-specified number of groups or classes. These

could represent different words for a speech recognizer or different objects for an visual image classifier. Inputs to a classifier may be binary- or continuous-valued.

- **Self-organization/Category Formation.** Self-organizing networks partition input examples into groups or clusters using unlabeled training data. This type of clustering or vector quantization is an efficient technique for reducing information that must be processed at higher levels with little loss in performance [55]. It also makes good use of the large amount of unlabeled training data that is typically available in speech and vision problems. The number of clusters formed may be pre-specified or determined from the examples.
- **Associative Memory.** An associative, or content-addressable memory provides a complete memory item from a key consisting of a partial or corrupted version of the memory. For example, it might return a complete article citation from only the author's name or a complete image of a face from only the bottom half.
- **Sensory Data Processing.** An enormous amount of realtime preprocessing is performed in the peripheral sensory vision and hearing centers. Neural networks can perform this function in real time using massive parallelism.
- **Computational Problems.** Custom neural network architectures can be designed to solve specific computation problems, such as the traveling salesman problem and other constrained optimization problems, using nonlinear analog computation.
- **Nonlinear Mapping.** Many neural networks can map a vector of analog inputs into an output vector using a nonlinear mapping function which can be learned from training data. These types of mappings are useful in many areas, including robot control and nonlinear signal processing.
- **Multi-sensor Automata.** A number of complex, multi-module neural network automata have been built with visual input and a robot arm to manipulate objects in an environment. These automata demonstrate how an eye or camera can learn to scan a scene using self-supervision, how control of a multi-jointed arm and hand can then be learned using self-supervision, and then how the eye and hand can be coordinated to perform simple tasks. These automata also demonstrate how inputs from multiple sensors can be fused to provide classification performance better than could be achieved with a single sensor.

3.3 ILLUSTRATIVE NEURAL NETWORK MODELS

Past work has lead to many different networks which address the above problems and computational tasks. Some of these networks are presented in Figures 3-2 and 3-3. Figure 3-2 emphasizes the biological inspiration and types of training supported by the networks and Figure 3-3 emphasizes the tasks the networks perform.

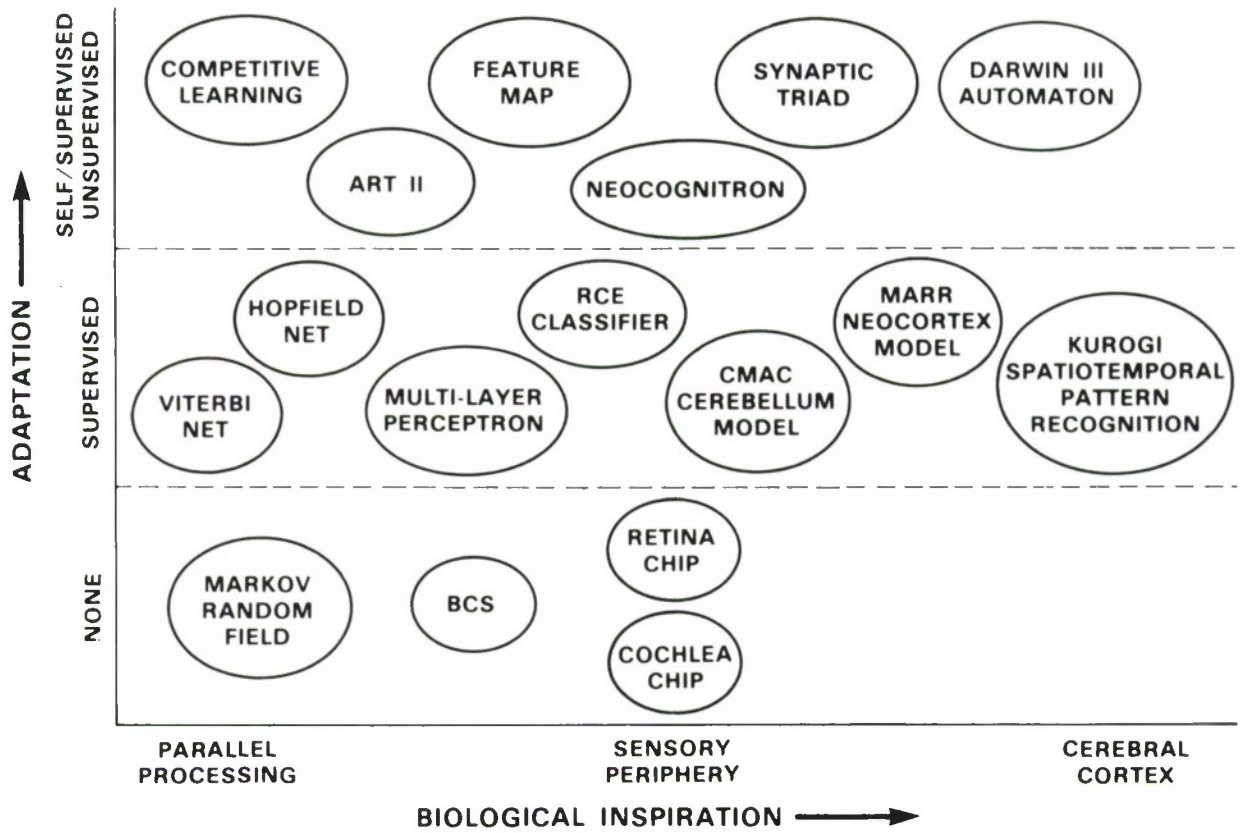


Figure 3-2. Some Neural Network Models Arranged by Degree of Biological Inspiration and the Type of Learning the Network Supports.

NEURAL NETS FOR A VARIETY OF TASKS

PATTERN CLASSIFICATION	SPEECH RECOGNITION	MACHINE VISION	ROBOTICS	SIGNAL PROCESSING	OPTIMIZATION/COMPUTATION
ADALINE (LMS)	MARTIN SPEECH PREPROCESSOR	BGS/FCS	CMAC CEREBELLUM MODEL	ADALINE	BOLTZMANN MACHINE
ART	MASKING FIELD	CELLULAR AUTOMATA	DARWIN III	MULTI-LAYER PERCEPTRON	CELLULAR AUTOMATA
BOLTZMANN MACHINE	MULTI-LAYER PERCEPTRON	CONNECTIONIST MODELS	INFANT		MARKOV RANDOM FIELD
COMPETITIVE LEARNING	SILICON COCHLEA	DARWIN II	MULTI-LAYER PERCEPTRON		TALK-HOPFIELD NETS
FEATURE MAP	SYNAPTIC TRIAD	HIGH-ORDER NETS	SENSOR NETS		WINNER-TAKE-ALL NET
HAMMING NET	TIME-CONCENTRATION NET	MARKOV RANDOM FIELD	TOPOGRAPHIC MAPS		
MULTI-LAYER PERCEPTRON (Back Propagation)	TIME-DELAY NET	NEOCOGNITRON			
PERCEPTRON	TRACE	PARAMETER NET			
REDUCED COLOUMB ENERGY (RCE)	VITERBI NET	SILICON RETINA			

Networks in Figure 3-2 are positioned vertically according to the type of adaptation used during training and horizontally according to the relationship between the network and biological nervous systems.

Networks in the lower third of Figure 3-2 use fixed weights and are not adapted during use. Networks in the middle third use supervised training, and networks in the upper third use unsupervised training or are self-supervised.

Networks in the left of Figure 3-2 are computational models with little connection to biological networks other than fine-grain parallelism. Networks in the middle have architectures modeled after early hearing and vision areas, and networks toward the right are attempts to model higher cortical areas. Since present knowledge of cortical function is minimal, these rightmost networks are only rough guesses that remain to be verified.

More detail on six important networks from Figure 3-2 is provided in Table 3-1.

Network	Function	Inter-Connects	Adapt	Type of Learning	Biological Inspiration
Hopfield Network	CAM, Energy Minimization	Feedback, Full	Yes	Supervised	Parallel Processing
Multi-Layer Perceptron (Back Propagation)	Classification, Nonlinear Mapping	Feed-forward, Full	Yes	Supervised	Parallel Processing
CMAC	Robotics, Nonlinear Mapping	Feed-forward, Full	Yes	Supervised	Cerebellum Model
Feature Map	Speech, Vector Quantization	Feed-forward, Full	Yes	Un-Supervised	Cortical Maps
Darwin III	Eye-Hand Coordination Automaton	Complex	Yes	Self-Supervised	Neuronal Groups
Retina Chip	VLSI Vision Front End	Grid, Local	No	None	Retina

Table 3-1.

Six Representative Neural Network Models

Networks in the lower third of Figure 3-2 are used for early vision processing (Markov random field, BCS, retina chip) and for speech preprocessing (cochlea chip). The *Markov random field network* [75, 140,207] represents a neural network architecture for an algorithm useful for early vision processing. The *BCS, or boundary contour system* [92], is a new neural network algorithm that forms boundaries used for early vision. The *retina chip* [184] and *cochlea chip* [183] are analog VLSI integrated circuits modeled after the retina and cochlea. The retina chip includes photodetectors with a logarithmic output and a wide dynamic range, circuitry to average inputs locally in space, and circuitry to enhance temporal changes

in contrast. The size of the initial retina chip is small (48-by-48 pixels) and it requires primarily local connections as shown in Table 3-1. It includes roughly 100,000 transistors, and has been demonstrated using live inputs. The retina and cochlea chips are good examples of the potential of integrating sensors and parallel neural processing in a compact package.

Networks in the middle third of Figure 3-2 are trained with supervision. Hopfield introduced many current researchers to the field of neural networks through his work with a recurrent network commonly known as the *Hopfield network*. This is a fully-connected iterative network, as shown in Table 3-1, which can be used as a content-addressable memory [109,111] or to solve combinatorial optimization problems such as the traveling salesman problem [112,258].

The *multi-layer perceptron* trained with a new algorithm called *backpropagation* [225] also introduced many researchers to the field of neural networks. It has been used successfully for many difficult problems, including forming text to phoneme rules [234], as a classifier in speech and vision problems [225,267,114,159], to classify sonar targets [82], and to form nonlinear mappings useful in nonlinear signal processing [151] and robotics [244]. As shown in Table 3-1, a multi-layer perceptron is a layered feedforward network where weights are adjusted with supervision to provide the desired output during training.

The *Viterbi network* [158] is a neural network architecture which implements a successful temporal decoding algorithm used for speech recognition using nonlinear analog processing. The *reduced Couloumb energy (RCE) classifier* [216,231,217] differs from the other classifiers in Figure 3-2 in that extra nodes are added when an error occurs due to the presentation of a novel input far from inputs which have already been seen. This allows rapid single-trial learning and the ability form complex decision regions rapidly. The *Marr/Albus cerebellum model* [5,171], called the *CMAC model* in Table 3-1, was originally a model of the cerebellum. This is a part of biological brains that is required to coordinate fine motor movements. The CMAC model forms complex nonlinear maps adaptively and has been used in a number of robotic applications [5,187,125].

The *Marr neocortex model* [170] is an example of an attempt to determine the function of cell layers in the neocortex, as is the *Kurogi spatiotemporal pattern recognition model* [150]. Kurogi's model is unique among high-level models in that it is one of the few to model internal neuron dynamics and specify how nerve action potentials are generated and processed on a spike-by-spike basis. Most other models include variables that represent the average cell firing rate. This is a neurobiological model which makes predictions whose details can be tested by neurobiologists.

The upper third of Figure 3-2 includes models that are trained without supervision or with self-supervision. The three left-most networks vector-quantize the inputs and form clusters. *Competitive learning networks* [227] form internal clusters from binary inputs while *Kohonen's feature map* forms clusters from continuous-valued inputs. Both networks form clusters using a fixed number of nodes and a sequential algorithm similar in purpose to the conventional k-means clustering algorithm [55]. The feature map algorithm is unique in that nodes representing clusters become arranged in a two-dimensional spatial grid or map where nearby nodes respond to stimuli with similar characteristics. The *adaptive resonance theory (ART) clustering algorithm* [32,31] forms a new cluster whenever an input is too far from an already existing cluster.

The *synaptic triad network* [48] is the only network in Figure 3-2 other than the Viterbi network that can recognize temporal pattern sequences. It self-organizes to form pattern sequence detectors that could be

used in speech recognition applications or other applications, including vision, involving recognition of temporal patterns. The *neocognitron* [72] is a multi-layer image classifier that can recognize handwritten characters and provides translation invariance. Feature detectors in each layer are created using competitive learning techniques. The right-most model, *Darwin III* [57,215] is a complex simulated automaton that learns to follow a moving target and touch the target with a multi-joined arm. It is the most recent instantiation of a developing theory of brain function called Neural Darwinism [57].

3.4 IMPORTANT TRAINING AND PERFORMANCE ISSUES

The chapters which follow review in detail the neural network models mentioned above, as well as others, using the taxonomy from Figures 3-1 and 3-3. Reviews of single- and multi-layer perceptrons (*Chapter 4*), associative memories (*Chapter 5*), classifiers (*Chapter 6*), and recurrent networks (*Chapter 7*) are followed by chapters dedicated to the application areas of vision (*Chapter 8*), speech (*Chapter 9*), and robotics (*Chapter 10*).

These chapters are not meant to be a thorough historical treatment. They are, rather, meant to be an overview that supplements reviews available in [11,67,90,103,157,226,235]. This review emphasizes the fundamental theoretical issues addressed by each network. These include:

- **Capability.** What tasks can this network perform? Is there a rich enough data representation available to represent a solution to the problem the network must solve? Does the network have the computational power to solve the problem?
- **Credit Assignment.** Given an existing set of variable weights, how is credit assigned to each weight during training for a correct or incorrect response?
- **Training Issues.**
 1. How much training data is required for a given level of performance?
 2. How long does it take during training until internal weights converge to final values?
 3. How does training time change as the problem is scaled up in size? Complexity analysis determines whether this time grows as a polynomial or exponentially in the number of inputs or nodes.
 4. How well does a classifier generalize? Given a limited set of training data, how well does the network perform on other test data not seen during training?
 5. Can new items, classes, or behaviors be added without disrupting current performance?
 6. Is the network response invariant to irrelevant variability in the input? How is this invariance obtained?

7. How is training time affected by internal data representations?

● **Performance/Operation Issues.**

1. How long does it take until the output settles and reaches the desired solution?
2. How accurate is the solution?
3. How is the network designed, and how does performance and capacity vary with network size?
4. How sensitive is the solution to component inaccuracies?
5. How fault-tolerant is the network when components are damaged?
6. How easy is the network to implement with different types of hardware?
7. Is network operation stable? Do iterations in networks with recurrent connections lead to a stable state where node outputs are constant, or do node outputs vary continuously?
8. How is performance affected by internal data representations?

4. SINGLE- AND MULTI-LAYER PERCEPTRONS

4.1 INTRODUCTION

Single- and multi-layer perceptrons are the simplest types of feedforward neural network models. Single-layer perceptrons can be trained using the perceptron convergence procedure or the LMS algorithm and can be used as classifiers and for adaptive signal processing. They can, however, only form simple half-plane decision regions. A three-layer perceptron can form any decision region region and can approximate any desired nonlinear input/output mapping. Such networks can be trained with a new algorithm called backpropagation. These networks have been successfully used in many classification problems as well as prediction and robotics problems where nonlinear mappings are required. They can form the decision region required by any classifier and could thus replace any conventional classifier.

4.2 SINGLE-LAYER PERCEPTRONS

Single-layer perceptrons were first introduced by Rosenblatt [222]. A single-layer perceptron can be used to classify a continuous-valued or binary-valued input vector into one of two classes. One such network is illustrated in the left of Figure 4-1. A single node computes a weighted sum of the input

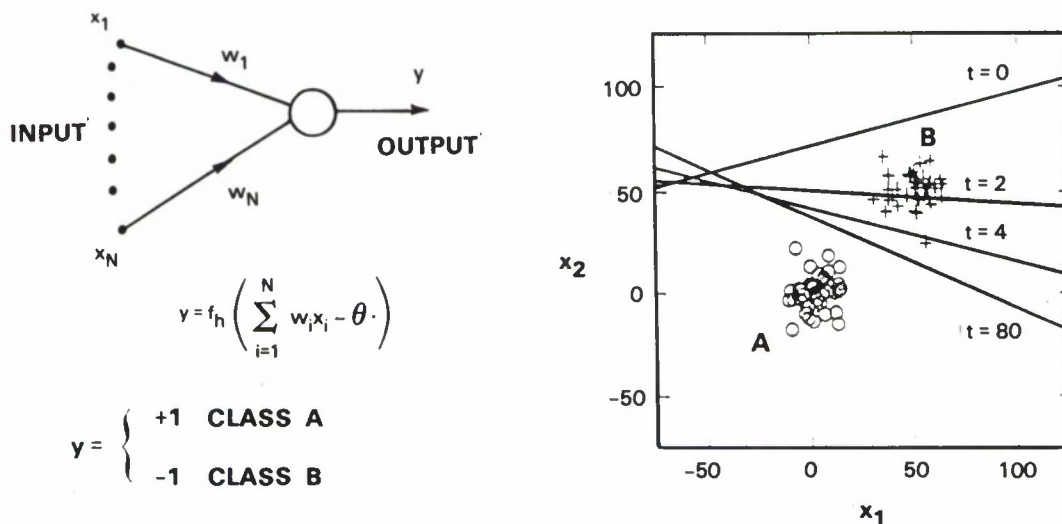


Figure 4-1. A Single-layer Perceptron and Half-plane Decision Regions Formed During Training Using the Perceptron Convergence Procedure.

elements, subtracts a threshold, and passes the result through a binary-valued nonlinearity denoted $f_h(\cdot)$. Each of the two possible outputs corresponds to a different classification response. The perceptron forms two decision regions separated by a hyperplane.

This hyperplane is a line as shown in the plot on the right of Figure 4-1 when there are only two inputs. This plot illustrates how weights change when a supervised error-correction training algorithm called the

perceptron convergence procedure [55,222] is used to adapt weights. In this example, there are two input classes denoted by crosses and circles. The decision boundary changed on trials 2, 4, and 80 after errors occurred.

Rosenblatt and others [188,194] focused on the capabilities of perceptrons when input patterns (the vector of inputs) are binary. In this case, each input pattern is a vertex on a hypercube where the hypercube has as many dimensions as there are inputs. If N represents the number of inputs, then there are 2^N possible input patterns in this case. Rosenblatt proved that if patterns in two input classes could be separated by a perceptron, then the perceptron convergence procedure would converge and find a solution. Such sets of patterns are called *linearly separable*. Further work demonstrated that a perceptron with M inputs could form any desired grouping or dichotomy of fewer than 2^M random input patterns with high probability [194]. Perceptrons are thus often said to have a capacity of 2^M patterns because they can almost always dichotomize any two classes made by selecting from 2^M binary patterns.

Other early work in an area called *threshold logic* explored the potential application of perceptrons as logic gates in computers [117,154,193]. Such gates can realize Boolean AND and OR functions and other linearly-separable functions, including “at least X of M .” This work led to procedures to determine those Boolean functions which are linearly separable for small numbers of inputs. It also led to network design procedures to realize desired Boolean functions with networks of perceptrons and a better understanding of the capabilities of perceptrons. Design procedures for feedforward and feedback networks of threshold logic gates were developed, culminating in the construction of some small computers [117]. This work ended with the availability of inexpensive logic gates using simpler Boolean AND, OR, NAND, and NOR logic functions.

Minsky and Papert carefully analyzed the capabilities of perceptrons in a book called *Perceptrons* that was recently revised and reprinted [188]. They demonstrated that perceptrons could not distinguish connected from unconnected figures on an input array of binary pixels. They also demonstrated that perceptrons couldn’t compute an exclusive OR Boolean function because this function is not linearly separable. A single-layer perceptron and decision regions for an exclusive OR function are illustrated in the upper row of Figure 4-2. The smooth closed contours labeled A and B in this figure are the input distributions for the two classes for a network with two inputs. The shaded areas are the decision regions created by single- and multi-layer perceptron classifiers. Distributions for the two classes for the exclusive OR problem on the left are disjoint and cannot be separated by a single straight line. One possible decision region for class A which a perceptron might create is illustrated by the shaded region in the first row of Figure 4-2. A more serious limitation of single-layer perceptrons noted by Rosenblatt [222] is that multiple simultaneous inputs on an input pixel array (for example, a simultaneous circle and square) lead to ambiguous responses. A good review of Rosenblatt’s book and Minsky and Papert’s work can be found in [169].

Only recently have complexity theory results been obtained which set bounds on the time required to train perceptrons to learn various linearly-separable Boolean mappings [96,97]. For example, it was proven that the time to learn an arbitrary linearly-separable function grows exponentially with the number of inputs. This time, t , is bounded by

$$2^M < t < M^M,$$

where M is the number of inputs to the perceptron. It was also proven, however, that the time to learn the “at least X of M ” function grows only polynomially with the number of inputs. The time to learn this


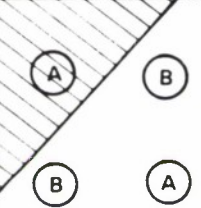
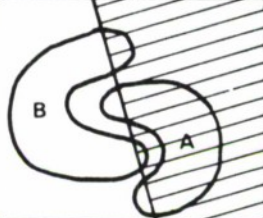


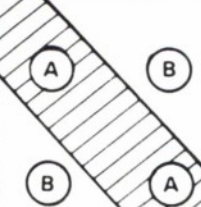
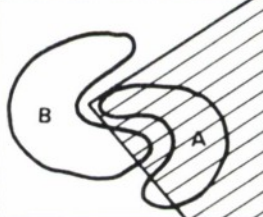
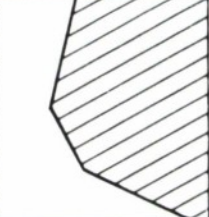

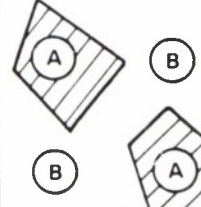
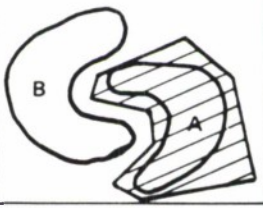

STRUCTURE	TYPES OF DECISION REGIONS	EXCLUSIVE OR PROBLEM	CLASSES WITH MESHED REGIONS	MOST GENERAL REGION SHAPES
 SINGLE-LAYER	HALF PLANE			
 TWO-LAYER	TYPICALLY CONVEX			
 THREE-LAYER	ARBITRARY			

Figure 4-2. Capabilities of Single- and Multi-layer Perceptrons. Smooth closed contours bound input distributions for two classes labeled A and B. Shading denotes decision regions for class A.

function is bounded by:

$$t < M^3.$$

Finally, the time to learn the “at least 1 of M ” function, which is a logical OR, was proven to grow even slower with the number of inputs:

$$t < M^2.$$

In empirical tests, this time was found to grow linearly with the number of inputs. These results demonstrate the utility of forming input data representations such that the Boolean function to be learned is an OR function that scales very well in large networks. They suggest hierarchical networks where data representations in lower layers require only an OR function to be computed in a final layer with adjustable weights. Such networks have been suggested by many researchers [96,114,264].

A special version of the perceptron convergence procedure designed to learn an OR function (called a *disjunctive Boolean function*) was recently developed by Littlestone [162]. He found that the number of errors that occur when learning an OR function using this algorithm grows very slowly with the number of inputs. The number errors for a perceptron with M inputs is bounded by $2M(\log_2(M) + 1) + 1$.

Connection weights and the threshold in a perceptron can also be adapted using the LMS algorithm [55,275]. This algorithm is used primarily when inputs take on continuous instead of binary values. The LMS algorithm uses a semilinear nonlinearity and adapts weights after every trial based on the difference between the actual and desired output. It is more appropriate when classes are not linearly separable. The LMS algorithm is one of the most technologically successful neural network algorithms. It resulted from early work of Widrow and Hoff on Adaline networks [275] and is currently used extensively in telephone echo cancelers and high-speed modems [276]. Work on adaptive algorithms such as the LMS algorithm led to a new branch of engineering called Adaptive Signal Processing. Much of this work is summarized in [276]. For example, it was proven that this algorithm converges when the gain term (a multiplicative factor on the change in weights after each trial) is in a given range. Relationships between convergence time, the RMS error after convergence, and the statistics of the input patterns are also available [276].

4.3 MULTI-LAYER PERCEPTRONS AND BACKPROPAGATION

A more complex structure than the single-layer perceptron is required when classes cannot be separated by a hyperplane. Two such situations are presented in the upper row of Figure 4-2. Distributions for the two classes for the exclusive OR problem on the left are disjoint and cannot be separated by a single straight line, as noted above. Input distributions for the second problem shown in this figure are meshed and also cannot be separated by a single straight line. Situations similar to these may occur when parameters such as formant frequencies are used for speech recognition.

Multi-layer perceptrons are feedforward networks with one or more layers of nodes between the input and output nodes. These additional layers contain hidden nodes that are not directly connected to both input and output nodes. Two- and three-layer perceptrons are shown in the middle and bottom rows of Figure 4-2. Multi-layer perceptrons overcome many limitations of single-layer perceptrons, but were generally not used in the past because effective training algorithms were not available. This has recently changed with the development of a new training algorithm called backpropagation. This algorithm was

reported as early as 1974 [274] and rediscovered by a number of workers [195,225]. This algorithm is a generalization of the LMS algorithm that uses a gradient search technique. It is named for the procedure used to pass the error signal back from the output to lower layers. Although it cannot be proven that the backpropagation algorithm converges, it has been shown to be successful for many problems of interest [225]. Multi-layer perceptrons trained with backpropagation have been used successfully in such diverse applications as forming letter-to-phoneme rules [234], classifying spoken vowels and digits [114,159], identifying phonemes in a speech recognizer [267], and for nonlinear signal processing [151]. Figure 4-3 illustrates how a circular decision region can be formed after 200 trials of training by a two-layer perceptron trained with backpropagation.

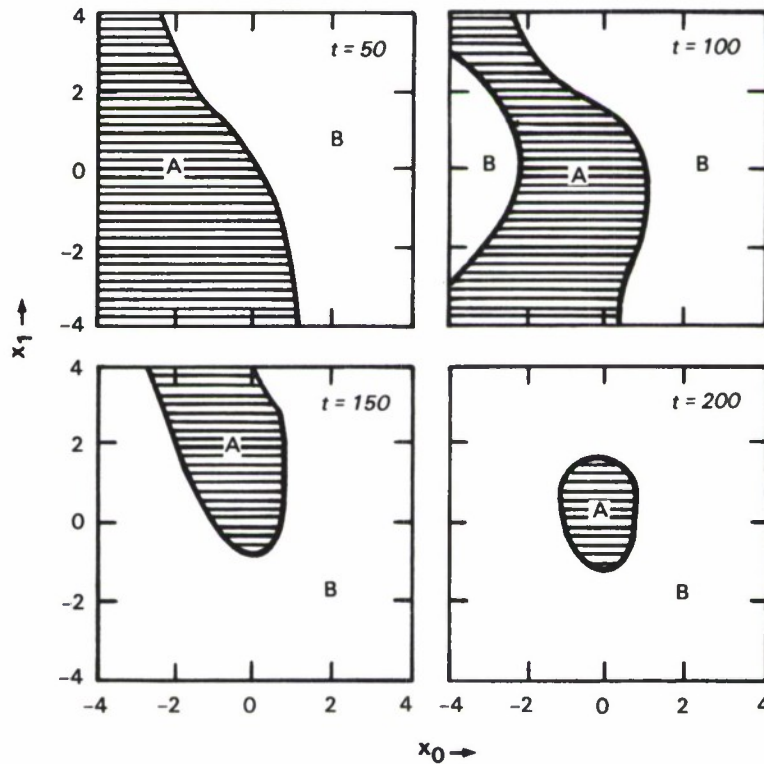


Figure 4-3. A Circular Decision Region (Shaded Area) for Class A Formed by a Two-layer Perceptron After 200 Trials Using Backpropagation.

The utility of the backpropagation algorithm stems from the surprising computational power of three-layer perceptrons with two hidden layers. These networks can form any desired decision region [29,157], as shown in the bottom of Figure 4-2. They can thus emulate any traditional deterministic classifier by producing the decision region required by that classifier. Kolmogorov also proved a theorem described in [163] which, in effect, demonstrates that a three-layer network can form any continuous nonlinear function of the inputs. This proof requires carefully specified nonlinearities with wide dynamic ranges. More recent theoretical work has demonstrated that continuous nonlinear functions can be approximated to arbitrary precision using three-layer perceptrons with sigmoidal nonlinearities [44]. A three-layer perceptron can

thus create any continuous likelihood function required in a classifier, given enough nodes. Empirical results [151] also demonstrate that many useful nonlinear mappings can be realized with backpropagation.

One difficulty with backpropagation training is that many presentations of the training data are frequently required for convergence when decision regions or desired mappings are complex. This is important during training but does not affect response time during use. It currently sets a practical limit to the size of networks that can be trained but is not a severe problem for single-layer networks or multiple-layer networks with restricted connectivity. Recent theoretical work [122] has investigated the difficulty of training multi-layer perceptrons to learn arbitrary Boolean mappings. It was found that this problem is NP-complete and thus grows exponentially with the size of the network. This is recent work and results for specific classes of Boolean functions and for networks with restricted connectivity patterns remain to be obtained.

A number of algorithms other than backpropagation have been proposed for training multi-layer perceptrons. Some involve training initial layers using some form of competitive learning [96,114,264] or using fixed weights in initial layers [5,114]. The output layer then is trained to learn an OR function, which scales well in large networks. Rapid learning of difficult tasks has been demonstrated with these networks [96,114]. Another algorithm, called the A_{R-P} or *Associative Reward Penalty algorithm* [20], can also be used to train multi-layer perceptrons. It uses simple binary correct/incorrect feedback and does not require a continuous error signal but typically converges much more slowly than backpropagation.

One interesting problem to which backpropagation has been applied is predicting future sample values in time series [151]. Here it was demonstrated that a network could be trained to predict the following samples in a chaotic sequence better than linear or low-order polynomial predictive techniques. Theoretical results and a new training procedure for problems that involve prediction are presented in [252]. This paper argues that one can learn with greater computational and statistical efficiency by taking advantage of the temporal structure of prediction problems. Both formal and simulation results are presented in support of this claim. New methods are closely related to older techniques used to solve the credit assignment problem [228,107]. This work sheds light on and provides a theoretical foundation for these earlier methods. It also shows that such methods are much more broadly applicable than previously thought.

5. ASSOCIATIVE MEMORIES

5.1 INTRODUCTION

A content-addressable memory, or associative memory, allows retrieval of a complete stored pattern from a noisy or incomplete input pattern key. Historically, associative memory modeling has been a major focus of neural network research. The most extensively studied associative memory models are based on storing and retrieving binary patterns. These models typically make use of the Hamming metric, and their connections take on only integer values; this makes them suitable for practical implementation. These models are adapted either by specifying weights *a priori* or incrementally by some simple storage or programming rule.

From the mid-1950s to the mid-1970s, a wide variety of associative memory models were studied. Some of this early work is described in [11,42,143]. Widespread interest in such models seems to have waned by 1975. Although many application areas were explored, these early memory models had little technological impact. The available technology favored the use of digital random access memories (RAMs) and read-only memories (ROMs) in conjunction with conventional computers. Thus, these models did not have a lasting impact beyond the world of academics and were significant primarily because they attempted to explain biological memory in simple mathematical terms.

Hopfield rekindled widespread interest in associative memory when he proposed his 1982 energy minimization model based on the outer product storage rule [110]. While outer product memories had been studied extensively, Hopfield's version captured attention because of its timeliness from a technological standpoint and the theoretical appeal of energy minimization.

A large number of researchers have exhaustively studied the Hopfield model, its variations, and its generalizations [2,7,51,128,179,78,33,147]. A general consensus has developed that the outer product model is extremely interesting but that it suffers from limited capacity and inefficient use of hardware. Since 1982, a number of new memory models have been proposed, and some old models have been rediscovered and improved upon. Noteworthy among these are the *unary* or *Hamming network* model [249,250,259,22,53,160], and the *sparse-distributed models* [22,124]. These models do not suffer from the capacity and performance limitations of the Hopfield model.

5.2 THE UNARY OR HAMMING NETWORK

This network was first described by Steinbuch [249,250] in 1961 and Taylor [259] in 1964. The idea was independently re-proposed in 1986 as a practical alternative to the Hopfield model by Baum, Moody, and Wilczek [22]; Domany and Orland [53]; and Lippmann, Gold, and Malpass [160].

Figure 5-1 shows the network diagram for a general feedforward associative memory. The unary model is a special case of this architecture. There are three sets of units: N input units, G internal units, and N' output units. The input units have discrete activation values $(\pm 1, 0)$, where 0 corresponds to the "don't know" or "don't care" state. The output units have discrete activation values ± 1 , while the internal units have continuous activation values in the range $(0, 1)$. The internal units have inhibitory lateral

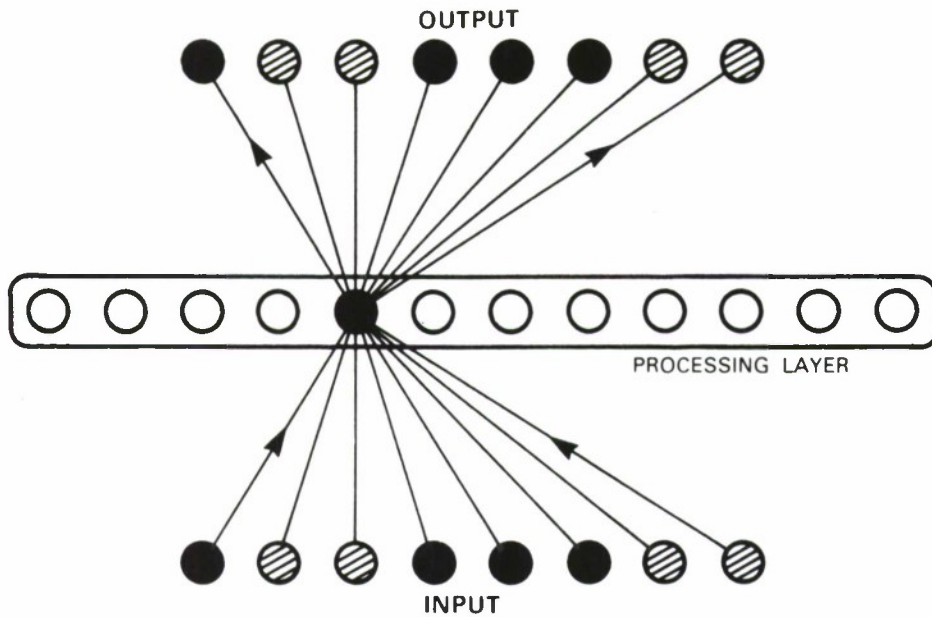


Figure 5-1. General Feedforward Associative Memory.

100205-1

connections which implement a winner-take-all circuit. The network is called the unary model because only one internal unit is active once a match is made. This distinguishes the model from distributed models in which a number of internal units are active when a match is made.

Each of the G internal units is associated with a pair of input and output memory registers. Each memory register is implemented as a vector of ± 1 resistive connections. Input vectors have N bits, while output vectors have N' bits. The memory capacity M , defined as the number of input/output memory pairs which can be stored, is limited by $M \leq G$. Note that M can be very much greater than both N and N' .

The associative memory function is computed by the network as follows:

1. A partial or noisy input key pattern is imposed as an activation pattern V_i on the N input units. Internal units compute the dot product between the inputs and weight vectors in parallel. The result for each node is N minus twice the Hamming distance between the input key vector and the stored weight vector.
2. The node with the largest input current corresponds to the best match in Hamming space between the input key vector and the stored vectors. The selection of the best match is accomplished by an analog winner-take-all circuit.
3. Once the internal units have converged to determine the winner, only one of them will be active. The sole "on" unit transmits a pattern of activity to the output units, completing the associative recall task.

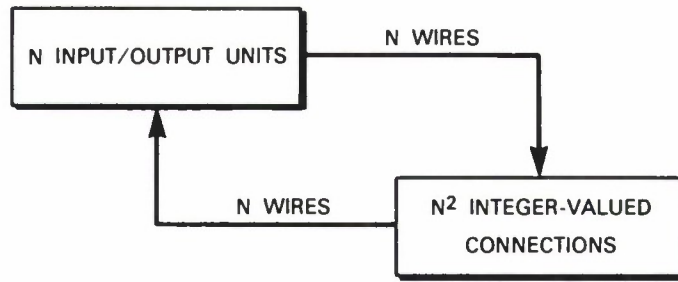


Figure 5-2. The Fully-connected Hopfield Outer Product Memory.

5.3 THE HOPFIELD MODEL

The Hopfield outer product memory, shown in Figure 5-2, has only one layer of units. These serve triple duty as input, output, and processing units. The units are globally interconnected and every unit is thus connected to every other unit. The dynamics of the discrete version of the model are given by matrix multiplication and thresholding:

$$V_i(t+1) = \text{sign}\left(\sum_{j=1}^N T_{ij} V_j(t)\right). \quad (5.1)$$

Here $V_i = \pm 1$ is the state of each of N units, and T_{ij} is the matrix of connections. The N units are updated *asynchronously* at random times. Hopfield showed that when the matrix T_{ij} is symmetric, the dynamics of equation 5.1 minimize a bounded energy function. A weight assignment rule, called the *outer product storage prescription*, provides such a symmetric T_{ij} .

The processes of retrieving information from a Hopfield memory begins by putting the network into an initial state which represents the input key pattern. The network then iterates from this initial state until it converges to a final stable state. Ideally, this final stable state would be the stored memory which is the closest in Hamming sense to the initial state key pattern.

Performance of the Hopfield Model. Much has been written about the dynamics, capacity and performance of this model and its variations. Only key points are summarized here.

The maximum number of memories M which can be stored in a Hopfield memory while still obtaining perfect recall is [2,179]

$$M \leq N/(4 \log N).$$

If more memories are stored, then the stable states of the network begin to differ significantly from the stored words. If an error rate of 5% can be tolerated, then the capacity is limited by [110,7]

$$M \leq 0.14 N. \quad (5.2)$$

These numbers should be contrasted with the capacity of the unary memory, which can have $M \gg N$ with no errors in the final state.

The hardware efficiency of the Hopfield model also is poor. Storage of M memories with N bits of information each requires N^2 connections each with $2M + 1$ possible values. The hardware efficiency ϵ , defined as the number of information bits per connection state, is therefore:

$$\epsilon = \frac{M}{N(2M + 1)} < \frac{1}{2N}.$$

This compares with $\epsilon = O(1)$ for the unary model.

The dynamics of the Hopfield model are complex. The basins of attraction for the stored memories tend to be irregular [128] and fill only a fraction of the volume of Hamming space. The remaining volume is occupied by the basins of spurious attractors which were not intended to represent actual memories [7]. The number of spurious attractors increases as the the number of memories stored reaches the capacity limit given in equation 5.2. Beyond this limit, the network “forgets” everything and no retrieval is possible.

A number of variations on the Hopfield memory have been proposed. These include the *pseudo-inverse model* [144,51], *higher-order correlation models*, and various generalizations including the *bi-directional associative memory (BAM)* [147]. None of these provides the hardware efficiency of the unary model.

5.4 SPARSELY-DISTRIBUTED FEEDFORWARD MODELS

Two sparsely-distributed feedforward models, the *Kanerva* [124] and *BMW* [22] models, are worthy of note. These models are sparsely-distributed in the sense that each memory is stored in a small fraction of all physical locations, and each location contains information about only a small fraction of all the memories stored. This is in contrast to the outer product and correlation matrix models in which all memories are stored at all physical locations.

The sparsely-distributed memories are analogous to the Unary memory in that they have three layers of units and store memory vectors in two layers of registers. Unlike the unary memory, each memory is stored in a set of registers and each register contains a superposition of stored memories.

Like the unary memory, the capacities of the sparsely-distributed models are not limited by the word sizes N , but rather by the number in internal units G . The capacity constraint for the BMW model is [22]

$$M < \frac{G}{\beta^2}, \tag{5.3}$$

where β is the signal-to-noise ratio required at the output layer. The capacity constraint of the Kanerva model is approximately [129]

$$M \leq 0.1G$$

beyond which spurious states – that is, unwanted states – begin to appear.

Sparsely-distributed memories fall short of the unary memory in terms of hardware efficiency. This is caused by the necessity of integer-valued connection weights rather than binary ± 1 connections and because superposed storage and consequent memory interference causes a substantial loss in capacity. The unary model is more hardware-efficient, even if it is replicated several times to achieve hardware fault-tolerance. The most appealing characteristic of the sparsely-distributed models, especially the Kanerva

model, is that they are the most biologically plausible of all the models yet proposed. Both Kanerva and Baum *et al.* have pointed out similarities between their models and the Marr [172] and Albus [6] models of the cerebellum.

6. CLASSIFICATION AND CLUSTERING MODELS

6.1 INTRODUCTION

There has been a continuing interest in neural network classification and clustering algorithms because of their ability to provide realtime response through fine-grain parallelism and because of the importance of these tasks. As noted in [214], classification of external objects based on sensory inputs is an essential requirement for learning, logic, and other mental functions.

Classification of static patterns is an area where neural network models have been very successful. Work has led to architectures which can be used to implement all important conventional classifiers using fine-grain parallelism [21,114,142,157,194]. In addition, as mentioned above, multi-layer perceptrons can form those decision regions required by any classifier and can thus simulate any traditional classifier. Work has also led to the development of new hierarchical classification algorithms which extend the field of pattern classification by providing adaptation and rapid learning using parallel architectures [31,96,114,142,217,231,264]. This chapter reviews work on classifiers focusing on classification of static patterns. Work on classifying time-varying patterns is reviewed in the chapter on speech recognition. A detailed review of neural network classifiers is available in [157] and recent comparisons of classifiers are available in [96,114].

6.2 A TAXONOMY OF CLASSIFIERS

A taxonomy of important neural network models that can be used to classify and cluster static patterns is presented in Figure 6-1. For clarity, this figure omits higher-order models and the ART II model [31], which is an extension of ART I for continuous-valued inputs.

The taxonomy in Figure 6-1 is first divided between networks with binary- and continuous-valued inputs. Below this, networks are divided between those trained with and without supervision. Networks trained with supervision include perceptrons and multi-layer perceptrons described above. These networks are provided with labels that specify the correct class for new input patterns during training. Most traditional statistical classifiers, such as Gaussian classifiers [55], are trained with supervision using labeled training data. Networks trained without supervision, such as the Kohonen's feature map forming networks [143,142], are used as vector quantizers or to form clusters. No information concerning the correct class is provided to these networks during training. A further difference between networks, not indicated in Figure 6-1, is the connectivity patterns of the networks. All models in this taxonomy – except the Hopfield, ART, and DARWIN II networks – use predominately feedforward connections.

The algorithms listed at the bottom of Figure 6-1 are those classical algorithms which are most similar to or perform the same function as the corresponding neural network. In some cases, a network implements a classical algorithm exactly. For example, the Hamming or unary network described above is a neural network implementation of the optimum classifier for binary patterns corrupted by random noise [73]. It can also be shown that the perceptron structure performs those calculations required by a Gaussian classifier [55] when weights and thresholds are selected appropriately.

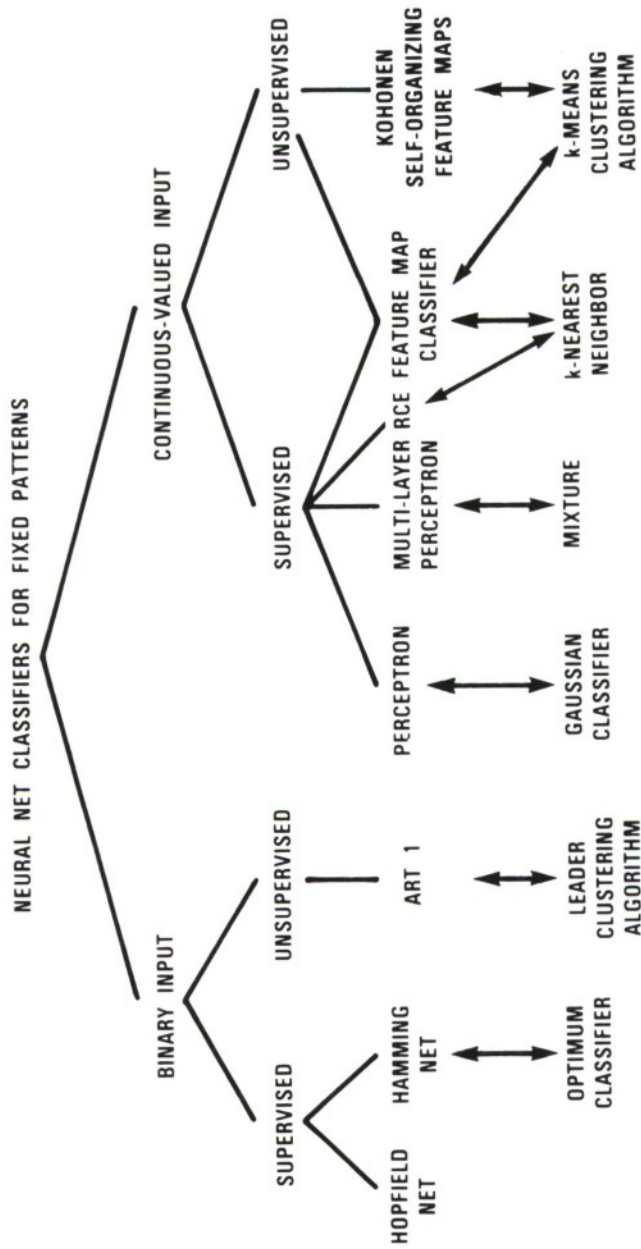


Figure 6-1. A Taxonomy of Neural Network Classification and Clustering Models.

In some cases, the neural network algorithms represent new classification algorithms that extend the theory of pattern classification. For example, perceptrons trained with the perceptron convergence procedure [222] behave differently than Gaussian classifiers; this led to the theoretical work described previously. The RCE classifier [217,231] is a unique extension to k-nearest neighbor classifiers that limits memory requirements and can be implemented using adaptive fine-grain parallelism. Kohonen's feature map network [143] performs an adaptive version of the k-means training algorithm in a manner that forms topographic maps similar to those that occur in the brain. The feature map classifier [114] combines an input layer organized using unsupervised training and the feature map algorithm with an output layer trained using perceptron-like techniques. Hierarchical networks such as this can form complex decision regions rapidly [96,114].

6.3 SUPERVISED CLASSIFIERS

The single- and multi-layer perceptron networks, the Hopfield network, and the Hamming network classifiers in Figure 6-1 have already been discussed in detail. The remaining supervised classifiers include the RCE classifier, the feature map classifier, and high-order networks. As mentioned above, the RCE, or reduced Coulomb energy, classifier is an extension to k-nearest neighbor classifiers that is adaptive and limits memory requirements. It is similar to classifiers described in [21] and can form arbitrary decision regions using hyperspheres with adaptable radii and position as shown in Figure 6-2. This figure shows an RCE network on the left and decision regions formed by this network using hyperspheres on the right.

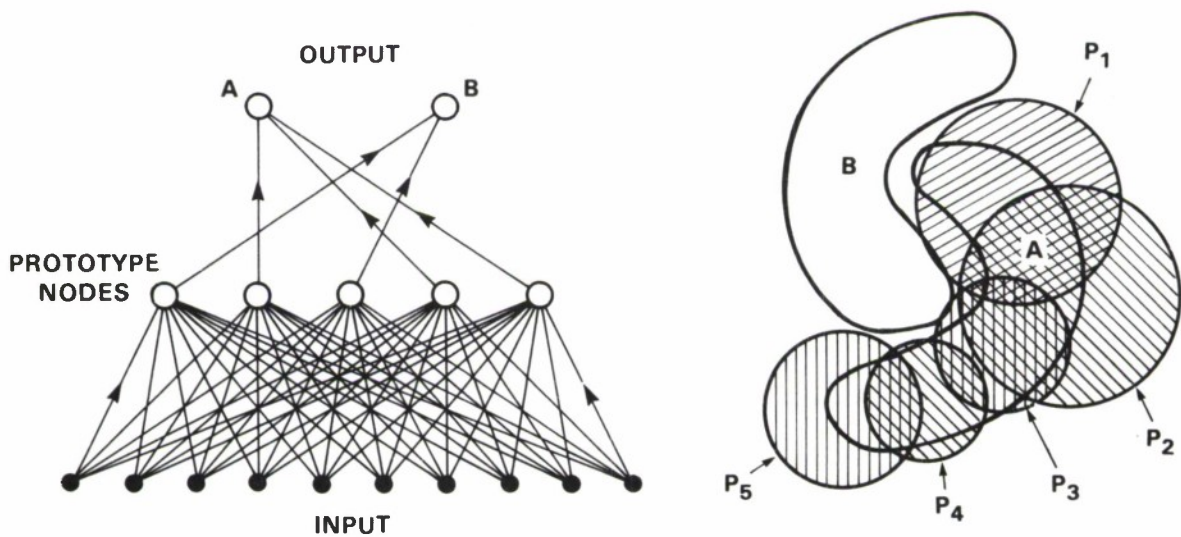


Figure 6-2. A Reduced Coulomb Energy (RCE) Classifier and the Decision Region for Class A (Shaded Area) Formed by Five Prototype Nodes.

6.3.1 RCE CLASSIFIER

The RCE classifier represents a unique approach to the problems of generalization and rapid single-trial learning. It limits the number of exemplars or nodes required, leads to a simple network realization, provides rapid single-trial learning, and eliminates the need to select a global maximum over many nodes when making a final classification decision. Theoretical analyses and experiments with this network [96, 217,231] demonstrate that it can form complex decision regions rapidly and can solve the exclusive OR problem in very few trials. Hampson [96] provides bounds on learning for specific problems and notes that this network can learn a specific instance in only one trial, while a perceptron requires a number of trials on the order of the number of inputs. He also demonstrates by simulation that this network can learn Boolean problems, called the symmetry problem and the multiplexer problem, more than an order of magnitude faster than a multi-layer perceptron trained with backpropagation. The RCE classifier also can be implemented efficiently on serial machines and has been used extensively for classification problems.

6.3.2 Feature Map Classifier

The feature map classifier listed in Fig 6-1 is a hierarchical network that uses combined unsupervised and supervised learning. It is representative of a number of hierarchical classifiers that have recently been developed [96,264] and is similar in design to the counterpropagation network described in [100]. It is presented in Figure 6-3.

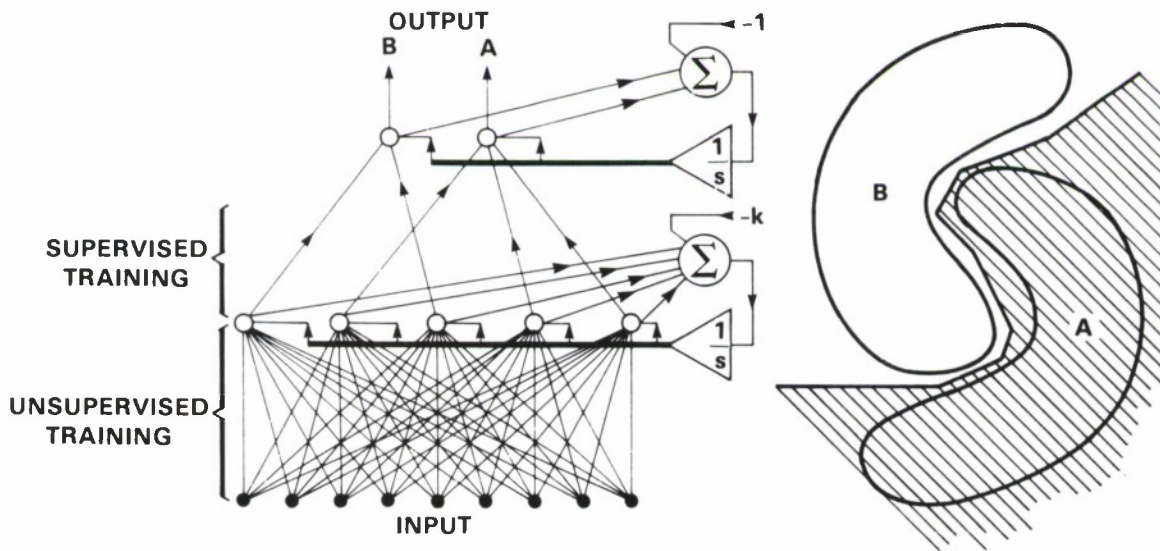


Figure 6-3. A Hierarchical Feature Map Classifier Which is Trained with Combined Supervised/Unsupervised Training and the Decision Region for Class A (Shaded Region) Formed by this Classifier.

The lower stage in Figure 6-3 vector quantizes the input and is trained first using Kohonen's feature map algorithm with unsupervised training data. The second perceptron-like stage is then trained with supervision. This approach is useful when much unsupervised data is available, as in the areas of vision and speech, but little supervised training is provided. As noted above, the second layer can be trained rapidly because it is computing a simple OR function to form classes. This hierarchical structure thus greatly reduces the amount of supervised training required. A theoretical analysis of this type of combined supervised/unsupervised training is available in [38]. This analysis demonstrates that the amount of supervised training required for a given level of performance can often be greatly reduced when unsupervised training is available. Similar results were obtained empirically by simulation in [114,96] and proven for some specific Boolean mappings in [264]. The feature map classifier can be used to implement a k-nearest neighbor classifier when weights are adjusted appropriately.

6.3.3 High-order Networks

Normal neural networks include simple processing elements that operate on linear functions of input variables. High-order networks include more complex elements that operate on high-order products and powers of input variables as well as on linear terms [225,77]. These networks have a long history [55,188,194,205]. Single-layer perceptrons which use high-order neurons can learn polynomial mappings useful for classification by using an extension of the perceptron convergence procedure or the LMS algorithm for training. This can eliminate the need for multi-layer networks and provide extremely rapid learning for problems where it is known *a priori* that a polynomial mapping is appropriate.

High-order networks can also provide excellent generalization by specializing the selection of high-order terms to match the specific problem. This specialization can provide geometric invariances for networks with inputs derived from a visual field. High-order networks can provide invariance to both a translation or change in scale of the input pattern. This invariance is pattern independent and is similar to the invariance provided by such pattern recognition feature operators as Fourier descriptors. In addition, the invariance is independent of the learning rule and the nonlinearity of the neuron. This invariance can be placed at any level of a multi-layer network, with only one layer of invariant high-order neurons; and all layers can then be trained with an algorithm such as backpropagation [225].

The complexity of high-order networks must be limited by *a priori* knowledge to restrict computation costs because the complexity of individual nodes grows exponentially with their order. Interconnections can be constrained by imposing invariances on the high-order terms and restricting the range of interconnections and the order. Scaling high-order networks is difficult, unless the number of high-order terms can be adequately constrained. Interest in high-order networks demonstrates their versatility in providing solutions to problems where the form of the input/output mapping required can be characterized *a priori*.

6.3.4 GMDH Algorithms

High-order networks and multi-layer perceptrons are related to a class of algorithms known as GMDH algorithms [63]. These algorithms, in essence, build a high-order network to solve modeling and classification problems. They are based on Kolmogorov's theorem, mentioned previously [163], which states that

any nonlinear function can be approximated by a multinomial. Given that a set of observations are noisy samples of a continuous phenomena, the modeler's job is to find the multinomial which "best" describes the observations. The main problem is to determine the order of the multinomial and the input variables which bear a relationship to the output variables. GMDH algorithms represent a heuristic solution to this problem.

They build a network of low order, usually multinomials of order two in two variables which, when fed forward using multiple layers, realize a multinomial of arbitrary degree. The procedure at any layer is to form products and linear terms from all possible pairs of outputs from from the previous layer. Weights to an output are formed using linear regression techniques to fit a quadratic to all pairs. Given n input variables, the number of quadratics generated is 2^{n-1} and the procedure is thus untenable without intervention. This is accomplished by applying a criterion after each layer has been added and tested which retains the "best" performers and drops the rest. Generally, the number of retained variables at the output decreases from layer to layer so that the procedure doesn't become unstable. More importantly, however, are the criteria employed to terminate the procedure and produce the model. Without them, the procedure would produce an exact fit to the data. These criteria take different forms. The most prominent criterion uses a cost function that includes a penalty term for each additional layer and another term equal to the mean squared error generated by the regression. The procedure stops when the next layer doesn't do any better than the last. At termination, the order of the multinomial and the important input variables will have been learned.

The algorithm has been used for classification and modeling. It has performed well in applications including modeling of econometric time series data and missile guidance laws. As with other high-order networks, GMDH algorithms are most useful when the underlying phenomena are not well understood and *a priori* constraints can be applied.

6.4 UNSUPERVISED CLASSIFIERS

The ART network, Kohonen's self-organizing feature maps, and the Darwin II network in Figure 6-1 cluster inputs using unsupervised training.

6.4.1 ART

ART, or adaptive resonance theory, networks [32,31] are complex nonlinear recurrent networks with feedback connections. These networks form a new cluster (adjust weights to and thus activate a new internal node) whenever an input pattern is sufficiently different from previously stored patterns. What constitutes 'sufficiently different' is determined by a global internal parameter called the *vigilance parameter* which must be adjusted externally to provide the desired sensitivity to differences in input patterns. The underlying traditional algorithm used in these networks is called the *leader clustering algorithm* [98]. ART networks have one of the most involved and complex architectures of all the classifiers listed in Figure 6-1. They have recently been included in a number of more complex multi-module neural network system designs [90].

6.4.2 Self-organizing Feature Maps

Kohonen [143] presented a sequential clustering algorithm included in Figure 6-1 which produces what he calls self-organizing feature maps similar to those sensory maps that are common in the neocortex. Maps are two-dimensional grids of nodes as shown in Figure 6-4. Each node represents a cluster center. Kohonen's algorithm insures that nodes which are spatially close in the grid respond best to input vectors that are close in a Euclidean sense. This algorithm is a sequential version of the k-means clustering algorithm [55] that does not require storage of all training tokens and uses a simple weight modification rule.

Kohonen's algorithm creates a vector quantizer by adjusting weights from common input nodes to output nodes arranged in a two-dimensional grid. Figure 6-5 illustrates how connection weights to nodes in the grid vary over time to sample the space spanned by the input examples. In this case, the input examples were uniformly distributed over the area plotted. After training, an input will cause only that output node corresponding to the cluster center nearest to the input to have a "high" output.

Kohonen [143] presents many examples and proofs related to this algorithm. He also demonstrates how the algorithm can be used in a speech recognizer as a vector quantizer [145] and how the algorithm can be modified to form decision regions similar to those formed by a traditional k-nearest neighbor classifiers [142]. This algorithm is a viable sequential vector quantizer when the number of clusters desired can be specified before use and the amount of training data is large relative to the number of clusters desired. It has been used for this purpose in the feature map classifier described above, in the counterpropagation network [100], and in a robotics application [218].

6.4.3 Darwin II

The final unsupervised clustering algorithm in Figure 6-1 is a network called Darwin II [57,215]. This is the second in a series of simulations created to explore a brain theory being developed called "neural Darwinism" [57]. This theory differs from other neural network design procedures in its emphasis on the importance of selection to enhance the outputs of those neuronal groups or subnetworks which respond best to specific input stimuli. The theory assumes that a collection of neuronal groups, called a *repertoire*, is formed during embryogenesis. Groups in a repertoire respond best to overlapping but similar input patterns due to the randomness of neuronal growth. One or more groups in a repertoire will respond to every input pattern and response to important unexpected inputs is thus insured. Training involves competition between groups, which amplifies the responses of specific groups to specific stimuli and associates those groups with each other and with a specific appropriate response.

Neural Darwinism is different from the common approach of designing a network topology and training it with supervision to provide a desired response. Instead, it assumes that there are, by design, many subnetworks. Only those with the desired response during training are selected. Important issues addressed by this approach include the need to respond to unexpected stimuli, the importance of interacting with the environment, and the need to generate biologically plausible models where the elemental units are small groups of neurons. Important issues this approach must address are how to both design and provide the large number of subnetworks required.

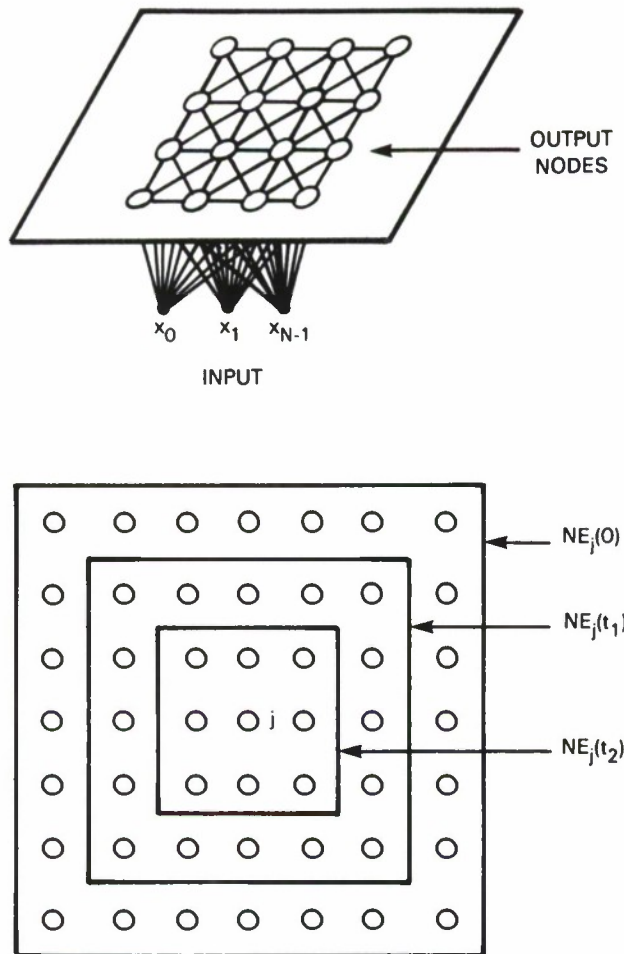


Figure 6-4. The Feature Map Network Uses an Architecture Made of Nodes Arranged in a Two-dimensional Grid (Top) and Requires Neighborhoods (Bottom) to be Defined Around Nodes. The neighborhoods shrink in size over time.

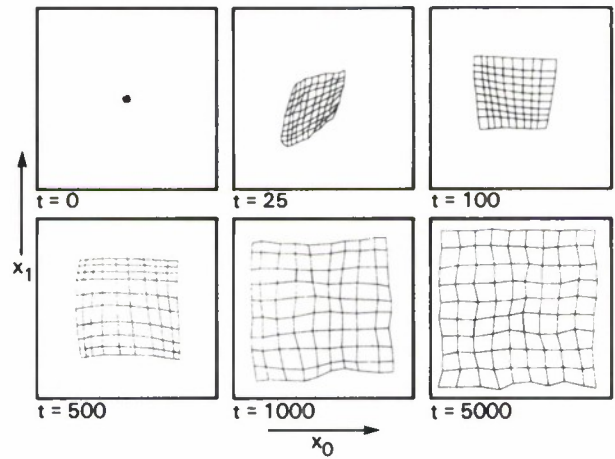


Figure 6-5. Connection Weights in a Feature Map Network Vary Over Time to Approximate the Probability Density Function of the Input Patterns. Line intersections represent weights from two inputs to 100 nodes and lines connect weight values for nearest-neighbor nodes.

7. RECURRENT NETWORKS

7.1 INTRODUCTION

Neural network models with recurrent connections can perform complex time-dependent calculations. Many of these models do not include learning, but use fixed weights to perform some specific function. Hopfield and Tank demonstrated how such networks could be designed using the concept of computational energy to solve combinatorial optimization problems such as the traveling salesman problem. Others have demonstrated how a recurrent network called a “winner-take-all” network can select that node with a maximum value.

One specific class of recurrent networks called *cellular automata* have been studied for theoretical reasons and also for their ability to simulate some types of complex physical phenomena such as fluid flow. A number of training algorithms for recurrent networks have been developed. The first used *simulated annealing* and was incorporated into a recurrent network with symmetric connections called a *Boltzmann machine*.

Recently, backpropagation training has been generalized for application to recurrent networks. A number of stability proofs have been derived for recurrent networks. The most general applies to a wide class of networks with symmetric connection weights.

The study of recurrent networks needs to expand because such networks can perform important functions in complex systems, such as automatic gain control or energy normalization and selecting a maximum. These functions are required for robust operation and to make decisions.

7.2 COMPUTATIONAL ENERGY

As noted in [42], Hopfield [109,112,258] was the first to explicitly use the concept of computational energy to both understand and design nonlinear neural networks. This concept was applied to recurrent networks with symmetrical feedback connections to tailor the final stable states of such networks to be solutions of specific problems. Hopfield and Tank [112,258] demonstrated how this concept could be used to design analog networks to solve many combinatorial optimization problems. These included the traveling salesman problem, linear programming problems, and a task assignment problem. Their work is an excellent illustration of the energy function design methodology and appears to produce networks that work well when the number of nodes is small. The effectiveness of the networks with large number of nodes, however, has not been demonstrated. Current results suggest that the networks may not scale well. It appears to be difficult to obtain good solutions when there are many variables in combinatorial optimization problems and there are thus many nodes [278]. Other neural network approaches to these problems are being explored, however, and these may scale better than current techniques.

A recent important development for recurrent networks is a generalization of the backpropagation training procedure [204]. Previous attempts to apply backpropagation to recurrent networks involved duplicating the network at every time step to form a gradually unfolding feedforward network [225]. The newer

generalized training algorithm can be applied to networks with asymmetric connections and to networks with symmetric connections as studied by Hopfield and Tank.

7.3 THE BOLTZMANN MACHINE

Combinatorial optimization networks developed by Hopfield and Tank do not always find the best solution in constrained optimization problems. They can, instead, get trapped in local minima of the network energy function. This problem can be alleviated by adding a stochastic aspect to the search performed as the network iterates using a technique called simulated annealing [130]. A network called the Boltzmann machine [4,105] uses this strategy. The Boltzmann machine training algorithm solved the credit assignment problem for the special case of recurrent networks with symmetrical connections and was demonstrated to be able to learn a number of difficult Boolean mappings [4,105]. Although it is theoretically interesting, the use of simulated annealing causes training times in a Boltzmann machine to be too long for practical applications on current hardware.

7.4 HIGH-ORDER NETWORKS

As noted previously, high-order networks include complex computing elements that operate on high-order products and powers of input variables as well as on linear terms. Lyapunov functions can be constructed for recurrent high-order networks with symmetric connections to guarantee stability [152]. High-order networks can also include conjunctive connections and modifiers to perform some logical operations [17,66,104] and a high-order extension of the Boltzmann machine has also been developed [233]. High-order associative memories offer greater storage density and convergence times that are often faster by an order of magnitude than those designed using linear neurons [201].

7.5 CELLULAR AUTOMATA

The field of cellular automata [280] shares with neural networks a common interest in the computational properties of systems composed of large numbers simple interacting elements. The main difference is that neural networks are more general and biologically motivated. Each node in a cellular automaton typically computes the identical input/output function, nodes are typically connected only to nearest neighbors, learning is not typically incorporated, and the overall system is typically synchronous and run by a global clock. The state or output of each node in a cellular automaton is discrete and typically takes on binary or no more than 20 output values. The output state of a node on one clock cycle depends on the previous states of neighbors and that node via a fixed update function that is not just a linear sum but can be any logical function. Since the update function is fixed and the same for all units in a cellular automaton, memory can only be held by the states of the units, rather than by connection weights.

Interest in cellular automata has been mainly theoretic in the past. They can be shown to be equivalent to Turing machines; they can be self-replicating; and they are capable of forming complex patterns and thus exhibiting complex "behavior" given only a few simple rules and specific initial conditions [13,280,209,

266]. The well-known game of Life [209], for example, is a cellular automaton that can form complex visual patterns from specific initial conditions. Interest in cellular automata has recently been revived with the availability of high-speed computers and special-purpose hardware. Much recent work in this area focuses on determining the ability of cellular automata to perform calculations required to simulate physical phenomena such as certain types of fluid flow.

Many neural network models share characteristics with cellular automata. The Trion model of cortical information processing [241] is one example of a hybrid that has characteristics of both cellular automata and neural networks. It is like a cellular automaton in that the state of each unit is discrete and ternary. Each unit represents a group of neurons, such as an orientation column in visual cortex, and the three states correspond to output above, at, or below the background firing rate (the possibility that a group of neurons could have three discrete states has been demonstrated in [279]). The update function in the Trion model depends on a weighted sum of the inputs, as in a neural network, although, like a cellular automaton, the inputs to each unit are restricted to a fixed spatial neighborhood defined relative to the unit, and the neighborhood weight matrix is the same for each unit initially. However, like a neural network, the weight matrix may change with time in accord with a synaptic rule, resulting in unit-dependent weights. The current state is computed from the weighted sum of the two previous time steps. Trion models with fewer than 10 units can produce thousands of temporally periodic firing patterns. These sophisticated oscillations are analogous to the stable states in a Hopfield memory.

7.6 SELECTING OR ENHANCING THE MAXIMUM VALUE

The need to select or enhance the node in a network with a maximum value occurs whenever a decision must be made in network. One network currently called a “winner-take-all,” or “maxnet,” is fully connected with symmetrical connections, except from a node to itself, and is used as a component of many networks [112,143,67,90]. This network was extensively analyzed by Grossberg [86,90]. He demonstrated that maximum values would be enhanced if nodal nonlinearities increased at a faster-than-linear rate. In some applications, strictly feedforward networks can be used as an alternative to pick a maximum value [157]. These feedforward networks require no recurrent connections and provide outputs rapidly and continuously.

7.7 STABILITY

Nonlinear neural networks with feedback connections have the potential of being unstable if designed incorrectly. Outputs of nodes in such networks might latch up to high or low values or vary continuously. A number of proofs have been developed to demonstrate that certain types of networks with feedback are globally stable. The most general proof has been developed by Cohen and Grossberg [36]. This proof demonstrates that a large class of networks with symmetric connections is globally stable. Grossberg has also developed a number of other recurrent subnetworks and explored the stability of these networks. This work is reviewed in [90].

8. VISION

8.1 INTRODUCTION

The primary goal of machine vision is to find the three-dimensional arrangement of objects and surfaces that produce an input two-dimensional array of intensity values. Compact implementations of accurate, realtime vision systems would find widespread use in many applications including:

- Rapid input of handwritten and printed documents and other images into machines;
- Visual surveillance;
- Automatic target detection and tracking;
- Assembly line monitoring and quality control;
- Aids for the blind;
- Visual inputs for automata;
- High-fidelity, low bit-rate image storage and transmission;
- Automatic analysis of medical diagnostic tests such as X-ray pictures, NMR images, and cell cultures;
- Automatic analysis of satellite images; and
- Security systems based on visual recognition.

Unfortunately, current machine vision systems perform poorly when faced with simple vision tasks that a child or bird could perform. Although human beings tend to take vision capabilities for granted, it has proved to be a difficult task to duplicate simple visual recognition tasks with machines. Some of the problems of machine vision are caused by the enormous computational power required to process images (a typical high-quality image front end may produce 100 images every second with 10^6 pixels per image), by the multiplicity of cues (shape, color, texture, size, distance, movement) which must be detected and integrated, and by the difficulties of extracting three-dimensional information from two-dimensional inputs.

Vision is a good application area for neural networks and, for this reason, the neural network approach is not a new paradigm in this field. Parallelism is clearly necessary to obtain any kind of near realtime throughput in a vision system and many researchers have focused on algorithms that use fine-grain parallelism. The early visual system is by far the most observable subsystem of the brain. Its patterns of connectivity have been explicitly mapped through several synaptic layers, and the connections themselves are more sparse and spatially organized than in many other parts of the brain. Many vision algorithms are modeled after or inspired by the early visual areas. Recent vision research is also exploring adaptive

networks that can be trained to perform some of the functions of early vision [115]. Finally, a large behavioral database, which was created by experimental psychologists, is available. Visual perception has been studied parametrically for over a century and has led to many illusions and perceptual effects that can be measured with high accuracy. This database has been used by many workers to suggest new algorithms and determine performance requirements for vision systems.

8.2 EARLY AND HIGH-LEVEL VISION

The vision problem can be broken up into a hierarchy consisting of early vision and high-level vision. *Early vision* processing must segment an image into distinct regions and assign various characteristics – such as shape, color, texture, distance, and motion – to each region. It must also recover intrinsic properties of objects, like color and shape, independent of the particular viewing conditions, such as intensity of the illuminant.

High-level vision processing must recognize objects using information from early vision processing and provide the three-dimensional position of each object. Early vision tasks are generally associated with anatomical structures from the retina through the first few levels of the visual cortex. High-level vision tasks are generally associated with higher levels of the visual cortex.

A key task of early vision is the detection and enhancement of contours or edges, and the corresponding grouping and segmentation of a visual input into regions. This is carried out pre-attentively without the intervention of expectancies from internal stored memories. Contours extracted by early vision processing are often defined as a spatial discontinuities in luminance. Contour extraction is, however, much more difficult because humans are able to detect contours between regions that have no such luminance discontinuities. Contours can also be caused by statistical differences in textural qualities, such as orientation, shape, density, or color; differences in binocular disparity or depth; accretion and deletion of texture elements in moving displays; and by illusions.

8.3 PHYSIOLOGY AND MACHINE VISION

Recent decades have seen remarkable strides in the observational techniques used by physiologists to study the functions of early vision. Combinations of electrode recording, staining, and imaging techniques have yielded increasingly detailed data on cell functions and patterns of connections among cells. After Hubel and Wiesel's pioneering work, the 1960s were a period of optimism for the imminent unlocking of the secrets of early vision. This optimism faded as increasingly complex subdivisions occurred in taxonomies of cell functions, and as increasingly complex perceptual and interdependent patterns of connections, many involving dense feedback, were found. The ordered pyramid of many simple inputs gradually thinning to a few complex objects, while an appealing metaphor, did not seem to adequately characterize early biological vision.

Meanwhile, advances in machine vision have had a curiously parallel set of developments. Many specialized modules that work well in limited domains have been developed, but a unified approach to performing even the most basic tasks of early vision has never emerged. This development is all the more

striking in view of the enormous surge of available computing power in the past few decades. To date, the rate-limiting factor on the production of autonomous early vision systems has been the lack of theoretical understanding of how to build such a system. The interdependence of diverse sources of visual information, which in human perception serves to cooperatively reduce ambiguity, serves instead in machine vision as cross-contaminating sources of noise.

8.4 SURVEY OF REPRESENTATIVE WORK

Far too many detailed neural network models of specific visual processes exist for all to be summarized in this section. For examples, see the proceedings of the IEEE First International Conference on Neural Networks, held in San Diego in June, 1987, and the proceedings of the IEEE Conference on "Neural Information Processing Systems - Natural and Synthetic," held in Denver in November, 1987. The following particular studies have been chosen as representative or illustrative of important trends in methods or results.

8.4.1 Head-centered Frame of Reference

Andersen and Zipser [283] have studied how neurons in the posterior parietal cortex of monkeys can perform the coordinate transformations needed to create a fixed internal head-centered frame of reference while the eyes move about in the head. Simulated visual inputs in retinal coordinates from the retina and inputs from eye muscles that provided information about eye position were fed to hidden units in a multi-layer perceptron as shown in the top of Figure 8-1. The multi-layer perceptron was trained using backpropagation to provide a stable output image in head-centered coordinates. After training, the receptive fields of the hidden units closely resembled those of units in the posterior parietal cortex as shown in the bottom of Figure 8-1. Although the use of backpropagation is unlikely in animals, the response regions formed help explain why specific types of receptive fields are observed and how they can be used to form new internal reference frames.

8.4.2 Receptive Field Analysis

Daugman [45,46] has mathematically analyzed the inherent uncertainties in temporal, spatial, orientation, and spatial frequency resolution for visual receptive fields. This analysis indicates that observed spatial and spectral response profiles of certain classes of cortical neurons are optimized for information resolution over the four response dimensions. In addition, he has decomposed images into sets of quasi-orthogonal "2D Gabor" primitives, that measure the local correlations of oriented structure in images. Gabor primitives exploit the non-random structure or coherence of images, which originates from the optical projections of solid objects that themselves have some unity of reflectance characteristics. They define weighting functions for multiple-scale, oriented receptive fields like those found in the primate cortex.

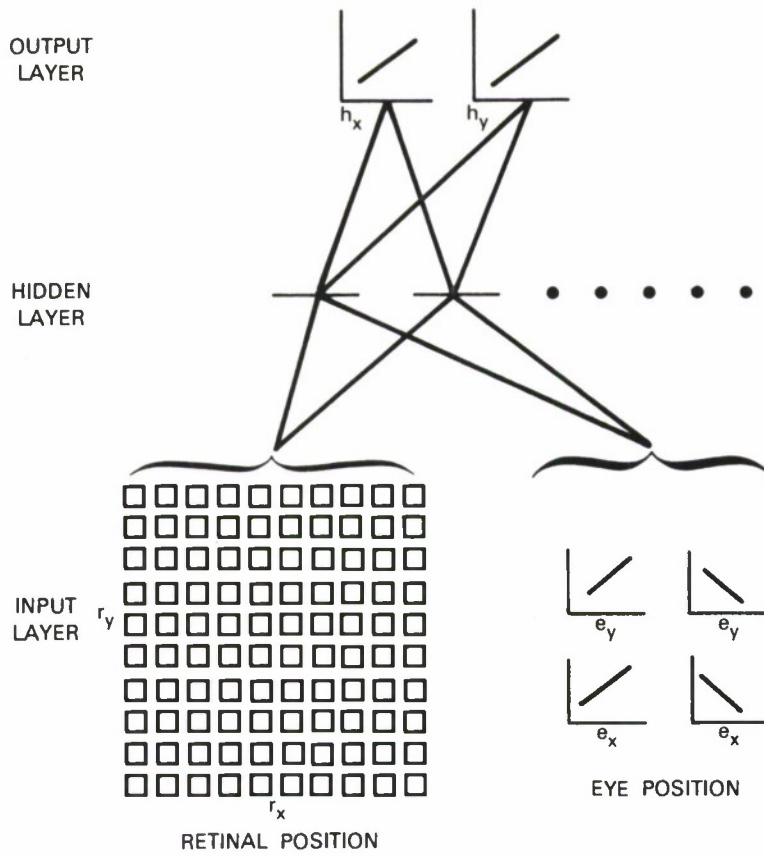


Figure 8-1. Multi-layer Perceptron (Top) Which Produced Receptive Fields (Bottom) Similar to Those Measured by Neurobiologists in the Posterior Parietal Cortex of Monkeys.

8.4.3 Connectionist Models

Feldman and Ballard [17,18,67,64,65] have analyzed connectionist approaches to problems in both early and high-level vision. Their approach has highlighted the importance of selecting the internal data representation that is best suited for specific problems, of fine-grain parallelism, and of studying neurobiology. Their papers provide a good introduction to the field of connectionist models, good overviews of different types of internal data representations useful in neural network models, and a good overview of many vision problems and approaches to these problems. They have implemented a number of vision algorithms on a large parallel computer called a Butterfly machine (see *Part V* of this Report for discussions of neural network hardware). Their group has also both written and distributed a neural network simulation program that is in use at many sites [65].

8.4.4 Neocognitron

The neocognitron [72] is a multi-layer image classifier designed to recognize black and white handwritten characters. It obtains translation invariance by using a multiplicity of identical input units that extract identical visual features from all parts of the input field and by slowly reducing the number of units in higher layers. Feature detectors in each layer are created using competitive learning techniques. This approach is limited by the large number of nodes that are required. Recent work has begun to explore the use of selective attention techniques to recognize multiple simultaneous inputs [71].

8.4.5 Markov Random Fields

Geman and Geman [75] have analyzed and applied Markov random field methods to image processing, including blur or noise removal, boundary detection, texture labeling, and object recognition. Given a model of image formation (including such “prior distributions” as luminance or spectral measures, or likelihood of edge discontinuities), the posterior distribution given the image data is analyzed to find the likeliest generators of the image data. By defining an energy or cost function and “running” the system to equilibrium by simulated annealing, the desired image processing is performed. In the Markov random field approach, the conditional distribution on all variable values reduces to the conditional distribution on neighbors, so iterative computations are bounded by the size of local neighborhoods in which computations must be performed for a given application. The approach requires the construction or postulation of a particular set of image processing constraints or prior distributions for each kind of image structure that one wishes to detect or enhance.

8.4.6 Boundary and Feature Contour Systems

Grossberg and his colleagues [87,88,93,92,95] have developed a massively parallel neural network model designed to perform early vision tasks including boundary detection, segmentation, and filling in of color and brightness. These models attempt to mimic the psychophysical phenomena of “emergent” segmentation. A segmentation is called “emergent” if regions can be perceived to be separate but do not differ in global statistics of luminance distributions or if the perceived boundary separating regions is sharp

and continuous while the luminance patterns are discontinuous or fuzzy. The model addresses perceptual data in boundary detection, completion, and sharpening and grouping of textural regions, through the dynamical interactions of nodes that are functionally identified with cells found in the early visual pathways of primates.

The model makes explicit the distinction between boundaries that form groupings of visual elements and those qualities of surface appearance commonly described by such terms as hue, saturation, or lightness. This distinction rests in part on the complementary attributes of the dynamics of network interactions underlying the boundary/feature distinction. Interactions in the boundary contour system (BCS) are directed inward between two or more orientationally tuned nodes that are sensitive to amount of contrast, but not direction of contrast (light-to-dark versus dark-to-light). A particular arrangement of short-range competition and long-range cooperation among orientation sensitive nodes ensures a rapid, active sharpening of boundaries, including emergent boundaries. Interactions in the feature contour system (FCS) are diffusive, flowing outwards from any single active node, and are sensitive to both amount and direction of contrast.

Patterns of activity in the BCS restrict the diffusion of activity in the FCS, and thus regions of like color or contrast typically match regions formed by boundary segmentations one-to-one. Thus the objects that people recognize in everyday environments generally have surface appearance attributes that are unitary and distinct from appearance attributes of other objects. Many laboratory curiosities, on the other hand (including illusory contours, Glass patterns, Beck/Julesz textural displays, and neon color spreading), present the observer with a seemingly paradoxical scission between recognizable regions and phenomenal color patterns. The network model clarifies how these phenomena are inevitable consequences of the strategies that the primate early visual system has evolved in order to coherently segment and group complex patterns of optical stimulation in real time and with cells of finite, quantized resolution. This work typifies an approach which relies heavily on psychophysical results (primarily results with illusions) to define the details of a complex model. The model has not yet been tested with real images.

8.4.7 A Computational Approach

One theoretical approach to the vision problem was initially proposed in [174,173]. A good summary of this work and of the machine vision field in general is available in [116]. The approach was to first determine a computational theory that specifies what needs to be computed and only then to devise an algorithm and hardware to perform those computations. Insights leading to new theories come primarily from the physics of inverse optics and physiological studies of early vision. It was found that computations often fit into a mathematical class of problems known as ill-posed problems which can be solved by mathematical techniques from an area called regularization theory [207]. A mathematical approach using Markov random fields can be used to solve such problems. This approach leads to algorithms that can be implemented using neural networks made of analog resistance arrays and switches [207,140,139]. Such analog preprocessors are currently under development at a number of laboratories. The Markov random field approach provides a useful framework for extracting specific features of regions and also for integrating multiple features [74]. Many of the vision algorithms, including those using Markov random fields, have been programmed to provide rapid throughput using a large parallel computer called the Connection Machine. Many algorithms have also been tested with real images.

The original presentation of this computational theory of vision separated the computational theory from the algorithm and the hardware implementation. Experience over the last 15 years with this approach, however, has demonstrated that these three components of vision research are intrinsically intertwined and cannot be treated as if they were independent [116]. Recent work stemming from this approach has begun to explore techniques to learn early vision algorithms. For example, it was recently demonstrated that an algorithm to extract intrinsic color can be learned from examples [115].

8.4.8 Self-organization in Early Vision

Linsker [155] has investigated the self-organization of cells with opponent (“on-center/off-surround”) and orientationally selective receptive fields. In his model, spontaneous cell activities and some simple rules for synaptic modification (Hebbian learning) result in spatial, or map-like, organization of cells. The receptive fields of these cells mimic those of biological neurons in early vision areas.

8.4.9 Silicon Retina

Mead and his colleagues [184,185] have developed an analog VLSI vision preprocessing chip modeled after the retina. The design not only replicates many of the important functions of the first stages of retinal processing, but it does so by replicating in a detailed way both the structure and dynamics of the constituent biological units. Thus the logarithmic compression in photon input to output signal characteristic of biological rods is accomplished not by digital processors computing logarithms but by analog circuits that result in a logarithmic transfer function. Similarly space and time averaging and temporal differentiation are accomplished by analog processes and a resistive network. Mead was able to synthesize these processes in part because they are so elementary and so observable in biology. While not all of the functions of biological vision have been as thoroughly analyzed as those of the retina, Mead’s work is a paradigm of what can be accomplished when biological discovery drives technological implementation.

8.4.10 Binocular Disparity

Schwartz and Yeshurun [230,282] have developed an algorithm to extract disparity or depth information from binocular data that was motivated by the existence of binocular disparity columns early vision areas of the brain. This algorithm is based on a cepstral analysis that could be performed using neural-like circuits if access were provided to information in nearby binocular disparity columns. They also emphasize the role of computational maps in the visual cortex and note that this map approach may lead to greater understanding of visual processing.

8.4.11 A Supercomputer Model

Travis, Gremillion, and Mandell [84] have implemented an extensive model of early visual processes on a supercomputer. Their work is noteworthy for its attempt to simultaneously describe several biologically-based stages (retina, lateral geniculate nucleus, primary visual cortex) and at the same time to model

specific physiological classes of cells in detail. They have begun to investigate the role of feedback and temporal delays, as well as spatial filtering, in processing signals at various stages.

8.4.12 Perceptually Motivated Features

Walters [268,269] has developed image segmentation and object recognition algorithms based on the use of perceptually motivated features. She has also emphasized the importance of data representation in neural network vision models. The results of psychophysical experiments suggest the existence of a hierarchy of visual features based on the relations between image contours. The human visual system appears to be pre-attentively, selectively sensitive to image contours which contain certain of these features. The results have been used to create computer vision algorithms for the segmentation of boundary images into sets which have a high probability of depicting a single object. A related network architecture, called *rho-space*, has been developed for the detection and representation of oriented edges. The input to the network is the output of oriented edge operators. Computations within the network are based on orientation-dependent, three-dimensional, excitatory and inhibitory neighborhoods. Among the networks capabilities are: natural representations of connectivity, filling in of incomplete or illusory contours, simultaneous coarse and fine representations or orientation information, and absence of reliance on domain-dependent knowledge or model-based processing.

9. SPEECH RECOGNITION

9.1 INTRODUCTION

Speech is the most natural form of communication. Compact implementations of accurate, realtime speech recognizers would find widespread use in many applications including:

- Automatic transcription,
- Simplified man-machine communication,
- Word-spotting,
- Aids for the deaf with realtime text or tactile output,
- Aids for the physically disabled which respond to voice commands,
- Sonar pattern recognition,
- Acoustic surveillance,
- Auditory input for automata,
- Automatic language translation,
- High-fidelity, low-bit-rate speech storage and communication, and
- Secure voice-keyed locks.

Compact recognizers could be used to simplify the complex arrays of switches and controls used in aircraft cockpits, to interrogate databases in command and control centers, to replace relatively large keyboards in compact equipment, and to automate data entry in many applications where data is currently entered manually.

Unfortunately, current speech recognizers perform poorly on talker-independent continuous speech recognition tasks that people perform without apparent difficulty. Although children learn to understand speech with little explicit supervision and adults take speech recognition ability for granted, it has proved to be a difficult task to duplicate with machines. This is due to variability and overlap of information in the acoustic signal, to the need for high computation rates (a human-like system must match inputs to 50,000 words in real time), and to the multiplicity of analyses that must be performed (phonetic, phonemic, syntactic, and pragmatic).

The best existing speech recognizers perform well only in highly constrained tasks. For example, talker-dependent isolated-word recognizers can be trained to recognize 105 words spoken without sentence context with 99% accuracy [196]. Talker-dependent word recognition accuracy with sentence context can be as high as 95% for 20,000 words from sentences in office memos spoken with pauses between words [16]. This translates to a sentence accuracy of only 50%. Performance is generally much worse for connected

speech, when training isn't provided for each talker, and in noisy and stressful environments. The best current algorithms use hidden Markov models (HMM) techniques [211] and require high computation rates for large vocabulary tasks. These rates are difficult to obtain with existing single-processor computers.

The *hidden Markov model (HMM)* approach to speech recognition is currently the most general framework for speech recognition research. It is based on information and communication theory results and provides automatic training and efficient temporal alignment and matching. Recent progress, however, has slowed due to a number of problems with this approach, including:

- Internal model structure is not learned but must be pre-specified before training. Model builders must specify sub-word models, provide phonemic transcriptions of words, and develop rules to describe allowable types of phonological variation caused by word boundary effects, and changes in talking style and dialect.
- Coarticulation is modeled poorly. Coarticulation caused by nasals or laterals may extend over many acoustic segments. This is not easily captured with a single-order Markov assumption.
- Poor acoustic-phonetic modeling results in confusions between similar sounding words.
- Supervised labeled training data is required to train high-level allophone and word models. The abundance of unsupervised speech data that is readily available thus cannot be used.

Recent work on neural networks raises the possibility of new approaches to the speech recognition problem. Neural networks offer two potential advantages over existing approaches. First, they could provide the high computation rates required for continuous speech recognition using many simple processing elements operating in parallel. Second, new neural network algorithms could perform better than existing algorithms if they could both adapt internal parameters over time to maximize performance and also self-organize by altering network structure to capture new phenomena as they are observed.

Work to date with neural networks has focused on talker-dependent isolated word recognition. A block diagram of a simple isolated word recognizer is shown in Fig 9-1. Four major operations are required:

1. A preprocessor must extract important information or parameters from the speech waveform. In most recognizers, an input pattern containing spectral information is extracted every 10 milliseconds using fast Fourier transform (FFT) or linear predictive coding (LPC) techniques.
2. Input patterns from the preprocessor must be compared to stored exemplar patterns in word models to compute local distances.
3. Local distance measures must be used to time-align input patterns to classification nodes in the word models and arrive at whole-word matching scores. Time alignment is necessary to compensate for variations in talking rate and pronunciation.

4. The word with the best score must be selected by picking the word model with the maximum output.

Neural networks have been proposed to perform all of the above operations.

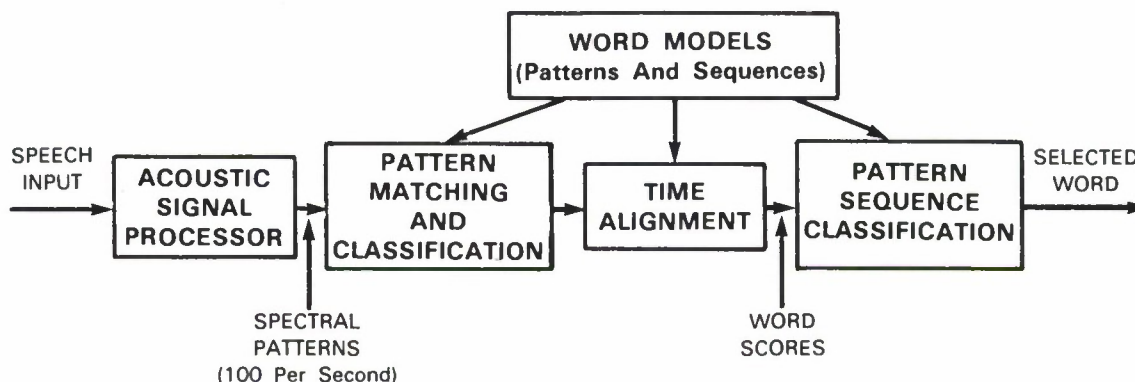


Figure 9-1. Block diagram of an Isolated Word Recognizer.

9.2 PHYSIOLOGICAL PREPROCESSORS

A number of researchers have proposed new forms of preprocessing motivated either by the physiology of the cochlea or by psychoacoustic results [34,50,76,166,176,183,191,236,238]. Two preprocessors [176, 191] use analog processing, including a filter-bank, to extract a small set of perceptually important binary-valued features in real time. One of these preprocessors [176] was used in a commercial realtime isolated word recognizer.

Two recent preprocessors [34,166] rely primarily on spectral magnitude information processed in a way to model psychoacoustic and physiological results in loudness, masking, and adaptation. Such processing can provide a small performance improvement over more conventional approaches [34].

Other recent preprocessors [50,76,183,236,238] use synchrony information similar to that which is available on the auditory nerve where timing information of nerve spikes is available. This phase information could increase recognition performance by supplementing the spectral envelope magnitude information used in current recognizers. Synchrony information is typically obtained by filtering the input using sharp filters with characteristics similar to those of the mechanical filters in the cochlea and then analyzing the phase information of the resulting filtered waveforms. Spectrograms obtained with steady-state vowels and speech sounds illustrate an improvement in ability to visually identify vowel formants (resonant frequencies of the vocal tract) in noise with this technique [50,76,236,238]. In addition, detailed comparisons to a standard front end with an existing speech recognizer reported in [76] demonstrated significant performance improvements in noise with the physiological front end. One preprocessor [183] is modeled after the cochlea and is being implemented in VLSI using analog processing. Both synchrony and spectral magnitude information could be extracted from this device. This work illustrates how pre-processing could be miniaturized and integrated with higher levels of processing.

9.3 CALCULATING DISTANCE SCORES TO EXEMPLAR PATTERNS

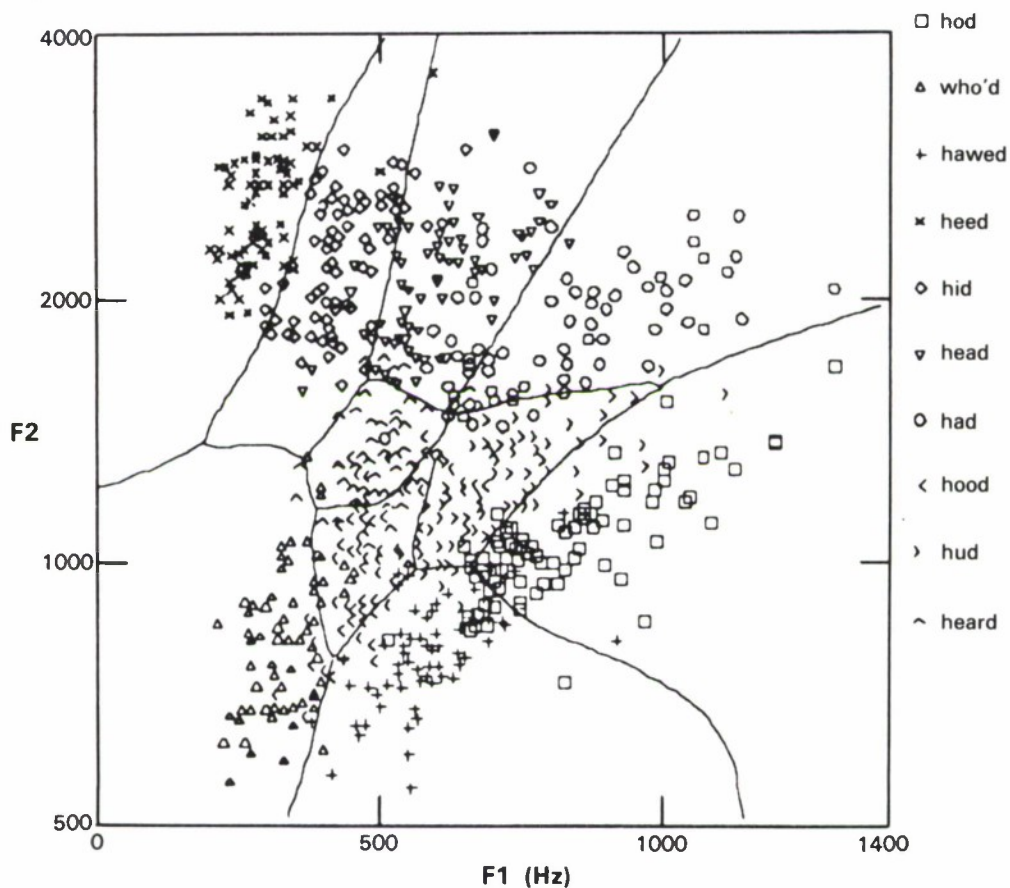
Conventional speech recognizers compare input patterns (vectors of parameters) to stored exemplar patterns whenever a new pattern is provided by the preprocessor. Neural networks could perform this function for almost all conventional recognizers using fine-grain parallelism because, as described above, multi-layer perceptrons can calculate any likelihood function.

Conventional recognizers frequently calculate the Euclidean distance between the input and all exemplar patterns. This can be performed in parallel using a neural network with an architecture similar to that of the lower half of the feature map classifier presented in Figure 6-3. The connection weights in this network then have to be set to represent the centroid locations of the exemplar patterns. Continuous-observation recognizers require all outputs of second layer nodes without selecting that node with a maximum value while discrete observation recognizers use node outputs after the node with a maximum value is selected. Both approaches have been used in experimental isolated word recognizers [27,145,158]. Recognizers that don't use Euclidean distance measures could use either a multi-layer perceptron with two layers of hidden units or a structure similar to that of a feature map classifier to obtain the desired likelihood functions.

An example from [114] presented in Figure 9-2 illustrates how neural network classifiers can be used to compare fixed input patterns to reference patterns for speech data. Input data consisted of the first two vowel formants from roughly 330 tokens of 10 different vowels spoken in /hVd/ context by 67 men, women, and children. These data were obtained from scatter plots in [203]. Half of the data was used for training to form reference patterns for the 10 vowel classes and half was used for testing. The decision regions shown in this Figure were formed by a two-layer perceptron trained with back propagation. Boundaries for these regions are near those that are typically drawn by hand to separate vowel regions. Four recognizers which can be implemented with neural network architectures were tested (k-nearest neighbor, Gaussian, two-layer perceptron, and feature map classifier) on this data. All provided an error rate of roughly 20%. The two-layer perceptron, however, required more than 50,000 supervised training trials for convergence with backpropagation. The feature map classifier required fewer than 50 supervised training trials.

9.4 CLASSIFYING WORDS AND PHONEMES WITHOUT TIME ALIGNMENT

A number of studies have used multi-layer perceptrons and other neural network classifiers to classify isolated words (primarily digits), phonemes, and vowels using pre-segmented speech tokens [27,29,30,60,81,158,199,210,267,272]. Most of these studies used multi-layer perceptrons with inputs derived from spectral analyses of a pre-segmented section of speech waveform. All inputs were typically applied at once as one whole spectrographic input pattern. In some cases, multi-layer perceptrons were used in conjunction with conventional time alignment techniques [27,30] to replace conventional classifiers or vector quantizers and, in one case, recurrent connections were provided [272]. Although many of these studies provided encouraging results, few careful comparisons to conventional classifiers were performed. In those cases where comparisons were performed, neural network classifiers typically did not provide



	% ERR	CONVERGED
K-NEAREST NEIGHBOR	18.0	n = 338
GAUSSIAN	20.4	n = 338
TWO-LAYER PERCEPTRON	19.8	n = 50,000
FEATURE MAP CLASSIFIER	22.8	n < 50

Figure 9-2. Decision Regions Formed by a Two-layer Perceptron Using Backpropagation Training and Vowel Formant Data.

100205-3

substantial performance improvements over more conventional techniques such as Gaussian or k-nearest neighbor classifiers [27,30,158,199].

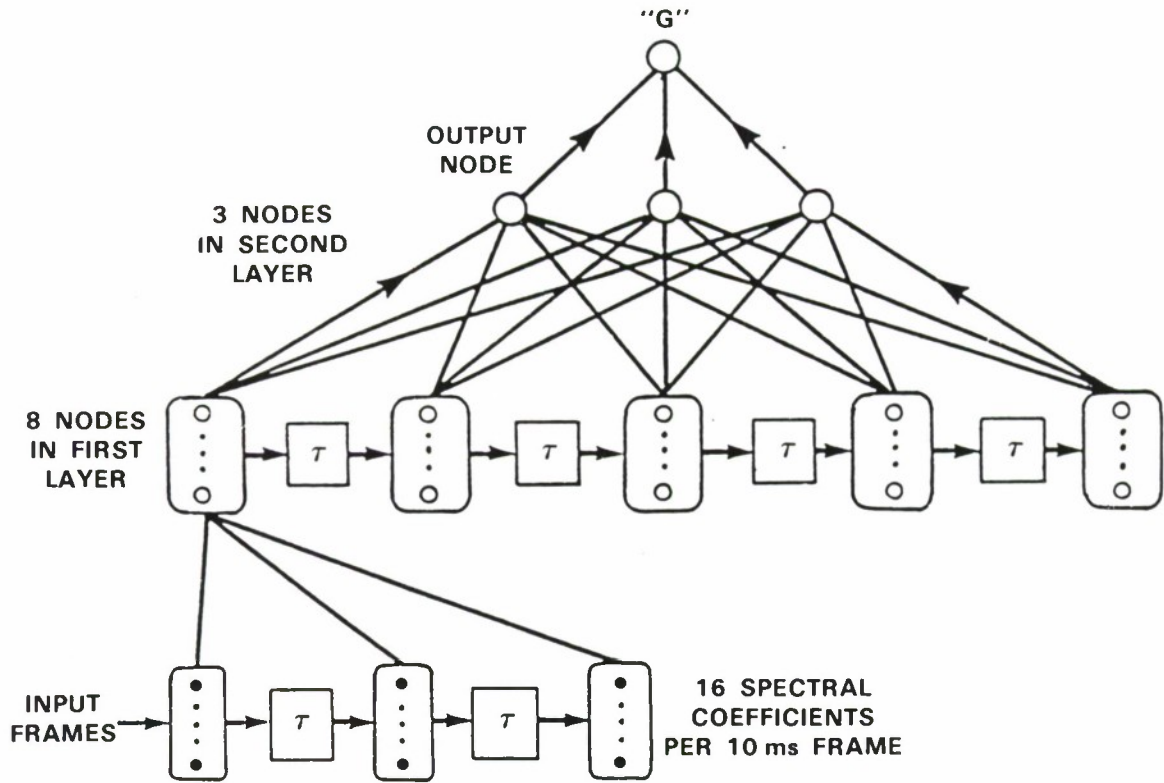


Figure 9-3. Time delay Neural Network Used to Classify the Voiced Stops "B,D,G".

Encouraging results were recently obtained by Waibel [267]. A multi-layer perceptron with time delays shown in Figure 9-3 was used to classify the voiced stops "b,d,g." The boxes labeled τ in this Figure represent fixed delays. Speech frames characterized by spectral coefficients are input on the lower left. The three boxes on the bottom thus represent an input buffer containing a context of three frames. Outputs of the nodes in these boxes (16*3 spectral coefficients) feed to eight hidden nodes in the first layer. Outputs from these nodes are buffered across the five boxes in the first hidden layer to form a context of five frames. Outputs from these boxes (8*5 node outputs) feed to three hidden nodes in the second hidden layer. Outputs from these three nodes are integrated in a final output node.

The time-delay network was trained using backpropagation and compared to a discrete-observation HMM recognizer that used 256 symbols and three-node phoneme models. This recognizer was trained using the forward-backward algorithm. Both systems were trained with roughly 2,000 voiced stops and tested with a different set of 2,000 voiced stops. Stops were excised manually from a corpus of 5,260 Japanese words spoken by three male talkers. The excised portion contained 15 frames (150 milliseconds) centered around vowel onset. The neural network classifier performed much better than the HMM recognizer. It provided an error rate of only 1.5% compared to the HMM error rate of 6.5%.

Boulard [27] constructed a similar neural network allophone classifier which had 14 delays in the input layer to provide a context of 15 frames. No delays were used in the hidden layer. Input nodes feed to 50 hidden units which feed to 26 output nodes (one per each of 26 allophones). This network was trained using backpropagation and 20 hand-segmented digits spoken by one male talker. Tests on 20 other digits resulted in no errors when node outputs were used as input distances for a dynamic time-warping recognizer.

9.5 NEURAL NETWORKS FOR TIME ALIGNMENT

Sequences of input observations from a preprocessor must be time-aligned with classifier nodes to provide good recognition performance by compensating for variations in talking rate and pronunciation. Conventional engineering approaches to this problem are to use the Viterbi algorithm as used in HMM recognizers or to use dynamic programming as used in dynamic time warping (DTW) recognizers. One obvious neural network approach is to attempt to implement a Viterbi or DTW decoder using a neural network architecture.

9.5.1 Implementing Viterbi Decoding Using Neural Networks

A neural network architecture that implements a Viterbi decoder for an HMM continuous-observation recognizer is shown in Figure 9-4; it is called a *Viterbi network*.

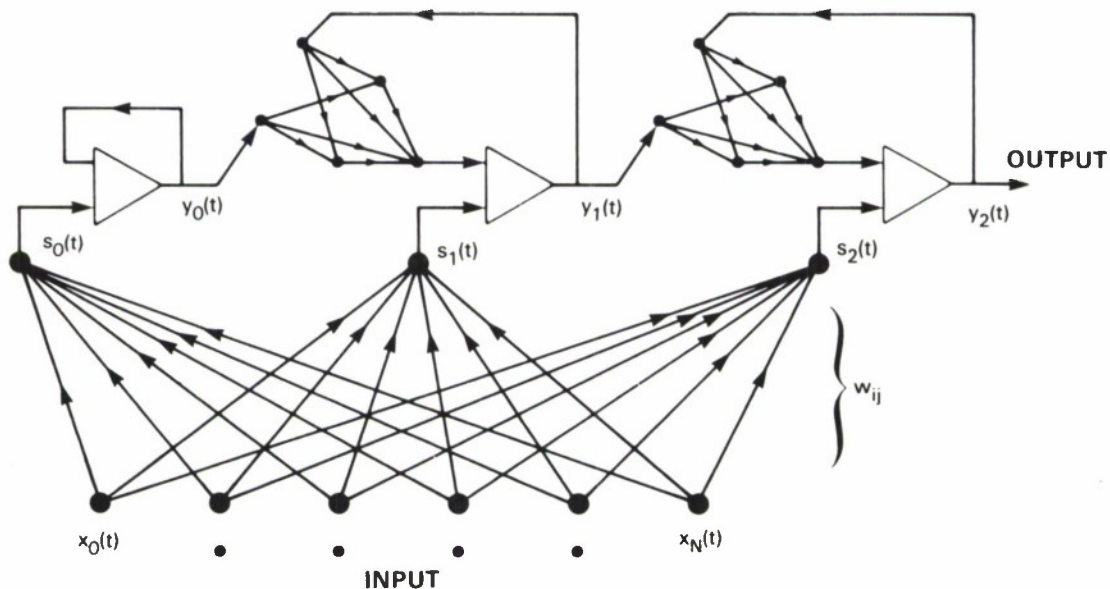


Figure 9-4. Neural Network Called a Viterbi Network that Implements a Viterbi Decoder as Used in HMM Speech Recognizers.

Classifier nodes in this network, represented by large open triangles, correspond to a nodes in a left-to-right HMM word model. Each classifier node contains a threshold logic node followed by a fixed delay. Nodes positioned above classifier nodes are threshold logic nodes and nodes below classifier nodes simply output the sum of all inputs. A temporal sequence of input vectors is presented at the bottom of the network and the matching score gradually builds up over time and is produced at the output. Subnetworks above the classifier nodes select the maximum of two inputs and subnetworks at the bottom compute the log probabilities required by a Gaussian classifier.

Studies performed using more than 4,000 word tokens from a large speech data base were used to evaluate the performance of this network. This database was made from tokens of 35 difficult words spoken by nine talkers. Weights in Viterbi networks with 15 classifier nodes were adjusted based on means, variances, and transition probabilities obtained from the forward-backward training algorithm using five training tokens per word. Inputs consisted of 12 mel cepstra and 13 differential mel cepstra that were updated every 10 milliseconds. Performance was good and almost identical to that of current HMM isolated word recognizers. The error rate was 0.56%, or only 23 out of 4,095 tokens wrong.

9.5.2 Psychological Models of Speech Recognition

A number of new neural network models which are primarily psychological models of speech perception have been proposed [59,167,175,224]. One model, called the *cohort model* [175], assumes a left-to-right acoustic phonetic analysis of speech in real time. It accounts for many psychophysical results in speech recognition, such as the existence of a time when a word becomes unambiguously recognized (recognition point), the word frequency effect, and recognition of contextually inappropriate words. This model, however, is descriptive and is not expressed as a computational model.

A second psychological model which is more mathematical and has been simulated using speech input is called the *trace* or *interactive activation model* [59,224]. This model is based on neuron-like nodes and emphasizes the benefits that can be obtained by using coarticulation information to aid in word recognition. The current form of this model is impractical because the problem of time alignment is not addressed and the entire network must be copied on every new time step.

The third psychological model is called *node structure theory* [167]. It is a qualitative neural theory of speech recognition and speech production. It is similar in many ways to the cohort and interactive activation models but much more comprehensive. The problem of rate and sequencing is considered seriously and problems such as stuttering, internal speech, and rhythm are considered. This theory, however, is again not expressed in terms of a formal mathematical model.

9.5.3 Computational Models of Time Alignment

A number of neural network models have been proposed that perform some component of the time alignment task [35,37,48,81,101,257,281]. Two of these models have been tested with speech input [48,81,257,281]. One of these two [257] assumes that variable length delays are available and that delays cause temporal dispersion to pulse inputs that increases with delay. Different delays are attached to feature detectors to form a matched filter that concentrates energy in time. A summing node for each word

produces a large output after that word is presented at the input. This technique provided reasonable digit accuracy for a few talkers [257] with limited testing. Detailed tests with a large speech database [80] using a hierarchical version of the model with both allophone and word models yielded performance no better than that of an existing HMM recognizer.

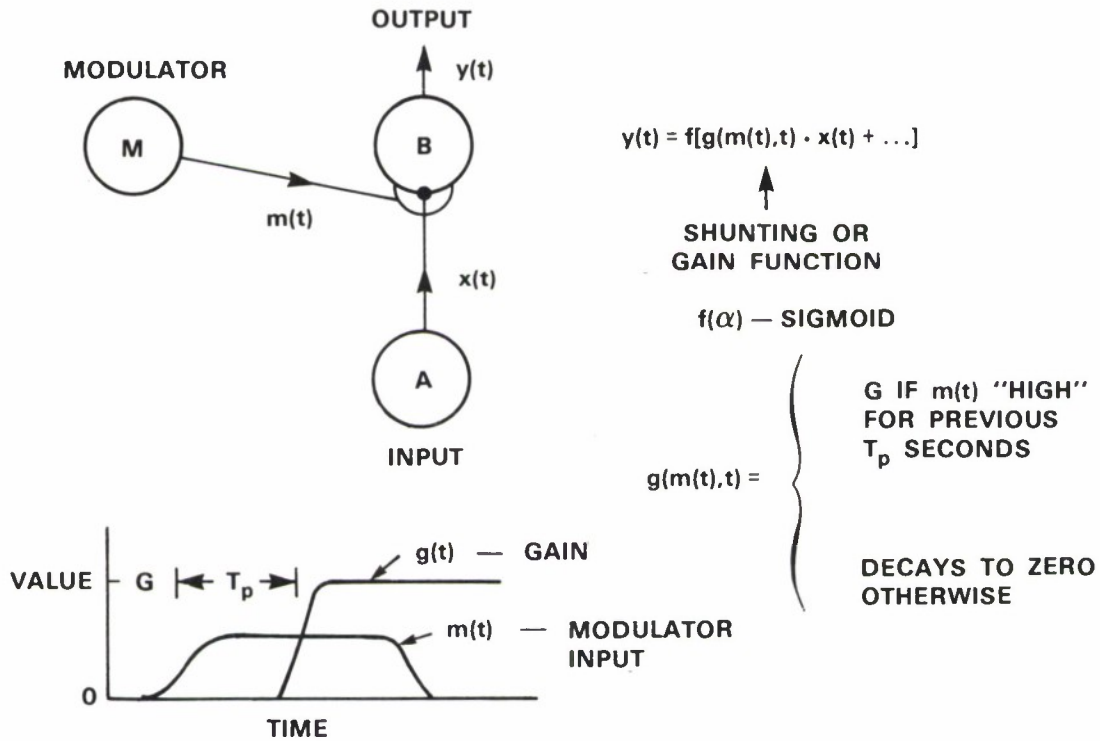


Figure 9-5. A Synaptic Triad that can be Used to Recognize Two-component Pattern Sequences.

Two closely related models which have been tested with a small amount of speech data are described in [48,281]. Both models include neurons with shunting or multiplicative nodes similar to those that have been proposed in the retina to compute direction of motion [206]. Three neurons are grouped to form a *synaptic triad* that can be used to recognize two component pattern sequences. Figure 9-5 shows one synaptic triad. Node A is the primary input to the triad and node B is the primary output. Node M is the modulator input. The signal from the modulator input $m(t)$ controls the instantaneous connection weight or gain $g(\alpha, t)$ from A to B. This gain modulates the input to node B. It decays to zero unless the modulator input is "high." After the modulator input has been "high" for T_p (time for potentiation) seconds, a slow internal integration builds up and switches the gain to a "high" value G . This is illustrated in the plot in Figure 9-5 where the modulator input first goes high and the gain rises to G after T_p seconds. The triad will have a "high" output only if the input sequence is M-A (output of node M is "high" followed by output of node A) and node M is "high" for at least T_p seconds. Other pattern sequences (A-A, M-M, A-M) will produce no output.

Synaptic triads can be arranged in sequences and in hierarchies to recognize features, allophones, and words [281]. In limited tests, handcrafted networks could recognize a small set of words spoken by one talker [281]. More interesting is a proposed technique for training such networks without supervision [48]. If effective, this training could make use of the large amount of unlabeled speech data that is available and lead to automatic creation of sub-word models. Further elaboration is necessary to describe how networks with synaptic triads could be trained and used in a recognizer.

Three related neural models have been described but never tested with speech input [35,37,101]. One model [101] is described as a *nearest matched-filter classifier for temporal patterns*. This network is similar in structure to a feature map network except nodes have different rise and fall time constants and nodes are interconnected such that only a few nodes are active at any time. No details are provided concerning how this network should be trained or how node outputs should be integrated to detect allophones or words.

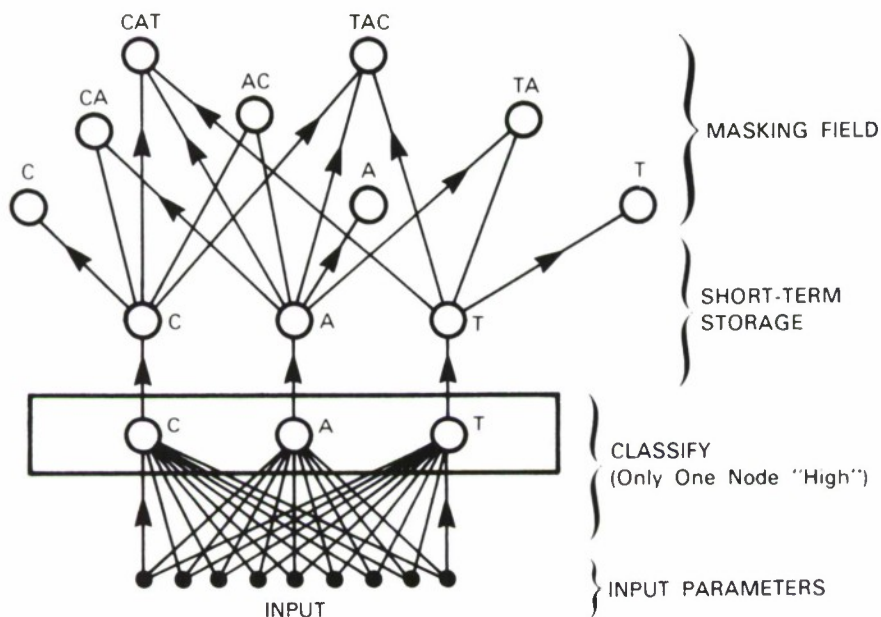


Figure 9-6. A Model Called a Masking Field can be Used to Detect Pattern Sequences.

A model described in [35], called a *masking field*, is shown in Figure 9-6. Input is applied to the bottom of this network which is, again, similar to a feature map network. Typically, only one node in this network has a "high" output at any time. These node outputs feed other nodes that provide short-term storage. The outputs of short-term storage nodes decay over time. Different input sequences thus lead to different amplitude patterns in short-term storage. For example, the input C-A-T sampled at the end of the word will yield an intensity pattern in short-term storage with node C low, node A intermediate, and node T high. The input T-A-C will yield a pattern with node C high, node A intermediate, and node T low. These intensity patterns are weighted and fed to nodes in a masking field with weights adjusted to detect different patterns. The masking field is designed such that all nodes compete to be active and nodes representing longer patterns inhibit nodes representing shorter patterns. It has difficulty recognizing patterns with

repeated sub-sequences because nodes in short-term storage corresponding to those sub-sequences could become saturated. Further elaboration is necessary to describe how masking fields should be integrated into a full recognizer.

A related speech recognition model is currently under development [37]. It uses many of the neural network components described in [90] to form a complex multi-module system. One interesting characteristic of this model is that speech production and recognition will be integrated and used to develop sub-word models and stabilize those models.

A final approach to pattern sequence recognition is to build an associative memory for pattern sequences as described in [131,132]. Here, a neural network with recurrent connections and delays is designed that spontaneously produces pattern sequences. When an external pattern sequence is applied, the internal sequence can phase lock to the external sequence and potentially fill in missing components.

10. ROBOTICS

10.1 INTRODUCTION

Current industrial robots are extremely limited in their capabilities. The neural network approach has the potential for adding the ability to learn and adapt, and for integrating information from multiple inputs. Neural networks are beginning to find practical applications in the field of robotics. Most robotic problems currently being attacked fall into the following four general classes:

- Programming and controlling the trajectories of multi-axis robotic arms,
- Robot navigation,
- Arm-camera (i.e., hand-eye) coordination, and
- Visual/tactile fusion for object recognition.

These problems have been addressed using a number of neural network models, including: the CMAC network, topographic sensory/motor maps, multi-layered perceptrons, simulated annealing, and Hopfield networks. In some cases, the neural network is a novel way of implementing an existing idea in robotics, while in other cases it is the heart of a new approach to robotics. The status of these projects ranges from being not yet debugged at the simulation level, to fully debugged laboratory mechanical models, to almost industrial applications. The research is described below in more detail with related work grouped together.

10.2 TRAJECTORY CONTROL

The problem of trajectory control is to design a control system to generate the desired trajectories of a robotic arm during a relatively fixed and routine task. This is a typical application in robotic assembly in fixed environments.

10.2.1 CMAC Models

One of the major inspirations in neural robotics has come from models of the cerebellum. There have been extensive experimental studies of the cerebellum in the last several decades. These studies have lead to the conclusion that the cerebellum is essential for the adaptation, learning, and execution of both reflexive and voluntary motor actions. Researchers have been afforded a rich source of experimental data to inspire and constrain their models.

The earliest formulations of cerebellar models were made by David Marr [171] and James Albus [5]. Their models are essentially the same except in how the neural weights are changed. Albus calls his model the Cerebellar Model Articulation Controller (CMAC). The CMAC is essentially a clever adaptive table look-up technique for representing complex, nonlinear functions over multi-dimensional, discrete input

spaces. It reduces the size of the look-up table through hash-coding, provides for response generalization and interpolation through a distributed, topographic representation of the inputs, and learns the appropriate nonlinear function through a supervised learning process that adjusts the content or weight of each address in the look-up table. A block diagram of the CMAC model is presented in Figure 10-1. Inputs in this Figure project through a hash coding stage onto table A , and then to output summers.

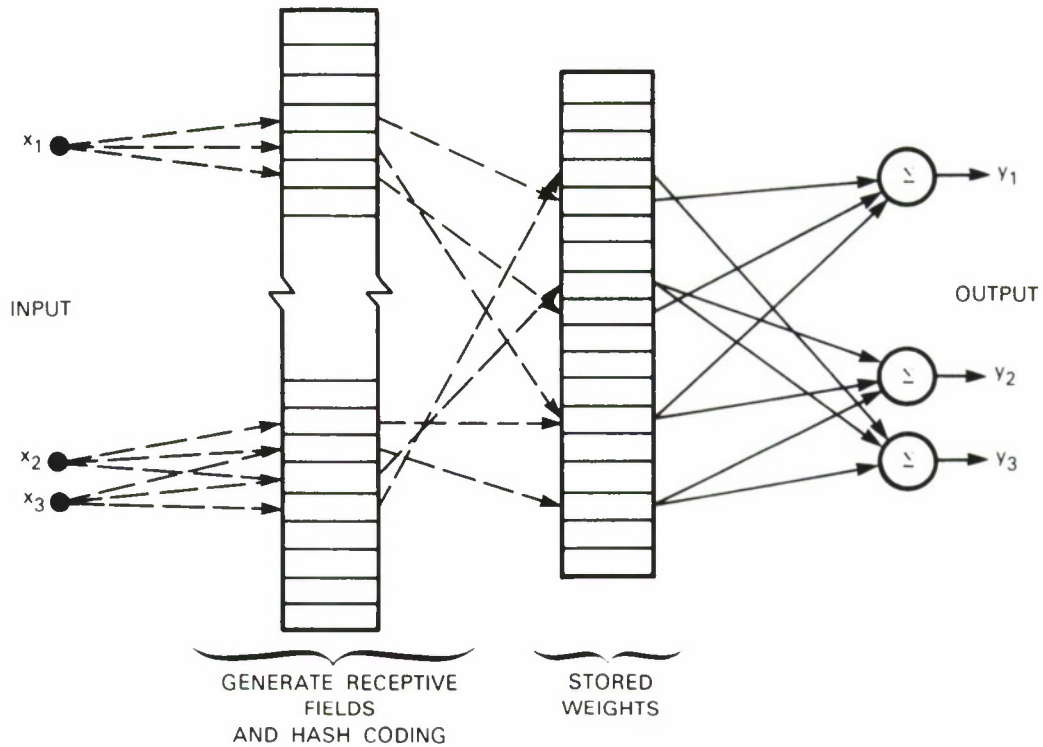


Figure 10-1. The CMAC Cerebellum Model Forms Nonlinear Input/Output Mappings Useful for Robotic Control.

Albus's original application of the CMAC [5] was in a novel control scheme. The CMAC was placed in a closed control loop where the inputs were high-level control commands along with the current values of the robot's joint variables, and the outputs were the drive signals to the robotic actuators. The task was to produce a desired trajectory through simple iteration of the learned nonlinear mapping, given the correct command state and given an initial configuration that was on the desired trajectory. Training procedures were proposed to achieve this goal but were not suitable for industrial application. Albus also explored the concept of hierarchical control within this novel CMAC system.

Jordan [119] proposed a recurrent model that is similar in structure to the recurrent network Albus originally developed. It produces one or more different desired output sequences. A multi-layer network has both state inputs which feed back the output state and plan inputs which designate the desired output sequence. An error-corrective training algorithm is provided for this network which allows "don't care"

outputs. Experiments demonstrate that this network can exhibit some of the effects of speech production including coarticulation.

One research group is applying the CMAC in a standard controller scheme in a very novel and clever fashion [187,118]. Here, the CMAC is used to learn the inverse kinematics of the robot, computing the correct torques for the next control cycle given the desired robot coordinates computed by a standard trajectory planner. The learning is done on-line, with the CMAC operating in parallel with a fixed-gain, linear feedback controller. Initially, the linear controller is doing all of the control, while the CMAC “looks over its shoulder” and learns the inverse kinematics from the approximate trajectories produced. As the CMAC learns, it corrects the errors of the linear controller, refining the trajectory, and thus its own learning data, in a positive feedback loop. Eventually, the CMAC provides all of the drive signals for the robotic actuators, since there is no longer any error for the linear controller to minimize. This system has been applied to the control of a five-degree of freedom robot arm, and is close to being used in an industrial application. Kawato’s group [125,126] has implemented a similar model on a six-degree-of-freedom robot, also controlling trajectories with nonlinear dynamics. Their approach is also aimed at an industrial application.

10.2.2 Backpropagation Models

As noted above, multi-layer perceptrons trained with backpropagation can learn arbitrary nonlinear input/output mappings from examples. These mappings can be used to model inverse kinematics, as was done by Miller with the CMAC model, or they can be used to learn nonlinear coordinate transformations required for controlling robot arms, as in [244]. Speed of convergence can be improved by initializing weights to force the network to produce desired input/output values at specific points in the input/output space. Such an initialization would use fixed weights in the first two layers to form bins or hypercubes, as in the CMAC model, or the hypercube classifier, described in [114]. Backpropagation can then be used to adjust this mapping adaptively.

10.3 ARM/CAMERA CONTROL

There is growing interest in applying robotics in applications where the environment of the robot can change in unforeseen ways. For example, various obstacles (people) may enter or leave the working area of the robot and need to be avoided, or a novel object may need to be grasped. One general approach to this type of problem is to provide a visual sense of the objects in the environment to the robotic controller, which then must compute an appropriate, perhaps entirely novel, trajectory.

10.3.1 Sensory/Motor Maps

Two research groups are applying sensory/motor topographic maps to this problem. An alternative cerebellar model has been developed by Grossberg and Kuperstein [91]. The learning algorithm and distributed representation in their model is similar to the CMAC model, but there are also some key differences. Their model focuses on the benefits of constraining all internal representations in the model to be

topographic. Topography is an ordered contiguous mapping of a surface. Topographic representations are based solely on the set of sensory transducers and motor effectors. By coupling topographic representations with reflexive behavior, Grossberg and Kuperstein show how a controller can learn motor parameters through self-supervised learning. In such learning there is no need to define the desired parameters external to the model. Moreover, this model is more consistent with experimental data on lesions, behavior, physiology, and anatomy of cerebellar circuits than the CMAC model. Grossberg and Kuperstein show simple computer simulations that illustrate self-supervised learning to control eye movements.

More recently, Kuperstein [149,148] has developed a new neural architecture and self-supervised learning algorithm to control adaptive sensory-motor coordination. This is based more on a neocortical model and is called *Infant (Interactive Networks Functioning on Adaptive Neural Topographies)*. Learning occurs via a circular reaction, an idea borrowed from Piaget. The circular reaction has two stages. In the first stage, random self-produced motor signals are used to generate the entire range of object manipulations one at a time. During each posture, with object in hand, visual input signals about the object are processed and combined into a target map through modifiable weights, producing computed motor signals. Errors in the target representation are determined by the differences between the actual motor signals and the motor signals computed from the visual input. These errors are used to incrementally change the weights, so that on future trials, the computed motor signals are closer to the actual motor signals. These changes, for all important postures, constitute the sensory-motor correlation and force visual and internal motor control signals to become self-consistent.

In the second stage of the circular reaction, learned sensory-motor correlation is used to recognize and manipulate objects which are similar to those that were experienced in the first stage. In this stage, an object first comes within view and reach, free in space. The eye foveates on the object and motor commands generated by eye muscles and visual input drive actuators to reach for the object. These signals are different for specific orientations of any one object and for different objects. The architecture used in the Infant model is composed of motor map representations interleaved within a sensory topography. This allows any number of topographic sensory inputs to be mapped onto any number of motor outputs. A simple computer simulation of this neural network for controlling a simulated robot arm with five degrees of freedom demonstrated how a cylinder could be grasped using binocular information from two eyes. The simulations show that the network can touch a cylinder to an average accuracy of 4% of the arm's length. Other recent work is examining the dynamics of planned arm movements using nonlinear neural networks [28].

10.3.2 Darwin III

Darwin III [215,57] is the one of the most complex neural network simulations yet developed. A block diagram of this network is presented in Figure 10-2. It is a simulated automaton made up of many subnetworks. Inputs to the automaton are from a simulated eye which scans a two-dimensional input array under control of opponent pair muscles. This eye has a large but low-resolution outer visual field and a smaller, high-resolution inner visual field on its simulated retina. Inputs are also provided by a multiple-jointed arm that can reach and "feel" objects presented on the input field. Darwin III is first trained to track objects presented on the input array by coordinated movement of eye muscles. The error signal to learn this task is the distance between the position of the object on the retina and the center or fovea of the retina. Darwin

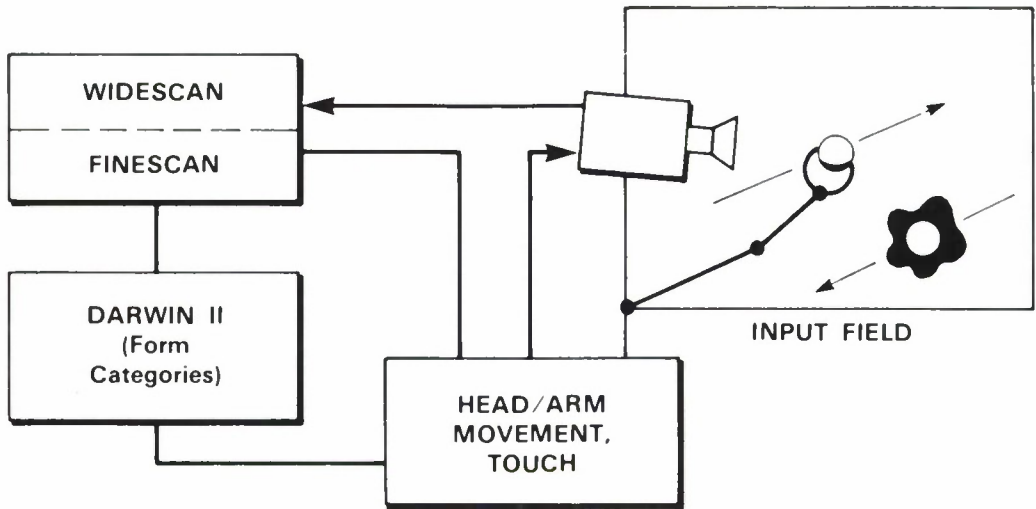


Figure 10-2. A Block Diagram of the Darwin III Simulated Automaton.

III is then trained to reach out, touch, and feel around the border of objects using self-supervision training techniques. Finally, it is trained to classify objects using supervision and both visual and touch inputs.

Darwin III is the most complex in the series of models created to explore a brain theory being developed called “neural Darwinism” [57]. This theory differs from other neural network design procedures in its emphasis on the importance of Darwinian-like selection to select those neuronal groups or subnetworks which respond best to specific input stimuli. The theory assumes that a collection of neuronal groups, called a *repertoire*, is formed during embryogenesis. Groups in a repertoire respond best to overlapping but similar input patterns due the randomness of neuronal growth. One or more groups in a repertoire will respond to every input pattern, and response to important unexpected inputs is thus insured. Training involves competition between groups, which amplifies the responses of specific groups to specific stimuli and associates those groups with each other and with a specific appropriate response.

Neural Darwinism is very different from the common approach of designing a network topology and training it with supervision to provide a desired response. Instead, it assumes that there are, by design, many subnetworks. Only those with the desired response during training are selected. Important issues addressed by this approach include the need to respond to unexpected stimuli, the necessity of classification and generalization, and the importance of interacting with the environment. The building of an automaton with well-defined tasks, senses, and the ability to manipulate external objects may also be an approach that is essential for building adaptive robots, as suggested by [54]. In addition, the experience with simulations of large multi-module systems such as Darwin III, which contains roughly 8,000 nodes with roughly 200,000 interconnections, will probably be essential in building more capable models.

Potential Energy Network

Hogan [106] has applied a *potential energy network* to the problem of reaching to and tracking an object in two-dimensional motion amidst arbitrarily placed objects with a multi-jointed arm that lies in the plane of the obstacles. The motion of the arm is severely constrained by the obstacles and yet very rapidly (it easily keeps up with rapid man-made movements of the object) reaches around them towards the object. The position of the obstacles and the object is observed by an overhead camera. A computer then constructs a potential energy surface with peaks at the obstacles and valleys in between leading to the energy minimum at the object. A gradient descent algorithm is used to calculate the trajectory of the arm from the energy surface. Development of this system has reached the stage of a working laboratory-grade demonstration.

10.3.3 Spatial Reasoning for Robotic Navigation

Jorgensen [120] used neural networks in two stages of a four-step navigation process. First, a Hopfield-type network was used to record a map of the positions of objects in the room. Subsequent sensor readings from any part of the room could then be used as a cue to the network to retrieve the full room map. Second, a simulated annealing network was used for planning a path between the objects to the goal. The associative network worked well in this task, but the path planning network was too slow. This work was carried out using real robots in a laboratory setting.

10.3.4 Linear Control

In the area of linear coordinate transformations, Pellionisz [200] has advocated the use of tensors to transform points between the separate coordinate systems used by sensory and motor systems. This approach is useful when transformations are linear, data representations are linear, and coordinate systems use nonorthogonal axes. It cannot be used easily in cases when nonlinearities are present. In those cases, the map-based approaches described above have a significant advantage.

11. NEURAL NETWORKS AND OTHER DISCIPLINES

11.1 INTRODUCTION

Neural networks are being used as a tool for modeling and theory development in many disciplines besides those mentioned above. This section reviews their use:

- By experimental psychologists to model classical conditioning,
- By cognitive scientists to model performance of humans on many tasks, and
- By computer scientists in field of artificial intelligence (AI).

Neural network models in the AI field complement more conventional computational symbolic methods. Some higher-level AI problems, such as graph matching and constraint propagation, map well onto neural networks. Neural network implementations provide advantages in robustness and potential speed and seem better at dealing with uncertain and conflicting evidence. Other problems, such as variable binding, do not map well onto neural networks. Neural network models, sometimes called *connectionist* or *PDP models*, are causing a major paradigm shift in the field of cognitive science. Here, they often model empirical data better, provide learning capabilities, and often seem more natural than a computational approach based on symbols and formal logic. Experimental psychologists have used neural networks to model classical conditioning animal learning data for many years. Surprisingly, it was found that the relatively simple LMS algorithm used to train perceptrons formed a good fit to much data. Modeling other temporal aspects of behavioral data has required adding more complex time dependencies to the network models.

11.2 HIGHER-LEVEL AI PROBLEMS

Virtually all of the work discussed so far as been focussed on the lowest (earliest) levels of intelligence. Neural-style models are also being applied to higher-level problems, which have been typically attacked by symbolic computational methods. This work overlaps significantly with work described below in the section on cognitive science and is usually denoted as “connectionist” or “PDP” modeling. Good examples of this work can be found in [177,270].

Some problems which have been viewed as high-level AI tasks have elegant formulations in neural network terms. Graph matching and constraint propagation are widely used techniques and these both map nicely to structured neural networks. Applications exploiting these techniques include those in word-sense disambiguation [40,41,271], parsing [62], semantic networks [239] and visual recognition [39]. The connectionist approaches to these tasks have already exhibited advantages in robustness and potential speed and seem better at dealing with uncertain and conflicting evidence.

There are, however, many other problems which are at this time much harder to express in neural network formalisms. From a practical point of view, this means that any near-term applications will be hybrid

systems. More interestingly, the Adaptive Knowledge Processing Panel believes it is worth discovering how nature solves these problems with connectionist hardware. There are some preliminary approaches [9,10,260,246,223], but these give up the advantages of neural networks. Much more work needs to be done on basic problems like variable binding [66] and action-sequences, and on their applications to more complex AI tasks.

11.3 COGNITIVE SCIENCE

Cognitive science is a relatively new discipline forged by pressure on the old boundaries between traditional subjects like psychology, linguistics, computer science, neurophysiology, and philosophy. Over the last few years, neural network-style models have become one of the dominant paradigms of cognitive science [177,229]. Something like one third of all papers at recent national meetings in this field have had a connectionist flavor. In addition to the swing of fashion, there are several technical reasons for this change.

Psychologists have been drawn to neural network models because they often do a better job of fitting data, especially involving timing or errors [49,177]. The adaptive or learning capabilities of neural networks have also provided an extended repertoire for cognitive scientists. More generally, the computational paradigm based on many interacting information sources appears to many cognitive scientists to be more natural than one based on symbols and formal logic. It is too early to be sure, but the widespread use of connectionist formalisms could facilitate basic advances in many fields – just as other formalisms such as calculus, statistics, and logic have done.

It is already clear that neural network formalisms are uniquely well-suited to expressing models at all levels. The same formal tools, simulators, and intuitions are being exploited for models ranging from detailed neural simulation through abstract models of language and thought. Among the benefits of this general applicability are a shared scientific language across disciplines and a completely natural framework for reduction of higher-level models to their constituents. Since cognitive science is crucially concerned with these matters, it is likely that connectionist (neural network) formalisms will continue to play an increasingly important role in the discipline. As always, the scientific advances made in modeling tasks like vision, speech, and so on will have important applications in practical tasks.

11.4 CLASSICAL CONDITIONING

In some ways, there is a much better understanding of biological learning processes at the behavioral level than at the physiological level. For example, many behavioral laws of animal learning have been found that hold across a wide range of species and response systems, whereas the physiological basis of learning is just starting to be understood in a few specific cases. The theory and data of animal learning stand as both an opportunity and a challenge to neural network research: An opportunity because animal learning is still far more adaptable and efficient than that of the best neural network learning algorithms; existing animal learning theory remains an untapped gold mine of ideas for neural network research. A challenge because there is an immense database of empirical animal learning results that remains only

partially explained. This database constitutes a stringent test of neural network models of learning that claim a strong connection to biological learning systems.

The best understood animal learning process is that known as classical conditioning. Neural network researchers since Hebb in 1949 [99] have attempted to explain classical conditioning as a result of synaptic learning rules. Early models were proposed by Grossberg [89,85] and Klopff [134]. Uttley [263] showed a formal relationship between his neuron model and the Rescorla-Wagner animal learning model, and Sutton and Barto [255] showed that the Adaline model [275] was essentially equivalent to the Rescorla-Wagner model. In the 1980s, this has become a very active area, with numerous new models and extensions of old models being explored and compared [94,133,135,254].

12. TOWARD A THEORY OF NEURAL NETWORKS

12.1 INTRODUCTION

This chapter reviews those issues that a theory of neural networks must address and also discusses what has been accomplished toward this goal. Although much has been learned through empirical studies and past experience, there is a need to take a broad look at the types of constraints on computation produced by architectures based on fine-grain parallelism. Theories must be developed to explain how neural networks operate. A theory of neural networks must address issues that are specific to these highly parallel systems. Some of the major questions that should be addressed include:

- **Capability.** What problems does a network have the potential to solve? Is there a rich enough data representation available to represent a solution to the problem the network must solve?
- **Memory Capacity.** How much information can be stored in the network? (This is similar to the capability question, but specialized for networks that learn and store items in memory.)
- **Learnability.** What types of problems are learnable? Can a network be redesigned or the data representation be altered to make a problem learnable? (*Learnable* normally means: the number of training samples required to perform a function successfully grows polynomially and not exponentially in the size of the network.) This is also called *the scaling problem*. It is addressed by a branch of mathematics called *complexity theory*. Learnability can also mean training time is long – but acceptable – for a give task.
- **Data Representation.** How is data represented in the network and how does data representation affect structure, capacity, capability and learnability?
- **Network Design.** How should a network be designed to perform specific functions? What types of nodal processing elements should be used? How should they be interconnected? What modes of operation should be used? How should data be entered, operated on, extracted, and stored?
- **Learning and Using Internal World Models.** How are internal models of an environment learned and then used to plan actions and make decisions? What is the best strategy for this type of learning and can interaction in an environment speed learning? How can learning be structured to build a hierarchy of competencies that lead to more and more capable behavior?

In addition to these more global issues, there are specific questions that must be addressed concerning credit assignment and network training and operation that were stated earlier.

12.2 CAPABILITY

Theoretical work has already explored the types of mapping problems that are learnable with specific types of networks. Many of these results were described earlier when specific network models were discussed. Some, however, are general results that apply to all learning systems. Major results are listed below in separate categories.

The capabilities of single-layer perceptrons with binary inputs and outputs and binary-valued nonlinearities have been studied by many researchers [97,154,162,178,188,193,194,222]. This work demonstrated that single-layer perceptrons could only form linearly-separable Boolean mappings between the inputs and the output. Such mappings include logical NOT, AND, and OR functions, the majority gate, and the more general “at least X of N ” function. A single-layer perceptron with N inputs can with high probability also form any desired dichotomy of 2^M input patterns [194]. Mappings that cannot be computed include the exclusive OR function and the parity function (the output is “high” if the number of bits is odd and “low” otherwise) [188]. In addition, it was proven that a single-layer perceptron presented with binary images on an input field could not determine whether an image was connected [188].

Early work on multi-layer perceptrons with binary inputs and outputs demonstrated that two-layer perceptrons with one layer of hidden units could form any Boolean mapping or logic function [154,188,194]. For a number of years, perceptrons called “threshold logic gates” were explored as an alternative to more conventional NAND, OR, AND, and NOR logic elements. This work even led to small experimental hardware computers [117].

Perceptrons with continuous-valued inputs and binary-valued nonlinearities have also been studied. Early work demonstrated that a two-layer perceptron with one layer of hidden nodes could separate N finite distinct input points into any desired dichotomy [194]. This proof says little about the more typical classification problem where inputs in different classes are clustered together in one or more compact regions. Recent work led to a simple constructive procedure which demonstrates that three-layer perceptrons can form any desired decision region and that two-layer perceptrons can form convex decision regions [29,157]. This is illustrated in Figure 4-2. Further work demonstrated that two-layer perceptrons could form both non-convex and disjoint decision regions [114]. Studies using backpropagation training have demonstrated that it is possible to learn complex decision regions [113,114].

Multi-layer perceptrons can also form complex nonlinear input/output mappings. Kolmogorov proved a theorem described in [163] which proved, in effect, that a three-layer perceptron could compute any continuous nonlinear function of the inputs. This proof, however, requires accurate, problem-specific nonlinearities in the nodes. Recent theoretical results [44] have demonstrated that a three-layer perceptron with two layers of hidden units and sigmoid nonlinearities can approximate any continuous input/output mapping to any desired precision. More work is needed to determine the number of nodes required to produce such mappings.

An extension of the constructive proof described in [29,157] can be used to construct three-layer perceptrons that form specific mappings. This extension quantizes the input space into hypercubes and provides the desired output value for each hypercube by assigning connection weights to the output node [151]. It essentially uses a lookup table technique to form nonlinear mappings. The same concept was used in

the “hypercube classifier” described in [114]. Studies using backpropagation have demonstrated that it is possible to learn some important nonlinear mappings [151].

12.3 MEMORY CAPACITY

Memory capacity has been studied primarily for associative memories. Results for fully-connected Hopfield associative memories [109,23] demonstrate that $M < .014 N$, where M is the number of N bit memories that can be stored in a Hopfield network with N nodes. This bound assumes a bit error rate of 5% can be tolerated and requires N^2 connection weights. A unary or Hamming network can store M memories using N input nodes, M intermediate nodes, and N output nodes [23,156]. This requires only $2NM$ connections. For example, a Hopfield network would require 250,000 connections to store 10 500-bit patterns, while a unary network would require only 10,000 connections. The capacity of the Kanerva model is approximately $M < 0.1G$, where G is the number of internal nodes [23].

12.4 DISTRIBUTION-FREE LEARNING

A new area in machine learning called distribution-free learning has developed over the past few years. It addresses the global question of what Boolean functions are learnable and rigorously defines this problem [265]. Initial results were obtained for Boolean functions learned from examples chosen at random but with fixed distributions [265]. A function was defined to be learnable if there is a learning algorithm which requires a number of examples (t) that grows as a polynomial function of the number of inputs (d). The learning algorithm need not classify all examples correctly after training, but must with probability $1 - \delta$ have an error rate that is less than ϵ . This definition opened up a new area of study and led to numerous theoretical studies [26,25,127,264]. This work is currently being extended to situations with continuous-valued inputs and noise [25,12]. A good overview is available in [8].

One negative result from distribution-free learning studies is that general linearly-separable Boolean functions that can be produced by perceptrons are not learnable [97,127]. Bounds on the number of trials required to learn general linearly-separable functions with the perceptron convergence procedure from [97] are

$$2^d < t < d^d .$$

This is clearly not learnable since the number of examples grows exponentially in the number of inputs.

A positive result is that monotone disjunctions (OR functions) are learnable. If binary inputs to a network are denoted x_1, x_2, \dots, x_d , then examples of monotone disjunctions include $x_1 \cup x_3 \cup x_7$ and $x_2 \cup x_4$. Not only are these functions learnable, but they are learnable by simple single-layer perceptrons [97,264]. The number of trials required to learn monotone disjunctions with the perceptron convergence procedure from [97] is

$$t < d^2 .$$

In simulations, the number of trials appeared to grow only linearly with d . Recently, a variant of the perceptron convergence procedure was developed that makes a bounded number of errors when learning

monotone disjunctions [162]. The number of errors for this algorithm is bounded by

$$errors < 2k(\log_2(d+1)) + 1 ,$$

where k is the number of terms in the disjunction and d is the number of inputs. Monotone conjunctions (AND functions) are also learnable [97,264]. The number of trials required to learn monotone conjunctions with the perceptron convergence procedure from [97] is

$$t < d^3 .$$

Another class of Boolean functions that is learnable are called k-DNF meaning functions in k-disjunctive normal form. These are functions made from disjunctions (OR) of terms made from conjunctions of (ANDing) inputs where each conjunction involves at most k inputs. For example, a 3-DNF function of inputs x_1, x_2, x_3, x_4, x_5 could be $(x_1 \cap x_3) \cup (\overline{x_2} \cap x_4 \cap x_5)$. The constructive proof mentioned above [29,157], which demonstrates that a three-layer perceptron with continuous-valued inputs can form arbitrary decision regions, uses this type of function. Weights in the upper two layers compute a k-DNF function of the outputs of first-layer nodes. This function also is a subset of DNF functions which can be used to form any arbitrary Boolean function.

A number of researchers have suggested training algorithms to use with two-layer perceptrons to learn k-DNF functions [97,114,162,264]. Most algorithms use some form of unsupervised competitive learning to learn conjunctions in the first layer and supervised learning to learn disjunctions in the second layer. [96,114]. Two hierarchical training algorithms were tested on problems that are k-DNF functions (the multiplexer problem from [19] and the symmetry problem from [225]). The new hierarchical algorithms required roughly 500 training trials while $A_{\tau-p}$ and backpropagation training required more than 75,000 training trials to learn these functions. When compared on a classification problem [114], hierarchical algorithms typically required fewer than 500 trials, while backpropagation training required more than 60,000 trials. These studies illustrate the improvements in training time that can be provided using hierarchical networks by tailoring the training algorithm to match the complexity of the problem. They demonstrate that data representations which lead to learning of disjunctions are very useful because they result in rapid learning. The map representations described in [17] may be common in biology for this reason.

12.5 LEARNING AND USING INTERNAL WORLD MODELS

The necessity of building and using internal models of the world has often been noted [15,247,253]. Once learned, internal models can be used as a simulation of the external environment. Actions and situations can be proposed to the model to predict likely consequences. After perhaps several iterations of this, the action or action sequence with the best consequence can be selected. It is only recently that inroads have begun to be made in this area. This work is opening up an important area of learning where internal models of a limited environment are built through interaction and do not have to be pre-programmed into the network. This type of learning will almost certainly be required to solve the common sense problem of artificial intelligence because rules for all possible real-world events are impossible to predict and code.

The majority of neural network research has focused on direct learning to solve specific tasks instead of on building internal models. Direct learning involves learning a specific action to solve each task. For example, a specific sequence of turns could be learned to reach one specific destination in a maze. A new destination, however, would require more training and learning of a new sequence of turns. The alternative model-based learning involves building an internal model of the environment and then using this model to generate actions to solve many different tasks. For example, a mental map of a maze could be learned and used to plan paths to any destination in the maze.

The concept of internal models as playing an important role in thought has a long history in philosophy and psychology. It was also mentioned frequently in early discussions of neural network approaches [15,247]. Results over the past few years have demonstrated that models of simple maze-like environments can be built and used in neural network architectures [253,256]. One new approach is to use backpropagation to train an internal network that mimics the input/output behavior of some aspect of interacting with the environment. This model can be trained through random exploration. For example, the map from simulated motor signals on articulators that produce sound to the produced sound as monitored by simulated auditory neurons could be learned by a multi-layer network. The input to this network could be the motor signals and the desired output could be the auditory neuron outputs. This network could be trained during an exploratory babbling phase. Although this approach has been suggested by many researchers, there is only one published report in this area [192]. A similar approach, however, is described in [149,148,215]. Here, topological maps are used to learn forward and backward mappings between visual inputs and motor outputs for simple tasks.

A new approach to building internal models by interacting with a deterministic environment is described in [220,221]. In this task, an automaton performs actions that affect the environment and is provided with sensations that describe the changes in the environment. The goal of the automaton is to construct a model of the environment that perfectly predicts the result of any proposed sequences of actions. The procedure developed was able to rapidly learn the structure of complex environments. For example, it was able to learn the structure of a Rubik's Cube environment with over 10^{19} global states in only a few minutes of computing time. This work represents the beginning of an important line of research that may provide bounds on the ability of automata to learn by exploring the environment. This type of research is clearly needed to help determine how complex behaviors can be built up over time by interacting with the environment.

12.6 NP COMPLETE COMPLEXITY RESULTS

Many studies use neural networks with only hundreds of nodes. To fully exploit the power of networks, they need to be scaled up to bigger sizes. Complexity theory addresses the effect of scaling on training time and also the difficulty of problems on sequential Von Neumann machines and on neural networks.

A number of researchers have examined the complexity of problems when solved using recurrent Hopfield-type networks and nodes with two-state binary outputs [1,79,161]. One study [1] demonstrated that combinatorial problems, such as the traveling salesman problem, which are NP complete for Von Neumann machines, are also NP complete on these neural networks when the number of nodes is polynomial in the number of cities. Others have explored the complexity of determining whether such networks

have stable states [79,161]. This problem is solvable in networks with symmetric connection weights but NP-complete in networks with asymmetric weights. A good summary of this and other recent work is available in [79].

Judd has recently presented complexity theory results for multi-layer perceptrons using sigmoidal nonlinearities [122]. He studied the dependence of learning time on the size of the network for networks that are fully connected between layers and proved there was no reasonable upper bound on the training time when arbitrary input/output mappings must be learned. The applicability of these results is, however, limited. It has already been demonstrated that specific Boolean mappings can be learned in a time that scales as a polynomial in the size of the network. In addition, it has been shown that hierarchical networks and high-order networks can speed up learning dramatically. Further work is necessary in this area to obtain results for specific important mappings, for networks with limited connectivity, and for hierarchical networks.

12.7 DATA REPRESENTATION

Past work on artificial intelligence and neurobiology has demonstrated the importance of data representation [18,67,226,268]. This importance has been substantiated by research with neural network models which support the rich variety of representations described previously. A good representation can often reduce the required training time by orders of magnitude at the expense of a larger network. Presently, the study of data structures in classical computer science is based entirely on general-purpose random access machines. Current results are thus difficult to apply to neural networks. There is a need to review and revamp the study of data structures with specific attention to the constraints, strengths, properties, and requirements of machines with highly distributed qualities.

12.8 NETWORK DESIGN

Experience with many different types of neural network models and data representations is beginning to provide design guidelines. Representations and models differ in many characteristics including:

- Resource requirements (number of nodes, complexity of nodes, fidelity of nodes, robustness of nodes, and number of connections), and
- Performance limits (retrieval time, learnability, scaling, error recovery, fault tolerance, training protocols, and generalization).

To support a particular task, a system could use static, cyclic, or dynamic processes; it could be stochastic or near-deterministic; the signals could be discrete or continuous; and it could be dedicated to that task or partially shared with others. There is a need for theoretical analyses of the tradeoffs between these requirements and resources and for codification of past empirical studies of different networks. There is also a need for more analysis and study of multi-module network structures similar to those observed in biological nervous systems.

12.9 GENETIC ALGORITHMS

One new technique that is only beginning to be applied to neural networks uses genetic algorithms developed by Holland [107,108]. These algorithms represent an approach to learning modeled after the theory of evolution and the concept that the most fit (best performing) characteristics of a model should survive in future generations. It requires the initial production of many alternative models for the first generation. The performance of these models is evaluated and a new generation of models is formed by merging and randomly mutating components of old models. The process is then repeated.

Genetic algorithms have been applied to a wide variety of problems, including function optimization, keyboard layout, semiconductor chip design, job shop scheduling, and communication network design [47,108,83]. In the field of neural networks, they have been combined with simulated annealing and applied to function optimization problems [3] and also used to design networks to find shortest paths [245]. They appear to provide a general, although potentially computationally intensive, approach to the problems of network design and choosing a good input representation.

13. NEUROBIOLOGY AND NEURAL NETWORKS

13.1 INTRODUCTION

As noted previously, there are two classes of neural network models – those that are intended as computational models of biological nervous systems, and those that are intended as biologically-inspired models of computational devices with technological applications. These classes are not necessarily mutually exclusive, but most neural network models fall into one class or the other. In the rest of this chapter, the former class of networks will be referred to as neurobiological models, and the latter simply as neural network models.

The goal of neural network research is to design new algorithms and machines that can solve problems that require “intelligent” analysis for their solution – i.e., those problems that are very difficult for conventional algorithms and machines, but are easy for ‘intelligent’ biological organisms. As suggested by its name, the field of neural networks is distinguished from other fields with similar goals – e.g., artificial intelligence – by its general strategy of incorporating features of biological nervous systems into its designs. This is an important difference, for it opens the possibility of grounding neural network technology in the secure foundation of natural science, and of accelerating its progress through exploiting the knowledge of how existing (biological) systems have solved the same problems that it is concerned with.

This chapter first provides a brief overview of brain physiology, highlighting the great differences between neural and conventional computation. The biological foundations of neural network research are then presented, demonstrating that neural networks are founded in several key features of the nervous system, but that there is still much to learn from neurobiology. In support of this conclusion, recent advances in neuroscience are briefly reviewed, followed by a discussion about how such knowledge could influence neural network research.

13.2 OVERVIEW OF BRAIN PHYSIOLOGY

The brain consists of roughly 10 to 1,000 billion neurons, cells with distinctive properties found only in the nervous system. There are hundreds of types of neurons, although they share a set of common features and functions. Figure 2-2 shows drawings of two characteristic types of neurons and a small neural circuit.

A neuron is characterized by a cell body ~ 30 microns in diameter, and thin (~ 1 micron) branching extensions called dendrites and axons that are specialized for neuron-to-neuron communication. The dendrites receive inputs from other neurons and the axon provides output to other neurons. The neuron is imbedded in an aqueous solution of small ions, and its selective permeability to these ions establishes a negative electrical potential of some tens of millivolts.

Neurons receive electrochemical input signals from other cells at small (~ 1 micron) discrete sites on their surface, as shown in Figure 13-1. These sites of neuron-to-neuron communication are called synapses, and number between 1,000 to 100,000 for each neuron. The input signals are combined in various ways, triggering the generation of an output signal by a special region near the cell body under certain conditions. The output signal is a large (~ 100 millivolt), brief ($\sim .001$ second), pulse that

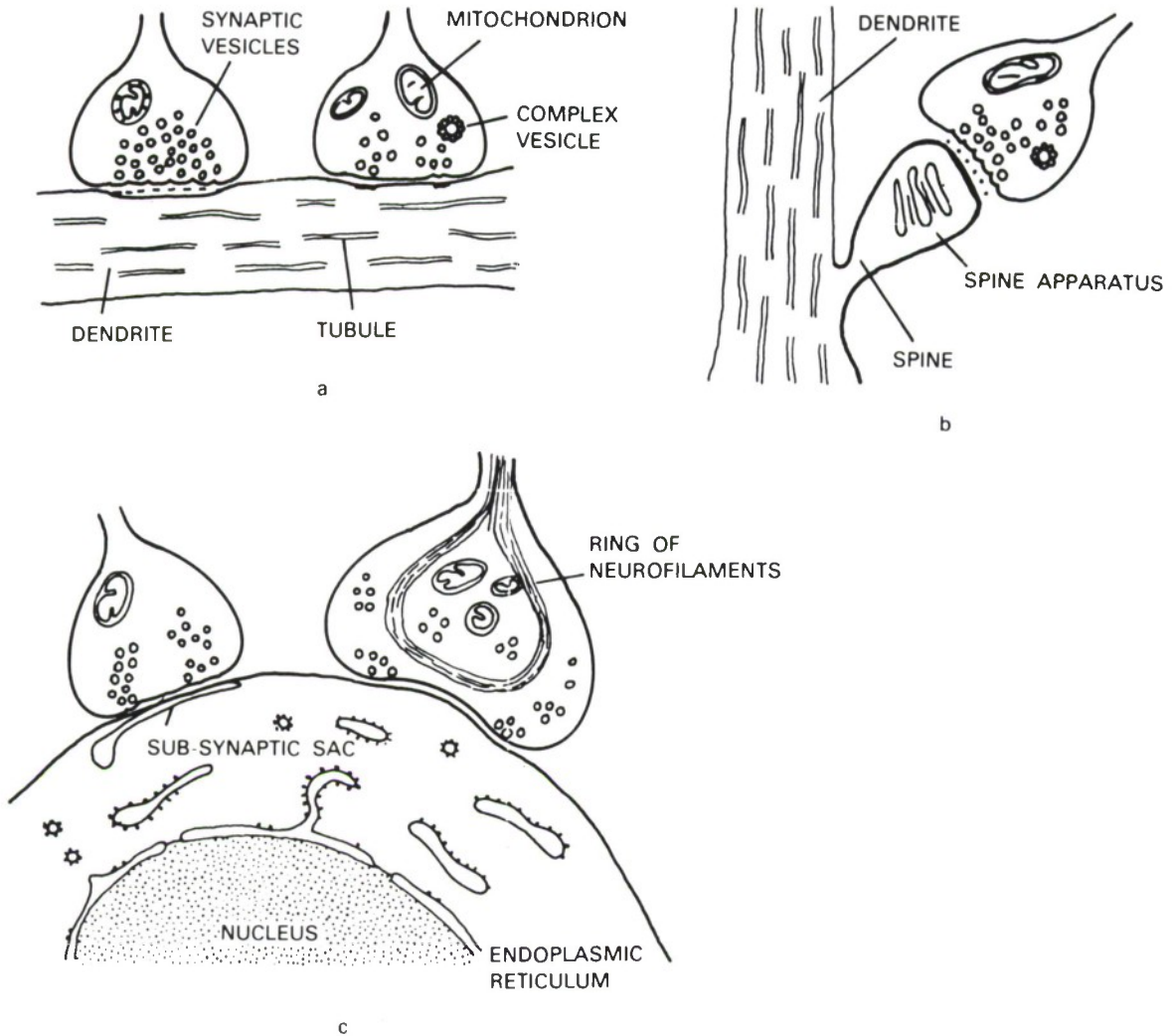


Figure 13-1. Three Types of Neuron-to-Neuron Connections Called Synapses.

propagates without attenuation down the axon at velocities up to 120 meters/second. These pulses, called action potentials, can be produced at varying rates, up to several hundred per second. The axon branches many times, delivering the same pulse to all of its synapses. Some neurons generate sustained moderate pulse rates for seconds, while others produce short bursts of a few pulses in a few milliseconds.

It is tempting to think of a neuron, with its pulse-like output, as a digital transistor with two states, pulse or no-pulse. However, this analogy requires clock-like synchronization of neuronal pulses on a time scale of the minimal inter-pulse interval – a few milliseconds. Such tight synchronization is not found in the brain (although brain regions are known to exhibit synchronized firing over larger spatio-temporal scales). Typically many pulses must be averaged over tens of milliseconds to determine the average firing rate on an axon. Given this need for averaging, it is surprising that humans can respond to complex stimuli in fractions of a second when neural elements have computing times of tens of milliseconds. This response time indicates less than 100 sequential processing steps. The relative slowness of individual neurons is presumably countered by the use of massive parallelism in the nervous system. Parallel processing is an emerging field in computer science – however, the style and fine-grain scale of neuronal parallel processing is of an entirely different order.

On the transmitting or pre-synaptic side of the synapse, the pulse releases a specific chemical, called a neurotransmitter, that diffuses across a gap ($\sim .01$ microns) to the receiving side of the synapse. On the receiving or post-synaptic side of the synapse, the neurotransmitter binds to specific receptor molecules anchored in the membrane, opening ionic channels and changing the electrochemical potential. The magnitude of this change is determined by many factors local to the synapse, such as the amount of transmitter released on the pre-synaptic side and the number of receptor molecules on the post-synaptic side. These factors can change with time, thus changing the effectiveness or “strength” of the synapse. It is not known how many of the thousands of synapses on a cell are strong, or even functional at a given time. Estimates of the number of active synapses necessary to cause a cell to “fire” (generate an output pulse) range from a few to hundreds.

It has long been thought that the neuronal input/output function was relatively simple. Synaptic potentials are spatially and temporally filtered in accord with the cable properties and geometry of the neuron and integrated at the cell body. The resulting potential triggers action potentials only when it is greater than a fixed threshold. This description is reasonably accurate for a large number of neurons, but modern methods have revealed much more complicated mechanisms. Many neuroscientists now think of a neuron more as a microprocessor rather than as a leaky capacitor feeding a one-shot flip-flop circuit. In addition, there are neurons without axons, synapses that are bi-directional, synapses onto other synapses and onto axons, and non-chemical, electrical synapses.

The neurons of the brain are organized into thousands of discrete structures, each with its own particular types of neurons, patterns of neuron-to-neuron connections, and role in brain function. Shapes and sizes vary from small globular nuclei a few millimeters in diameter with tens of thousands of neurons, to large cortical sheets a few millimeters thick by one meter square with tens of billions of neurons. Large-scale information processing tasks, such as speech or vision, are performed by systems composed of many interconnected structures, each serving a small though specific subtask. This modular architecture is largely responsible for past successes in understanding the brain. Carefully controlled experiments can selectively study one brain system at a time, untangling its components from those of others, and elucidating their role

in the systems function. Figure 13-2 shows a hierarchical block diagram of some of the many processing stages identified in the visual cortex of the macaque monkey.

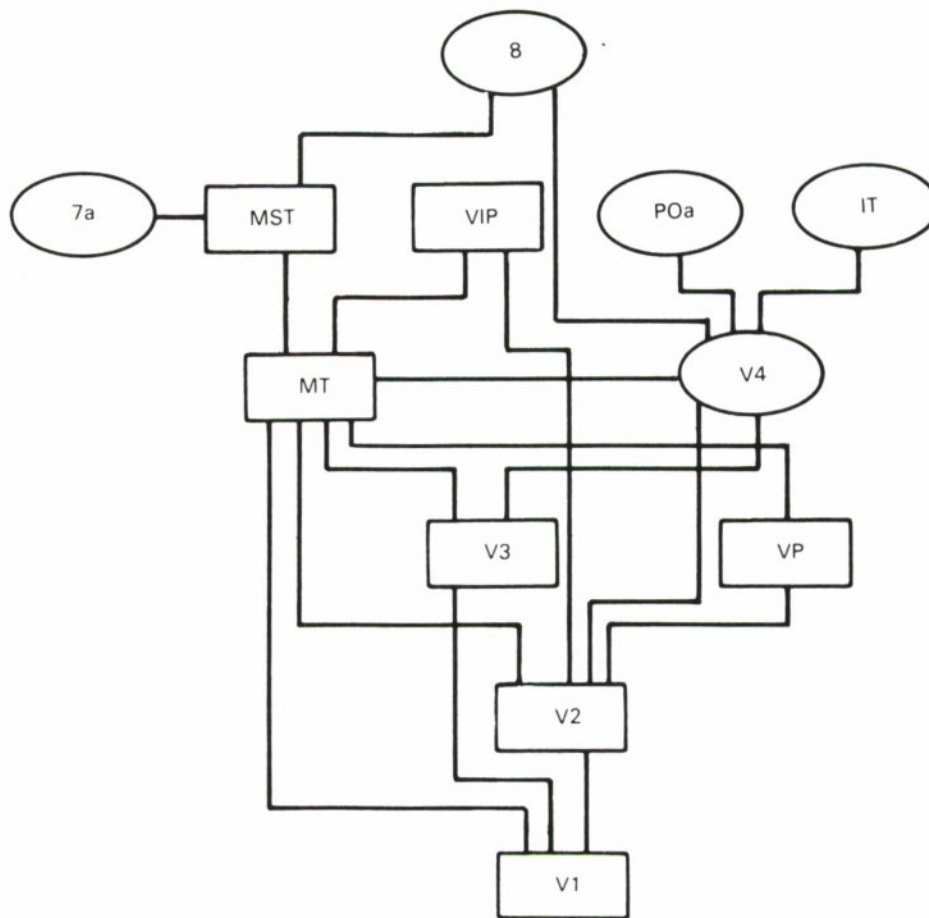


Figure 13-2. A Hierarchy of Important Processing Stages in the Visual Cortex of the Macaque Monkey. The majority of connections are reciprocal and inputs from the eye enter V1.

The growth and formation of brain structures is, of course, ultimately governed by genetic information. However, the equivalent of a schematic at the level of individual neurons does not exist. Rather, genetics specifies the types and numbers of neurons and the general patterns of connections, but leaves the details of neuronal wiring up to the adaptive processes of development. In a similar way, it is generally presumed that the acquisition of complex skills such as speech, vision, and movement – achieved with little supervision during the first years of life – is due to adaptive processes of neuronal re-organization operating within and upon the existing processing structures.

Present understanding is largely confined to the sensory and motor systems. The structures involved with more high-level cognitive functions – such as categorization, memory, decision making, and planning – have been tentatively identified, but much is still a mystery. For example, a multitude of specific

structures in the visual system that analyze aspects of the elements in an image, such as their color, motion and depth, can be pointed out – but there is no knowledge of how those aspects are grouped into a representation of the physical objects in the scene, nor how the objects are recognized, categorized, and named. However, there are no presently foreseeable scientific obstacles barring an understanding of such cognitive functions, although this will likely require a greater interaction between theory, computational models, and experiment.

13.3 BIOLOGICAL FOUNDATION OF NEURAL NETWORKS

This section briefly discusses the biological foundation of neural networks. More complete accounts may be found in recent reviews [13,42,180].

Work in neural networks is generally oriented towards achieving rather high-level intelligent functions, such as pattern recognition, categorization, and associative memory. The biological knowledge of these functions is far from complete, but it is very clear that neurons and synapses are the fundamental devices used. It is also clear that these devices are not programmed in the conventional manner; rather, problem-specific knowledge is acquired by a learning process which alters the neuronal parameters directly. These are the two principal facts of biology that have been applied to neural networks. They are the equivalent of the transistor and of the logically structured program in conventional computers. In addition, the algorithms for calculating the output of a model neuron from its input and the high synaptic connectivity used in model networks both derive from biological observations. Modern neuroscience provides a great wealth of additional information that has only just begun to be applied to neural network modeling. This is because the path from this more recent biological information to the desired intelligent functions is relatively tenuous, and the simple ideas of neurons, synapses, and learning, are themselves surprising powerful.

The few principles of neurons, synapses, and learning constitute the biological foundation of most neural networks. They are, of course, insufficient to specify a network with the kinds of high-level intelligent functions mentioned above. In order to achieve these functions, the biological foundation is supplemented with cleverly invented ideas, some drawn from other disciplines, notably physics. This non-biological approach is appropriate considering the technological goals of the research, the lack of clear alternative biological solutions, and the possibility that future research will verify that such imported ideas are in fact biological. However, if biological realism is not sufficiently maintained, neural networks will lose the ability to interact profitably with neuroscience. The synergistic relationship between neuroscience and neural networks depends on the plausibility of the neural networks as models of biological computation.

13.4 RECENT RESULTS IN NEUROBIOLOGY

Neural networks are based on relatively old biological knowledge (ca. 1950), although at the time the field began, this knowledge was relatively current. Since then, knowledge of the nervous system has mushroomed. The old ideas of the importance of neurons, synapses, and learning are still true – i.e., the original ideas of neural networks are still valid, but due to new techniques, a great deal more is known about these and other structures and phenomena. In this subsection important biological facts and principles

that have been largely ignored in neural network research, and yet could significantly advance neural network technology, will be outlined. We will emphasize the critical role played by biologically oriented neural network models and theoretical analysis in translating biological knowledge into computational technology will be emphasized. Furthermore, the resulting computational models can greatly contribute to understanding the complex biological phenomena.

13.4.1 Neurons

It is now recognized that single neurons are capable of more functions than the standard one of linear summation followed by nonlinear thresholding employed in neural networks. These enhanced functions are in the analog domain and arise from nonlinear interactions between dendritic inputs and the electrical state of the cell. This information is possible because of breakthroughs in intracellular recording technology, both electrical and optical. However, this information must be supplemented by mathematical models of membrane biophysics and computational models of how the analog functions are used in the larger computations of the system. Mathematical models of membrane biophysics are well advanced [212,202,138], and are now being extended to the network level in studies of the hippocampus [261]. Computational models of analog retinal motion detectors have also been proposed [141]. An understanding and implementation of such analog processes could lead to great increases in the computational power of individual nodes in neural networks. The technological feasibility of applying such analog processing to neural networks has been demonstrated by the construction of an artificial retina in silicon [181].

13.4.2 Synapses

The molecular mechanisms for changing synaptic efficacy, and hence neuronal signaling, are now becoming known. The work on long-term potentiation (LTP) and the NMDA receptor partially justifies the popular Hebb rule used by modelers [153], but significantly extends it as well. It is especially notable that the role of these synaptic mechanisms in behavioral learning is also being investigated [248,189]. All of these developments promise more powerful learning rules for neural networks – however, biologically oriented models relating synaptic changes to network phenomena and behavior are needed to guide the experimental work.

13.4.3 Map Representations

The importance of highly structured map-like representations in the brain is now very clear, especially in the visual system [61,52,237], where the maps are linked together in a modular, hierarchical fashion (see also the previous section on early vision research). Neuroethologists are coupling physiology with ethological and behavioral studies to pinpoint the role of specific neuronal maps in solving specific real-world problems. For example, studies of the bat are revealing how multiple cortical maps, shown in Figure 13-3 are used to solve the problem of in-flight echo-location [251]. Investigations of the target location system of the barn owl [136,146] have uncovered the presence of “computational maps” [137] that construct an acoustic map of space from binaural phase and intensity information. The acoustic map is adaptively fused or calibrated with the visual map derived from the retina, and is used to orient the head

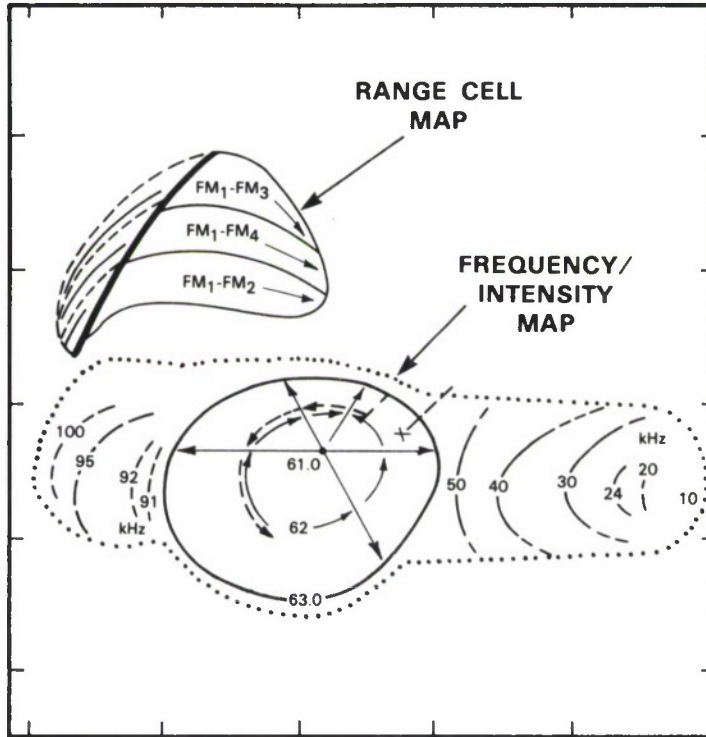


Figure 13-3. Range Cell and Frequency/Intensity Topological Maps Measured in the Auditory Cortex of the Bat. Neurons in these maps are responsive to specific characteristics of the returned bat sonar signal.

towards targets. Theoretical work could accelerate progress in this field by making experimentally testable models of how these maps are constructed and used to control behavior. In return, neural networks could gain powerful new map-like paradigms for computation and control. Models with these aims are being actively pursued [14,91,148,197].

13.4.4 Self-organization

It has long been known that the sensory systems go through brief critical periods in post-natal development in which the circuits are fine-tuned in response to environmental exposure. The environmental and physiological factors essential to this process have been extensively studied in the visual system, as reviewed in [277]. Recent work is determining the internal biochemical mechanisms [213] involved. Network models are needed to explain how all these factors work together to produce the appropriate adult circuits, and there has been considerable work in this area (see [43] and the papers cited therein). In addition to brief periods of self-organization during critical periods, there is strong evidence for reorganization in adult brains as well. This has long been predicted due to the neurological evidence of recovery after brain trauma. Recent work in the somatosensory system has demonstrated such processes of recovery at the neuronal level [123]. The cortical topographic map of the skin is dynamically regulated by the differential use of skin regions. The cortical map territory representing a given skin region will shrink or expand as that skin region is stimulated less or more than neighboring skin regions, with consequent decreases or increases of sensitivity, respectively. Models seeking to explain how this self-regulation arises from neuronal mechanisms are clearly needed, and several have been proposed [58,69,198,219]. Understanding of neuronal self-organization phenomena could lead to neural networks whose internal representations of sensory information would be maximally efficient and adaptive to changes in the environment, as discussed in [155].

13.4.5 Network Circuitry

The knowledge of the circuitry of the brain is expanding rapidly due to new tracing methods. Special dyes are injected inside a cell, filling it and marking its inputs and outputs for microscopic analysis. It has always been clear that there is much more structure in the nervous systems than neural network models consider, but now neuroscience is in a position to actually determine what the circuitry is. Studies of very young animals have shown that much of the structural complexity of brains does not come from learning or synaptic weight modification within globally connected networks, as assumed by most neural networks. No matter how much sensory deprivation a developing cat is subjected to, it will still have a retina, a lateral geniculate nucleus, a striate cortex and so forth. They might not work very well, but the basic information for macroscopic development seems to come from genetic and developmental processes rather than environmental sources. It seems unlikely that the clever designs of local brain circuits will be discovered solely with the current methods of neural networks, which start with global connectivity and change the weights to optimize an input/output mapping. However, this method may provide helpful leads in more macroscopic mapping problems, as suggested by recent modeling work [283].

13.4.6 Network Dynamics

The availability of powerful computers and electronics has enabled the simultaneous recording of electrical activity at many closely spaced sites in the brain, both electrically and optically. This promises to reveal new modes of network dynamics and information processing. However, due to the tremendous data rates involved, severe data reduction methods must be employed to solve the “needle in the haystack” problem. Networks models are needed to suggest what the important spatial and temporal scales might be in order to guide the data reduction process and the experimental design itself. Model based data reduction methods have been extensively applied in the olfactory bulb by Freeman [70] and are under development by Shaw for the visual cortex [242].

13.4.7 Learning

Neurobiologists are extensively exploring the physiological correlates of learning in higher mammals [273,262]. The response properties of single cells can now be determined while the animal is awake and learning a task or an association between stimuli. Pharmacological agents can be administered to block or aid the learning-related changes. These techniques, as well as others mentioned above, are determining the neuronal bases of learning and memory. Learning, of course, is one of the key features of neural networks. Most network models use a form of supervised learning in which the correct outputs for a large set of inputs is already known and is used to set the connection strengths to achieve the desired input/output mapping. However, there are equally important learning tasks that cannot include an omniscient teacher, such as any situation in which novelty is present. Biological research can aid neural networks in the design of such unsupervised learning paradigms (see the previous chapter on models of classical conditioning), and network models are expected to be an essential part of the discovery process itself. Recent efforts along these lines, in which there is a closed feedback loop between network modeling and experiment, has suggested a new mechanism for categorical memory [165]. This new memory model, unlike most neural network classifiers, explicitly uses time in its representation of stimulus categories. The output response pattern evolves in time, at first giving a classification response, followed by more and more of a stimulus-specific response. If two stimuli are presented at the same time, the output oscillates between the two correct outputs, thus solving the superposition problem that plagues categorizers which exclusively use spatial representations. Of course, for problems with many stimuli, further attentional methods are needed, as discussed below.

13.4.8 Attention

Animals are able to make sense of their complex sensory environments because they can attend to just those features or objects that are important to the task at hand, thus serializing their analysis of the parallel information flow provided by the senses. This essential function has been extensively studied by psychologists for the last 20 years, and is now under neurophysiological investigation (see [208] and the articles cited therein). The ability to attend to one of many objects seems especially remarkable when considering the parallel and distributed manner in which objects are represented in the nervous system. Even within a single modality, features of a single object are represented on a multitude of neural structures, and yet the

features of multiple objects are not mixed up with each other (the apples in a basket of oranges are seen as red and not orange). The problem of attention is thus related to the problem of sensory fusion.

13.4.9 Organizing Principles

Mountcastle has proposed a general organizing principle for information processing based on the modularity of brain structure and function, and the evidence that groups of neurons, rather than single neurons, are the irreducible processing elements [190]. Edelman has put forward a complementary theory of higher brain function that posits that the brain is a selective system, sharing principles in common with the immune system and Darwinian evolution [56]. This theory argues that the paradigm of information processing that is implicitly, if not explicitly, employed in both neuroscience and neural network research is an inadequate paradigm for understanding brain function and for building truly intelligent machines [57]. These organizing principles are currently very controversial within neuroscience itself – however, neural network models based on these principles have been developed [121,68,240,48,198], demonstrating new computational ideas and stimulating fresh insights into the theories.

14. CONCLUSIONS CONCERNING ADAPTIVE KNOWLEDGE PROCESSING

The Adaptive Knowledge Processing Panel of the Neural Network Study has drawn a number of conclusions about the field of neural networks and its needs and prospects. These are delineated below.

The neural network community needs sponsorship of basic research rather than funding of large-scale demonstrations of neural network capability. The field of neural networks is young but has good potential. Much basic research is required before new applications should be expected in the fields of robotics, vision, and speech.

The development of advanced neural network models should be encouraged. Researchers should explore more complex multiple-module neural network systems with many subnetworks. These systems should explore the use of hierarchical training using both unsupervised and supervised learning. They should address the problems of data fusion, selective attention, and both recognition and production of temporal pattern sequences. More complex models should not limit nodal processing elements to use only simple summations followed by nonlinearities. More biologically-plausible nodes should be studied. These nodes could exhibit complex temporal behavior and operations such as multiplication or shunting.

The development of improved learning algorithms should be encouraged. Researchers should explore techniques to automatically discover input features that provide good classification. Input features limit the performance of any following classifier or higher-level processing. Researchers should also attempt to develop faster techniques for supervised and unsupervised learning and learning techniques that work well when error feedback is noisy or delayed. They should, in addition, explore algorithms that learn to develop and use internal models of the world. Such models can be used, for example, to predict the effect of possible environmental interactions. Researchers should explore, too, the benefits of different types of data representations for different tasks as well as algorithms to recognize and predict temporal sequences.

Theoretical work in neural networks should be encouraged. The development of the field of neural networks requires further basic theoretical work. This includes research on complexity theory to determine the scaling properties of different algorithms and work to determine performance bounds on adaptive algorithms for different tasks. It also includes work that demonstrates how the complexity of a model (size and number of parameters) should be adjusted to match the amount of training data provided. Theory should also be developed to study improvements in learning that can be provided by modularizing a task and by selecting a data representation carefully. The behavior of training algorithms in non-stationary environments should be studied, as should the fault tolerance of different network architectures. Finally, theorists should study the stability and dynamics of network architectures with feedback connections.

Work in vision-oriented applications of neural networks should be encouraged. Image recognition is an important application area for neural networks. The neural network approach, however, is not a new paradigm in the field of machine vision. Here, massive parallelism is clearly necessary to obtain any kind of near realtime throughput. Many researchers are thus currently developing and using algorithms that utilize fine-grain parallelism and/or are modeled after early visual areas in the brain. Common databases do not exist, however, and should be developed as part of any new work in vision. Some of the important problems that should be addressed by vision work include:

- Detecting and fusing features from early vision (edges, color, texture, motion, depth),
- Scanning of an image, and obtaining translation, rotation, and size invariant representations of elements of a scene,
- Forming a stable body-centered internal representation of the external world.

Work in speech recognition should be encouraged. Speech recognition is another significant application area for neural networks. The neural network approach represents a new paradigm for speech recognition researchers, massive parallelism is clearly required to match acoustic inputs to 50,000 words in real time as humans do, and the performance of current speech recognizers is well below that of humans. Initial work applying neural networks in this area has been promising. Although the computational power required for speech recognition is substantial, it is much more tractable than vision and allows more extensive simulation studies. In addition, some common speech databases currently exist which could be used as benchmarks.

A solution to the speech problem will help in other areas, such as vision, where recognition hierarchies exist. Focusing on speech recognition also implies finding a solution to the problem of recognizing and storing sequences of temporal patterns. This is a major unsolved important problem in the field of neural networks. Solutions to this problem would find applications in the areas of robotics and vision. Other important problems that should be addressed include:

- Constructing sub-word, and word models automatically without excessive supervision,
- Developing better acoustic feature extraction,
- Developing rapid search techniques,
- Providing speaker independence,
- Providing good performance for continuous speech, and
- Learning and using internal models of the world.

Work in robotics should be encouraged. Robotics is a still another major application area for neural networks. Adaptive neural networks represent a different paradigm for robotic researchers that has not been thoroughly exploited. Most robots are pre-programmed to perform a specific task and are not adaptive. Initial work exploring adaptive neural network algorithms in this area has been promising. In addition, initial work on automata that coordinate eye and hand movements suggest some potential solutions to the problems of motor coordination and fusion of sensory data. Work in this area also would benefit by the specification of specific graded tasks which could be used as benchmarks. Some of the important problems that should be addressed by robotics work include:

- Fusing inputs from multiple senses,
- Adapting to variable loads,

- Planning trajectories,
- Avoiding obstacles,
- Selecting from among several possible tasks or objects, and
- Learning goal-oriented sequences.

Exploring the capability of simulated automata that “learn by doing,” initially using small simulated environments. Applications of robotics research include the development of many types of autonomous controllers that can operate in those types of uncertain environments that exist in many military applications, safe controllers for cars and other vehicles, aids for the physically handicapped, and factory automation.

Neural network models should be developed and tested on real data. Following initial development and analysis, all algorithms should be simulated as soon as possible and both tested and further developed on real data. This will require the creation of graded tasks and databases in the areas of vision and robotics. A number of shared databases already exist for speech recognition.

Sufficient computational power should be provided for neural network research. The development of adaptive learning algorithms with large databases leads to heavy computational requirements. These should be met by providing computer facilities for simulation. Ideally, these facilities should be on site near the researchers.

Interdisciplinary cooperation should be encouraged. The field of neural networks uses theoretical results and insights from many research areas. Those who sponsor neural network research should encourage interactions between modelers, neurobiologists, engineers, cognitive scientists, physicists, mathematicians, computer scientists, and others.

Neural network research efforts should focus on computational models. New computational models should not be justified by solely by neurobiological inspiration or ability to match psychophysical data. New models must offer improvements over existing algorithms. A new algorithm could have reduced computational complexity, or perform better (more accurate, shorter training time, faster operation, more capable, or more fault tolerant) or be easier to implement than existing algorithms.

The dissemination of information about neural network research should be encouraged. This is a new field covering many different areas. Information is currently scattered over many journals. Joint conferences should be encouraged, as well as review articles and books that can be understood by an intelligent reader.

REFERENCES

1. Y. S. Abu-Mostafa, "Connectivity versus entropy," in *Neural Networks for Computing, American Institute of Physics Conference Proceedings No. 151*, (J. S. Denker, ed.), 1986.
2. Y. Abu-Mostafa and J. St. Jacques, "Information capacity of the Hopfield model," *IEEE Trans. Inf. Theory*, vol. 7, pp. 1–11, 1985.
3. D. H. Ackley, *A Connectionist Machine for Genetic Hillclimbing*. Boston, MA: Kluwer Academic Publishers, 1987.
4. D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for Boltzmann machines," *Cognitive Science*, vol. 9, pp. 147–160, 1985.
5. J. Albus, *Brain, Behavior, and Robotics*. BYTE Books, 1981.
6. J. Albus, "A theory of cerebellar function," *Mathematical Bioscience*, vol. 10, pp. 25–61, 1971.
7. D. Amit, H. Gutfreund, and H. Sompolinsky, "Storing infinite numbers of patterns in a spin-glass model of neural networks," *Phys. Rev. Lett.*, vol. 55, no. 14, pp. 1530–1533, 1985.
8. J. B. Amsterdam, "The valiant learning model: extensions and assessment," Masters Thesis in Electrical Engineering and Computer Science, Massachusetts Institute of Technology, January 1988.
9. J. A. Anderson and G. E. Hinton, "Models of information processing in the brain," in *Parallel Models of Associative Memory*, (G. E. Hinton and J. A. Anderson, eds.), Lawrence Erlbaum Associates, Inc., 1981.
10. J. A. Anderson and M. C. Mozer, "Categorization and selective neurons," in *Parallel Models of Associative Memory*, (G. E. Hinton and J. A. Anderson, eds.), Lawrence Erlbaum Associates, Inc., 1981.
11. J. A. Anderson and E. Rosenfield, *Neurocomputing: Foundations of Research*. Cambridge, MA: MIT Press, 1988.
12. D. Angluin and P. D. Laird, "Identifying k-DNF formulas from noisy examples," Tech. Rep. YALEU/DCS/TR0478, Yale University, 1987.
13. M. A. Arbib, *Brains, Machines and Mathematics*. Springer-Verlag, 1987.
14. M. A. Arbib, "Levels of modelling of mechanisms of visually guided behavior," *Behavioral and Brain Sciences*, vol. In Press, 1988.
15. M. A. Arbib, *The Metaphorical Brain*. New York, NY: Wiley, 1972.
16. A. Averbuch, L. Bahl, and R. B. et al., "Experiments with the tangora 20,000 word speech recognizer," in *Proceedings IEEE International Conference on Acoustics Speech and Signal Processing*, (Dallas, Texas), pp. 701–704, April 1987.

17. D. H. Ballard, "Cortical connections and parallel processing: Structure and function," *Behavioral and Brain Sciences*, vol. 9, pp. 67–120, 1986.
18. D. H. Ballard, "Interpolation coding: A representation for numbers in neural models," *Biological Cybernetics*, vol. 57, pp. 389–402, 1987.
19. A. G. Barto, "Learning by statistical cooperation of self-interested neuron-like computing elements," *Human Neurobiol.*, vol. 4, pp. 229–256, 1985.
20. A. G. Barto and M. I. Jordan, "Gradient following without back propagation," in *1st International Conference on Neural Networks*, IEEE, June 1987.
21. B. G. Batchelor, "Classification and data analysis in vector space," in *Pattern Recognition*, (B. G. Batchelor, ed.), ch. 4, pp. 67–116, Plenum Press, London, 1978.
22. E. Baum, J. Moody, and F. Wilczek, "Internal representations for associative memory," Tech. Rep. NSF-ITP-86-138, Institute for Theoretical Physics, University of California, Santa Barbara, CA 93106, 1986. Accepted for publication in *Biological Cybernetics*, 1988.
23. E. B. Baum, J. Moody, and F. Wilczek, "Internal representations for associative memory," Univ. of California (Santa Barbara) Preprint NSF-ITP-86-138, Institute for Theoretical Physics, 1987.
24. T. V. P. Bliss and T. Lomo, "Long-lasting potentiation of synaptic transmission in the dentate area of the anaesthetized rabbit following stimulation of the perforant path," *Physiology*, vol. 232, pp. 331–356, 1973.
25. A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth, "Learnability and the vapnik-chervonenkis dimension," Tech. Rep. UCSC-CRL-87-20, University of California, Santa Cruz, CA, November 1987.
26. A. Blumer, A. Ehrenfeucht, D. Haussler, and M. K. Warmuth, "Occam's razor," Tech. Rep. UCSC-CRL-86-2, University of California, Santa Cruz, CA, February 1986.
27. H. Bourlard and C. Wellekens, "Multilayer perceptron and speech perception," in *1st International Conference on Neural Networks*, pp. IV-407, IEEE, June 1987.
28. D. Bullock and S. Grossberg, "Neural dynamics of planned arm movements: Emergent invariants and speed-accuracy properties during trajectory formation," *Psychological Review*, vol. 95, 1988.
29. D. J. Burr, "A neural network digit recognizer," in *Proceedings of the International Conference on Systems, Man, and Cybernetics*, IEEE, 1986.
30. D. J. Burr, "Speech stabilization and robust front ends for neural networks," in *Conference on Neural Information Processing Systems – Natural and Synthetic*, IEEE, Nov. 1987.
31. G. A. Carpenter and S. Grossberg, "ART 2: self-organization of stable category recognition codes for analog input patterns," *Applied Optics*, vol. 26, pp. 4919–4930, 1987.

32. G. A. Carpenter and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Computer Vision, Graphics, and Image Processing*, vol. 37, pp. 54–115, 1987.
33. H. Chen and et al., "Higher order correlation model for associative memory," in *Neural Networks for Computing, AIP Conference Proceedings 151*, (J. S. Denker, ed.), 1986.
34. J. Cohen, "Application of an adaptive auditory model to speech recognition," in *110th Meeting of the Acoustical Society of America*, (Nashville, Tennessee), November 1985.
35. M. Cohen and S. Grossberg, "Masking fields: A massively parallel neural architecture for learning, recognizing, and predicting multiple groupings of patterned data," *Applied Optics*, vol. 26, pp. 1866–1891, 1987.
36. M. A. Cohen and S. Grossberg, "Absolute stability of global pattern formation and parallel memory storage by competitive neural networks," *IEEE Transactions on Systems Man and Cybernetics*, vol. SMC-13, pp. 815–826, 1983.
37. M. A. Cohen, S. Grossberg, and D. Stork, "Recent developments in a neural model of real-time speech analysis and synthesis," in *1st International Conference on Neural Networks*, IEEE, June 1987.
38. D. B. Cooper and J. H. Freeman, "On the asymptotic improvement in the outcome of supervised learning provided by additional nonsupervised learning," *IEEE Transactions on Computers*, vol. C-19, pp. 1055–63, November 1970.
39. P. R. Cooper and S. C. Hollbach, "Parallel recognition of objects composed of pure structure," in *Proceedings, DARPA Image Understanding Workshop*, (Los Angeles, CA), DARPA, February 1987.
40. G. W. Cottrell, *A Connectionist Approach to Word Sense Disambiguation*. PhD thesis, Computer Science Department, University of Rochester, 1985.
41. G. W. Cottrell, "Connectionist parsing," in *Proceedings, 7th Annual Cognitive Science Society Conference*, (Irvine, CA), 1985.
42. J. D. Cowan and D. H. Sharp, "Neural nets and artificial intelligence," *Daedalus*, vol. 117, pp. 85–121, January 1988.
43. J. D. Cowan and D. H. Sharp, "Neural networks," *Daedalus*, 1987.
44. G. Cybenko, "Continuous valued neural networks with two hidden layers are sufficient," Tech. Rep., Department of Computer Science, Tufts University, March 1988.
45. J. G. Daugman, "Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters," *Journal of Optical Society of America*, vol. 2, pp. 1160–1169, 1985.

46. J. G. Daugman and D. M. Kammen, "Image statistics, gases, and visual neural primitives," in *1st International Conference on Neural Networks*, pp. Volume IV, 163–176, IEEE, June 1987.
47. L. Davis, *Genetic Algorithms and Simulated Annealing*. Los Altos, CA: Morgan Kaufmann, 1987.
48. S. Dehaene, J. Changeux, and J. Nadal, "Neural networks that learn temporal sequences by selection," *Proceedings National Academy Science, USA, Biophysics*, vol. 84, pp. 2727–2713, 1987.
49. G. S. Dell, "Positive feedback in hierarchical connectionist models: Applications to language production," in *Connectionist Models and Their Implications*, (D. L. Waltz and J. A. Feldman, eds.), Ablex Publishing Corporation, 1988.
50. L. Deng and C. D. Geisler, "A composite auditory model for processing speech sounds," *Journal of the Acoustical Society of America*, vol. 82, pp. 2001–12, Dec 1987.
51. J. S. Denker, "Neural network models of learning and adaptation," *Physica*, vol. 22D, pp. 216–222, 1986.
52. E. A. DeYoe and D. C. V. Essen, "Concurrent processing streams in monkey visual cortex," *Trends in Neuroscience*, vol. 11, pp. 219–226, 1988.
53. E. Domany and H. Orland, "A maximum overlap neural network for pattern recognition," *Physics Letters A*, vol. 125, pp. 32–34, 1987.
54. H. L. Dryfus and S. E. Dryfus, "Making a mind versus modeling the brain: artificial intelligence back at a branchpoint," *Daedalus*, vol. 117, pp. 15–43, January 1988.
55. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. John-Wiley & Sons, New York, 1973.
56. G. M. Edelman, "Group selection and phasic reentrant signaling: a theory of higher brain function," in *The Mindful Brain*, (G. M. Edelman and V. B. Mountcastle, eds.), pp. 51–100, MIT Press, 1978.
57. G. M. Edelman, *Neural Darwinism*. Basic Books, NY, 1987.
58. G. M. Edelman and L. H. Finkel, "Neuronal group selection in the cerebral cortex," in *Dynamic Aspects of Neocortical Function*, (G. M. Edelman, W. E. Gall, and W. M. Cowan, eds.), pp. 653–695, Wiley, 1984.
59. J. L. Elman and J. L. McClelland, "Exploiting lawful variability in the speech wave," in *Invariance and Variability in Speech Processes*, (J. S. Perkell and D. H. Klatt, eds.), New Jersey: Lawrence Erlbaum, 1986.
60. J. L. Elman and D. Zipser, "Learning the hidden structure of speech," ICS Report 8701, Institute for Cognitive Science, University of California, San Diego, La Jolla, CA, February 1987.
61. D. C. V. Essen and J. H. R. Maunsell, "Hierarchical organization and functional streams in the visual cortex," *Trends in Neuroscience*, vol. 63, pp. 370–375, 1983.

62. M. A. Fandy, *Learning in Structured Connectionist Networks*. PhD thesis, Computer Science Department, University of Rochester, 1988.
63. S. Farlow, *Self-organizing Methods in Modeling*. Marcel Dekker, 1984.
64. J. Feldman, "Connectionist models and parallelism in high level vision," *Computer Vision, Graphics and Image Processing*, vol. 31, pp. 178–200, 1985.
65. J. Feldman, M. Fandy, and N. Goddard, "Computing with structured neural networks," *IEEE Computer Journal*, vol. To Appear, 1988.
66. J. A. Feldman, "Dynamic connections in neural networks," *Biological Cybernetics*, vol. 46, pp. 27–39, 1982.
67. J. A. Feldman and D. H. Ballard, "Connectionist models and their properties," *Cognitive Science*, vol. 6, pp. 205–254, 1982.
68. L. H. Finkel and G. M. Edelman, "Interaction of synaptic modification rules within populations of neurons," *Proceedings of the National Academy of Science*, vol. 82, pp. 1291–1295, 1985.
69. L. H. Finkel and G. M. Edelman, "Population rules for synapses in networks," in *Synaptic Function*, (G. M. Edelman, W. E. Gall, and W. M. Cowan, eds.), pp. 711–757, Wiley, 1987.
70. W. J. Freeman, "EEG Analysis Gives Model of Neuronal Template-Matching Mechanism for Sensory Search with Olfactory Bulb," *Biological Cybernetics*, vol. 35, pp. 221–234, 1979.
71. K. Fukushima, "Neural network model for selective attention in visual pattern recognition and associative recall," *Applied Optics*, vol. 26, pp. 4985–4992, 1987.
72. K. Fukushima and S. Miyake, "Neocognitron: a new algorithm for pattern recognition tolerant of deformations and shifts in position," *Pattern Recognition*, vol. 15, p. 445, 1982.
73. R. G. Gallager, *Information Theory and Reliable Communication*. New York: John Wiley and Sons, 1968.
74. E. Gamble and T. Poggio, "Visual integration and detection of discontinuities: the key role of intensity edges," Tech. Rep. AI-970, Massachusetts Institute of Technology Artificial Intelligence Laboratory, October 1987.
75. S. Geman and D. Geman, "Stochastic relaxation, Gibbs distribution and the Bayesian restoration of images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 6, pp. 721–741, 1984.
76. O. Ghitza, "Robustness against noise: the role of timing-synchrony measurement," in *Proceedings IEEE International Conference on Acoustics Speech and Signal Processing*, (Dallas, Texas), pp. 2382–2385, April 1987.
77. C. L. Giles and T. Maxwell, "Learning, invariance, and generalization in high-order networks," *Applied Optics*, vol. 26, pp. 4972–4978, December 1987.

78. G. Gindi, A. Gmitro, and K. Parthasarathy, "Hopfield model associative memory with nonzero-diagonal terms in memory matrix," *Applied Optics*, 88.
79. G. H. Godbeer, "The computational complexity of the stable configuration problem for connectionist models," Masters Thesis in Department of Computer Science, University of Toronto, September 1987.
80. B. Gold, "A neural network for isolated word recognition," in *Proceedings IEEE International Conference on Acoustics Speech and Signal Processing*, (New York, NY), pp. 44–47, April 1988.
81. B. Gold, R. Lippmann, and M. L. Malpass, "Some neural net recognition results on steady state vowels," in *1st International Conference on Neural Network*, IEEE, June 1987.
82. R. P. Gorman and T. J. Sejnowski, "Analysis of hidden units in a layered network trained to classify sonar targets," *Neural Networks*, vol. 1, pp. 75–89, 1988.
83. J. J. Grefenstette, ed., *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, (Hillsdale, NJ), Lawrence Erlbaum, July 1987.
84. M. Gremillion, A. Mandell, and B. Travis, "Neural nets with complex structure: A model of the visual system," in *1st International Conference on Neural Networks*, pp. Volume IV, 235–246, IEEE, June 1987.
85. S. Grossberg, "Classical and instrumental learning by neural networks," in *Progress in Theoretical Biology*, (R. Rosen and F. Snell, eds.), New York, NY: Academic Press, 1974.
86. S. Grossberg, "Contour enhancement, short term memory, and constancies in reverberating neural networks," *Studies in Applied Mathematics*, vol. LII, no. 3, pp. 213–257, 1973.
87. S. Grossberg, "Cortical dynamics of three-dimensional form, color, and brightness perception: I. Monocular theory," *Perception and Psychophysics*, vol. 41, pp. 87–116, 1987.
88. S. Grossberg, "Cortical dynamics of three-dimensional form, color, and brightness perception: II. Binocular theory," *Perception and Psychophysics*, vol. 41, pp. 117–158, 1987.
89. S. Grossberg, "Embedding fields: A theory of learning with physiological implications," *Journal of Mathematical Psychology*, vol. 6, pp. 209–239, 1969.
90. S. Grossberg, "Nonlinear neural networks: Principles, mechanisms, and architectures," *Neural Networks*, vol. 1, pp. 17–61, 1988.
91. S. Grossberg and M. Kuperstein, *Neural Dynamics of Adaptive Sensory-motor Control: Ballistic Eye Movements*. Holland, Amsterdam: Elsevier/North, 1986.
92. S. Grossberg and E. Mingolla, "Neural dynamics of perceptual grouping: Textures, boundaries, and emergent segmentations," *Perception and Psychophysics*, vol. 38, pp. 141–171, 1985.

93. S. Grossberg and E. Mingolla, "Neural dynamics of surface perception: Boundary webs, illuminants, and shape-from-shading," *Computer Vision, Graphics and Image Processing*, vol. 37, pp. 116–165, 1987.
94. S. Grossberg and N. A. Schmajuk, "Neural dynamics of attentionally modulated Pavlovian conditioning: Conditioned reinforcement, inhibition, and opponent processing," *Psychobiology*, vol. 15, pp. 195–240, 1987.
95. S. Grossberg and D. Todorović, "Neural dynamics of 1-d and 2-d brightness perception: A unified model of classical and recent phenomena," *Perception and Psychophysics*, vol. In Press, 1988.
96. S. E. Hampson and D. J. Volper, "Disjunctive models of Boolean category learning," *Biological Cybernetics*, vol. 56, pp. 121–137, 1987.
97. S. E. Hampson and D. J. Volper, "Linear function neurons: Structure and training," *Biological Cybernetics*, vol. 53, pp. 203–217, 1986.
98. J. A. Hartigan, *Clustering Algorithms*. New York: John Wiley and Sons, 1975.
99. D. O. Hebb, *The Organization of Behavior*. New York, NY: Wiley, 1949.
100. R. Hecht-Nielsen, "Counterpropagation networks," *Applied Optics*, vol. 26, pp. 4979–4984, 1987.
101. R. Hecht-Nielsen, "Nearest matched filter classification of spatiotemporal patterns," *Applied Optics*, vol. 26, pp. 1892–1899, 1987.
102. D. Hillis, *The Connection Machine*. Cambridge, MA: MIT Press, 1985.
103. G. E. Hinton, "Connectionist learning procedures," Tech. Rep. CMU-CS-87-115, Carnegie Mellon University, Computer Science Department, June 1987.
104. G. E. Hinton, "A parallel computation that assigns canonical object-based frames of reference," in *Proceedings of 7th International Joint Conference on Artificial Intelligence*, (A. Drina, ed.), p. 683, 1981.
105. G. E. Hinton and T. J. Sejnowski, "Learning and relearning in Boltzmann machines," in *Parallel Distributed Processing*, (D. E. Rumelhart and J. L. McClelland, eds.), ch. 7, Cambridge, MA: MIT Press, 1986.
106. Hogan Presentation to DARPA Neural Net Study, 1987.
107. J. H. Holland, "Escaping brittleness: the possibilities of general-purpose learning algorithms applied to parallel rule-based systems," in *Machine Learning: An Artificial Intelligence Approach, Volume II*, (R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, eds.), Los Altos, CA: Morgan-Kaufmann, 1986.
108. J. H. Holland, K. J. Holyoak, R. E. Nisbett, and R. P. Thagard, *Induction: Processes of Inference, Learning, and Discovery*. Cambridge, MA: MIT Press, 1986.

109. J. J. Hopfield, "Neural networks and physical system with emergent collective computational abilities," *Proceedings of the National Academy of Sciences, USA*, vol. 79, pp. 2554–2558, April 1982.
110. J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences, USA*, vol. 79, Apr. 1982.
111. J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proceedings of the National Academy of Sciences, USA*, vol. 81, pp. 3088–3092, May 1984.
112. J. J. Hopfield and D. W. Tank, "Computing with neural circuits," *Science*, vol. 233, pp. 625–633, August 1986.
113. W. Y. Huang and R. P. Lippmann, "Comparisons between conventional and neural net classifiers," in *1st International Conference on Neural Networks*, pp. IV–485, IEEE, June 1987.
114. W. Y. Huang and R. P. Lippmann, "Neural net and traditional classifiers," in *Conference on Neural Information Processing Systems – Natural and Synthetic*, IEEE, Nov. 1987.
115. A. Hurlbert and T. Poggio, "Learning a color algorithm from examples," in *Conference on Neural Information Processing Systems – Natural and Synthetic*, IEEE, Nov. 1987.
116. A. Hurlbert and T. Poggio, "Making machines (and artificial intelligence) see," *Daedalus*, vol. 117, pp. 213–239, January 1988.
117. S. L. Hurst, *Threshold Logic: An Engineering Survey*. London: Mills and Boon Limited, 1971.
118. W. T. M. III, "Sensor-based control of robotic manipulators using a general learning algorithm," *IEEE Journal of Robotics and Automation*, vol. RA3, pp. 157–165, 1987.
119. M. I. Jordan, "Serial order: a parallel distributed processing approach," ICS Report 8604, Institute for Cognitive Science, University of California, San Diego, La Jolla, CA, May 1986.
120. C. C. Jorgensen, "Neural network representation of sensor graphs for autonomous robot navigation," in *1st International Conference on Neural Networks*, IEEE, June 1987.
121. G. N. R. Jr. and G. M. Edelman, "Selective networks and recognition automata," *Annals of the New York Academy of Science*, vol. 426, pp. 181–201, 1984.
122. S. Judd, "The complexity of learning in constrained neural networks," in *Conference on Neural Information Processing Systems – Natural and Synthetic*, IEEE, Nov. 1987.
123. J. H. Kaas, M. M. Merzenich, and H. P. Killackey, "The reorganization of somatosensory cortex following peripheral-nerve damage in adult and developing mammals," *Annual Review of Neuroscience*, vol. 6, pp. 325–356, 1983.
124. P. Kanerva, *Self-propagating Search: A Unified Theory of Memory*. Cambridge: Bradford Books, MIT Press, 1988. in press.

125. M. Kawato, K. Furukawa, and R. Suzuki, "A hierarchical neural network model for control and learning of voluntary movement," *Biological Cybernetics*, vol. 57, pp. 169–185, 1987.
126. M. Kawato, Y. Uno, M. Isobe, and R. Suzuki, "A hierarchical model for voluntary movement and its application to robotics," in *IEEE First International Conference on Neural Networks*, (San Diego), IEEE, June 1987.
127. M. Kearns, M. Li, L. Pitt, and L. Valiant, "Recent results on boolean concept learning," in *Proceedings of the Fourth International Workshop on Machine Learning*, (Irvine, California), pp. 337–352, June 1987.
128. J. D. Keeler, "Basins of attraction of neural network models," in *Neural Networks for Computing: AIP Conference 151*, (J. S. Denker, ed.), 1986.
129. J. D. Keeler, "Comparison between sparsely distributed memory and hopfield-type neural network models," *Cognitive Science*, 1987.
130. S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 229, pp. 671–679, 1983.
131. D. Kleinfeld, "Sequential state generation by model neural networks," *Proceedings National Academy Science, USA, Biophysics*, vol. 83, pp. 9469–9473, 1986.
132. D. Kleinfeld and H. Sompolinsky, "Associative neural network model for the generation of temporal patterns: theory and application to central pattern generators," *Journal of Neuroscience*, vol. Submitted Paper, 1987.
133. A. H. Klopf, "Drive-reinforcement learning: A real-time learning mechanism for unsupervised learning," in *IEEE 1st International Conference on Neural Networks*, (San Diego, CA), IEEE, June. 1987.
134. A. H. Klopf, *The Hedonistic Neuron: A Theory of Memory, Learning, and Intelligence*. New York, NY: Harper & Row, 1982.
135. A. H. Klopf, "A neuronal model of classical conditioning," Tech. Rep. AFWAL-TR-87-1139, Avionics Laboratory, Air Force Wright Aeronautical Laboratories, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio, October 1987.
136. E. I. Knudsen, "Synthesis of a neural map of auditory space in the owl," in *Dynamic Aspects of Neocortical Function*, (G. M. Edelman, W. E. Gall, and W. M. Cowan, eds.), pp. 375–395, Wiley, 1984.
137. E. I. Knudsen, S. du Lac, and S. D. Esterly, "Computational maps in the brain," *Annual Review of Neuroscience*, vol. 10, pp. 41–65, 1987.
138. C. Koch, "Cable theory in neurons with active linearized membranes," *Biological Cybernetics*, vol. 50, pp. 15–33, 1984.

139. C. Koch, "Computing motion using neural networks," in *Conference on Neural Information Processing Systems – Natural and Synthetic*, IEEE, Nov. 1987.
140. C. Koch, J. Marroquin, and A. Yuille, "Analog "neuronal" networks in early vision," *Proceedings National Academy Science, USA, Biophysics*, vol. 83, pp. 4263–4267, 1986.
141. C. Koch, T. Poggio, and V. Torre, "Retinal ganglion cells: A functional interpretation of dendritic morphology," *Phil. Trans. R. Soc. London B*, vol. 298, pp. 227–264, 1982.
142. T. Kohonen, "An introduction to neural computing," *Neural Networks*, vol. 1, pp. 3–16, 1988.
143. T. Kohonen, *Self-organization and Associative Memory*. Springer-Verlag, Berlin, 1984.
144. T. Kohonen, *Self-organization and Associative Memory*. Berlin: Springer-Verlag, 1984.
145. T. Kohonen, K. Makisara, and T. Saramaki, "Phonotopic maps — Insightful representation of phonological features for speech recognition," in *Proceedings of the 7th International Conference on Pattern Recognition*, IEEE, August 1984.
146. M. Konishi, T. T. Takahashi, H. Wagner, and W. E. S. C. E. Carr, "Neurophysiological and anatomical substrates of sound localization in the owl," in *Functions of the Auditory System*, (G. M. Edelman and W. E. Gall, eds.), ch. 24, Wiley, In Press.
147. B. Kosko, "Bidirectional associative memories," in *IEEE Trans. on Systems, Man, and Cybernetics*, 1987.
148. M. Kuperstein, "An adaptive neural model for mapping invariant target position," *Behavioral Neuroscience*, vol. In Press, pp. 148–162, 1988.
149. M. Kuperstein, "Adaptive visual-motor coordination in multijoint robots using a parallel architecture," in *IEEE International Conference on Automatic Robotics*, (Raleigh, NC), pp. 1595–1602, IEEE, March 1987.
150. S. Kurogi, "A model of neural network for spatiotemporal pattern recognition," *Biological Cybernetics*, vol. 57, pp. 103–114, 1987.
151. A. Lapedes and F. Farber, "Nonlinear signal processing using neural networks," in *Conference on Neural Information Processing Systems – Natural and Synthetic*, IEEE, Nov. 1987.
152. Y. C. Lee, G. Doolen, H. H. Chen, G. Z. Sun, T. Maxwell, H. Y. Lee, and C. L. Giles, "Machine learning using a higher order correlation network," *Physica D*, pp. 276–306, 1986.
153. W. B. Levy and N. L. Desmond, "The rules of elemental synaptic plasticity," in *Synaptic Modification, Neuron Selectivity, and Nervous System Organization*, (W. B. Levy, J. A. Anderson, and S. Lehmkuhle, eds.), pp. 105–121, Lawrence Erlbaum, 1985.
154. P. M. Lewis and C. L. Coates, *Threshold Logic*. New York: John Wiley and Sons, 1967.

155. R. Linskerdt, "Self-organization in a perceptual network," *IEEE Computer Journal*, vol. 21, pp. 105–117, March 1988.
156. R. P. Lippmann, B. Gold, and M. L. Malpass, "A comparison of hamming and hopfield neural nets for pattern classification," Technical Report TR-769, MIT Lincoln Lab, 1987.
157. R. P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Magazine*, vol. 4, pp. 4–22, April 1987.
158. R. P. Lippmann and B. Gold, "Neural classifiers useful for speech recognition," in *1st International Conference on Neural Network*, pp. IV–417, IEEE, June 1987.
159. R. P. Lippmann and E. A. Martin, "Multi-style training for robust isolated-word speech recognition," in *ICASSP 87*, pp. 705–708, April 1987.
160. R. Lippmann, B. Gold, and M. Malpass, "A comparison of hamming and hopfield neural nets for pattern classification," Tech. Rep. 769, Massachusetts Institute of Technology, Lexington, Massachusetts, 1987.
161. J. Lipscomb, "On the computational complexity of finding a connectionist model's stable state of vectors," Masters Thesis in Department of Computer Science, University of Toronto, October 1987.
162. N. Littlestone, "Learning when irrelevant attributes abound," in *Proceedings of the 28th Annual Symposium on Foundations of Computer Science*, pp. 68–77, IEEE, Oct. 1987.
163. G. G. Lorentz, "The 13th problem of Hilbert," in *Mathematical Developments Arising from Hilbert Problems*, (F. E. Browder, ed.), Providence, R.I.: American Mathematical Society, 1976.
164. G. Lynch, *Synapses, Circuits, and the Beginnings of Memory*. Cambridge, MA: MIT Press, 1986.
165. G. Lynch, R. Granger, J. Larson, and M. Baudry, "Cortical encoding of memory: hypotheses derived from analysis and simulation of physiological learning rules in anatomical structures," in *Neural Connections, Mental Computations*, (L. Nadel, ed.), In Press.
166. R. F. Lyons, "Experiments with a computational model of the cochlea," in *Proceedings IEEE International Conference on Acoustics Speech and Signal Processing*, (Tokyo, Japan), pp. 1975–1978, 1986.
167. D. G. MacKay, *The Organization of Perception and Action*. New York: Springer Verlag, 1987.
168. J. Makhoul, S. Roucos, and H. Gish, "Vector quantization in speech coding," *IEEE Proceedings*, vol. 73, pp. 1551–1588, 1985.
169. C. V. D. Malsburg, "Frank Rosenblatt: Principles of neurodynamics: Perceptrons and the theory of brain mechanisms," in *Brain Theory*, (G. Palm and A. Aertsens, eds.), Berlin: Springer-Verlag, 1986.

170. D. Marr, "A theory for cerebral neocortex," *Proceedings Royal Society London*, vol. 176, pp. 161–234, 1970.
171. D. Marr, "A theory of cerebellar cortex," *J. Physiology, London*, vol. 202, pp. 437–470, 1969.
172. D. Marr, "A theory of cerebellar cortex," *J. Physiol. Lond.*, vol. 202, pp. 437–470, 1969.
173. D. Marr, *Vision*. San Francisco: Freeman, Cooper and Co., 1982.
174. D. Marr and T. Poggio, "From understanding computation to understanding neural circuitry," in *Neuronal Mechanisms in Visual Perception*, (E. Poppel, R. Held, and J. E. Dowling, eds.), pp. 470–488, Neuroscience Research Progress Bulletin 15, 1977.
175. W. D. Marslen-Wilson, "Functional parallelism in spoken word-recognition," in *Spoken Word Recognition*, (U. H. Frauenfelder and L. K. Tyler, eds.), Cambridge, MA: MIT Press, 1987.
176. T. Martin, *Acoustic Recognition of a Limited Vocabulary in Continuous Speech*. PhD thesis, University of Pennsylvania, 1970. Electrical Engineering.
177. J. L. McClelland and D. E. Rumelhart, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 2: Applications*. Cambridge, MA: MIT Press, 1986.
178. W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, 1943.
179. R. McEliece and et al., "The capacity of the Hopfield associative memory," *IEEE Trans. Inf. Theory*, vol. 1, pp. 33–45, 1987.
180. T. M. McKenna, "Physiological constraints on the formal representation of neurons," in *Stochastic Methods in Biological Intelligence*, (M. Habib and J. Davis, eds.), Plenum, 1988.
181. C. Mead, "A sensitive electronic photoreceptor," in *Chapel Hill Conference on Very Large Scale Integration*, pp. 463–471, 1985.
182. C. A. Mead, *Analog VLSI and Neural Systems Course Notes*. Computer Science Dept., California Institute of Technology, 1986.
183. C. A. Mead, "Networks for real-time sensory processing," in *Conference on Neural Information Processing Systems – Natural and Synthetic*, IEEE, Nov. 1987.
184. C. A. Mead, "Silicon models of neural computation," in *1st International Conference on Neural Networks*, IEEE, June 1987.
185. C. A. Mead and M. A. Mahowald, "A silicon model of early visual processing," *Neural Networks*, vol. 1, pp. 91–97, 1988.
186. M. M. Merzenich, G. Recanzone, W. M. Jenkins, T. T. Allard, and R. J. Nudo, "Cortical representational plasticity," in *Neurobiology of the Neocortex: Dahlem Conference, Chichester*, (P. Rackic and W. Singer, eds.), Berlin: John Wiley and Sons, July 1988.

187. W. T. Miller, F. H. Glanz, and L. G. Kraft, "Application of a general learning algorithm to the control of robotic manipulators," *International Journal of Robotics*, vol. 6, pp. 84–98, February 1987.
188. M. A. Minsky and S. A. Papert, *Perceptrons – Expanded Edition*. Cambridge, MA: MIT Press, 1988.
189. R. G. M. Morris, E. Anderson, G. Lynch, and M. Baudry, "Selective impairment of learning and blockade of long-term potentiation by an n-methyl-d-aspartate receptor antagonist, ap-5," *Nature*, vol. 319, pp. 774–776, 1986.
190. V. B. Mountcastle, "An organizing principle for cerebral function: the unit module and the distributed system," in *The Mindful Brain*, (G. M. Edelman and V. B. Mountcastle, eds.), pp. 7–50, MIT Press, 1978.
191. P. Mueller and J. Lazzaro, "A machine for computation of acoustical patterns with application to real-time speech recognition," in *AIP Conference Proceedings 151, Neural Networks for Computing*, (J. S. Denker, ed.), (Snowbird, Utah), AIP, 1986.
192. P. Munro, "A dual back-propagation scheme for scalar reward learning," in *Proceedings of the 9th Annual Conference of the Cognitive Science Society*, Lawrence Erlbaum Associates, Inc., July 1987.
193. S. Muroga, *Threshold Logic and Its Applications*. New York: John Wiley and Sons, 1971.
194. N. J. Nilsson, *Learning Machines*. McGraw Hill, N.Y., 1965.
195. D. B. Parker, "Learning logic," Tech. Rep. TR-47, Center for Computational Research in Economics and Management Science, MIT, April 1985.
196. D. B. Paul, "A speaker-stress resistant hmm isolated word recognizer," in *ICASSP 87*, pp. 713–716, April 1987.
197. J. Pearson, J. J. Gelfand, W. E. Sullivan, R. M. Peterson, and C. D. Spence, "Neural network approach to sensory fusion," in *SPIE Conference on Sensor Fusion*, The International Society for Optical Engineering, April 1988.
198. J. C. Pearson, J. J. Gelfand, and J. R. Bergen, "Applications of new neural network architectures and systems," Presented to the DARPA Neural Network Study, December 1987.
199. S. M. Peeling, R. K. Moore, and M. J. Williams, "The multi-layer perceptron as a tool for speech pattern processing research," in *Proceedings IOA Autumn Conference on Speech and Hearing*, 1986.
200. A. Pellionisz and R. Llinas, "Tensorial approach to the geometry of brain function: Cerebellar coordination via a metric tensor," *Neuroscience*, vol. 5, pp. 1125–1136, 1980.
201. P. Peretto and J. J. Niez, "Long term memory storage capacity of multiconnected neural networks," *Biological Cybernetics*, vol. 54, p. 53, 1986.

202. D. H. Perkel and B. Mulloney, "Electrotonic properties of neurons: the steady-state compartmental model," *J. Neurophysiol.*, vol. 41, pp. 621–639, 1978.
203. G. E. Peterson and H. L. Barney, "Control methods used in a study of vowels," *The Journal of the Acoustical Society of America*, vol. 24, pp. 175–84, March 1952.
204. F. J. Pineda, "Generalization of back-propagation to recurrent neural networks," *Physical Review Letters*, vol. 59, pp. 2229–2232, 1987.
205. T. Poggio, "On optimal nonlinear associative recall," *Biological Cybernetics*, vol. 19, p. 201, 1975.
206. T. Poggio and C. Koch, "Synapses that compute motion," *Scientific American*, vol. 256, pp. 46–52, May 1987.
207. T. Poggio, V. Torre, and C. Koch, "Computational vision and regularization theory," *Nature*, vol. 317, pp. 314–319, 1985.
208. M. I. Posner and D. E. Presti, "Selective attention and cognitive control," *Trends in Neuroscience*, vol. 10, pp. 13–17, 1987.
209. W. Poundstone, *The Recursive Universe*. Chicago, Illinois: Contemporary Books, Inc., 1985.
210. R. W. Prager, T. D. Harrison, and F. Fallside, "Boltzmann machines for speech recognition," *Computer, Speech and Language*, pp. 3–27, 1986.
211. L. R. Rabiner and B. H. Juang, "An introduction to Hidden Markov Models," *IEEE ASSP Magazine*, vol. 3, pp. 4–16, January 1986.
212. W. Rall, "Core conductor theory and cable properties of neurons," in *Handbook of Physiology, the Nervous System*, American Physiological Society, 1978.
213. J. P. Rauschecker and S. Hahn, "Ketamine-xylazine anaesthesia blocks consolidation of ocular dominance changes in kitten visual cortex," *Nature*, vol. 326, pp. 183–185, 1987.
214. G. N. Reeke and G. M. Edelman, "Real brains and artificial intelligence," *Daedalus*, vol. 117, pp. 143–173, January 1988.
215. G. N. Reeke and G. M. Edelman, "Selective neural networks and their implications for recognition automata," *International Journal of Supercomputer Applications*, vol. 1, pp. 44–69, 1987.
216. D. L. Reilly, L. N. Cooper, and C. Elbaum, "A neural model for category learning," *Biological Cybernetics*, vol. 45, pp. 35–41, 1982.
217. D. L. Reilly, C. Scofield, C. Elbaum, and L. N. Cooper, "Learning system architectures composed of multiple learning modules," in *1st International Conference on Neural Networks*, IEEE, June 1987.
218. H. Ritter and K. Schulten, "Extending Kohonen's self-organizing mapping algorithm to learn ballistic movements," in *Neural Computers*, (R. Eckmiller and C. v.d.Malsburg, eds.), pp. 393–406, Berlin: Springer-Verlag, 1988.

219. H. Ritter and K. Schulten, "On the stationary state of kohonen's self-organizing sensory mapping," *Biological Cybernetics*, vol. 54, pp. 99–106, 1986.
220. R. L. Rivest and R. E. Shapire, "Diversity-based inference of finite automata," in *Proceedings of the 28th Annual Conference on Foundations of Computer Science*, (Los Angeles, California), pp. 78–87, October 1987.
221. R. L. Rivest and R. E. Shapire, "A new approach to unsupervised learning in deterministic environments," in *Proceedings of the Fourth International Workshop on Machine Learning*, (Irvine, California), pp. 364–375, June 1987.
222. F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books, 1962.
223. R. Rosenfeld and D. S. Touretzky, "Scaling properties of coarse-coded symbol memories," in *Conference on Neural Information Processing Systems – Natural and Synthetic*, IEEE, Nov. 1987.
224. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Interactive processes in speech perception: the trace model," in *Parallel Distributed Processing*, (D. E. Rumelhart and J. L. McClelland, eds.), ch. 15, Cambridge, MA: MIT Press, 1986.
225. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing*, (D. E. Rumelhart and J. L. McClelland, eds.), ch. 8, Cambridge, MA: MIT Press, 1986.
226. D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Cambridge, MA: MIT Press, 1986.
227. D. E. Rumelhart and D. Zipser, "Feature discovery by competitive learning," in *Parallel Distributed Processing*, (D. E. Rumelhart and J. L. McClelland, eds.), ch. 5, Cambridge, MA: MIT Press, 1986.
228. A. L. Samuel, "Some studies in machine learning using the game of checkers," in *Computers and Thought*, (E. A. Feigenbaum and J. Feldman, eds.), New York: McGraw-Hill, 1978.
229. W. Schneider, "Connectionism: Is it a paradigm shift for psychology," *Behavior Research Methods, Instruments, and Computers*, vol. 19, pp. 73–83, 1987.
230. E. L. Schwartz and Y. Yeshurun, "Towards a non-network approach to neural modeling: Some basic issues of measurement, simulation, and computational significance of brain maps," in *1st International Conference on Neural Networks*, IEEE, June 1987.
231. C. L. Scofield, D. L. Reilly, C. Elbaum, and L. N. Cooper, "Pattern class degeneracy in an unrestricted storage density memory," in *Conference on Neural Information Processing Systems – Natural and Synthetic*, IEEE, Nov. 1987.
232. T. Sejnowski and C. R. Rosenberg, "NETtalk: A parallel network that learns to read aloud," Technical Report JHU/EECS-86/01, Johns Hopkins University, 1986.

233. T. J. Sejnowski, "Higher-order boltzmann machines," in *Neural Networks for Computing, American Institute of Physics Conference Proceedings No. 151*, (J. S. Denker, ed.), 1986.
234. T. J. Sejnowski and C. M. Rosenberg, "Parallel networks that learn to pronounce english text," *Complex Systems*, vol. 1, pp. 145–168, 1987.
235. O. G. Selfridge, R. S. Sutton, and C. W. Anderson, "Selected bibliography on connectionism," Tech. Rep. TR87-509.4, GTE Laboratories Inc., Waltham, MA, December 1987.
236. S. Seneff, "A computational model for the peripheral auditory system: application to speech recognition research," in *Proceedings IEEE International Conference on Acoustics Speech and Signal Processing*, pp. 37.8.1–37.8.4, April 1986.
237. M. I. Sereno, "Review: The visual system," in *Organization of Neural Networks*, (I. W. v. Seelen, U. M. Leinhos, and G. Shaw, eds.), pp. 167–184, Weinheim: VCH Verlagsgesellschaft, 1988.
238. S. Shamma, "Speech processing in the auditory system. ii: lateral inhibition and the central processing of speech evoked activity in the auditory nerve," *The Journal of the Acoustical Society of America*, vol. 78, pp. 1622–1632, 1985.
239. L. Shastri, *Evidential Reasoning in Semantic Networks: A Formal Theory and Its Parallel Implementation*. PhD thesis, Computer Science Department, University of Rochester, 1985.
240. G. L. Shaw, D. J. Silverman, and J. C. Pearson, "Model of cortical organization embodying a basis for a theory of information processing and memory recall," *Proceedings of the National Academy of Science*, vol. 82, pp. 2364–2368, 1985.
241. G. L. Shaw, D. J. Silverman, and J. C. Pearson, "Trion model of cortical organization: toward a theory of information processing and memory," in *Brain Theory*, (G. Palm and A. Aertsen, eds.), pp. 177–191, Springer-Verlag, 1986.
242. G. L. Shaw, D. J. Silverman, and J. C. Pearson, "Trion Model of Cortical Organization and the Search for the Code of Short-Term Memory and of Information Processing," in *Systems with Learning and Memory Abilities*, (J. Delacour and J. C. S. Levy, eds.), p. In Press, Elsevier, 1988.
243. G. M. Shepherd, *Neurobiology*. New York, NY: Oxford University Press, 1988.
244. A. Sideris, A. Yamamura, and D. Psaltis, "Dynamical neural networks and their application to robot control," in *Conference on Neural Information Processing Systems – Natural and Synthetic*, IEEE, Nov. 1987.
245. T. R. Smith, "Calibration of neural networks using genetic algorithms, with application to optimal path planning," in *SOAR 87, NASA Conference Publication 2491*, pp. 519–526, August 1987.
246. P. Smolensky, "On variable binding and the representation of symbolic structures in connectionist systems," Tech. Rep. CU-CS-355-87, Department of Computer Science, University of Colorado, Boulder, February 1987.

247. G. Sommerhoff, *The Logic of the Living Brain*. London: Wiley, 1974.
248. U. Staubli, M. Baudry, and G. Lynch, "Olfactory discrimination learning is blocked by leupeptin, a thiol-proteinase inhibitor," *Brain Research*, vol. 337, pp. 333–336, 1985.
249. K. Steinbuch, "Die Lernmatrix," *Kybernetik*, vol. 1, pp. 36–45, 1961.
250. K. Steinbuch and U. Piske, "Learning matrices and their applications," *IEEE Transactions on Electronic Computers*, pp. 846–862, 1963.
251. N. Suga, "The extent to which biosonar information is represented in the bat auditory cortex," in *Dynamic Aspects of Neocortical Function*, (G. M. Edelman, W. E. Gall, and W. M. Cowan, eds.), pp. 315–374, Wiley, 1984.
252. R. S. Sutton, "Learning to predict by the methods of temporal differences," Tech. Rep. TR87-509.1, GTE Laboratories Inc., April 1987.
253. R. S. Sutton and A. G. Barto, "An adaptive network that constructs and uses an internal model of its environment," *Cognition and Brain Theory Quarterly*, vol. 4, pp. 217–246, 1981.
254. R. S. Sutton and A. G. Barto, "A temporal-difference model of classical conditioning," in *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, pp. 355–378, 1987.
255. R. S. Sutton and A. G. Barto, "Toward a modern theory of adaptive networks: Expectation and prediction," *Psychological Review*, vol. 88, pp. 135–171, 1981.
256. R. S. Sutton and B. Pinette, "The learning of world models by connectionist networks," in *Proceedings of the 7th Annual Conference of the Cognitive Science Society*, pp. 54–64, Lawrence Erlbaum Associates, Inc., 1985.
257. D. Tank and J. J. Hopfield, "Concentrating information in time: analog neural networks with applications to speech recognition problems," in *1st International Conference on Neural Networks*, IEEE, June 1987.
258. D. W. Tank and J. J. Hopfield, "Collective computation in neuronlike circuits," *Scientific American*, vol. 257, pp. 104–114, December 1987.
259. W. Taylor, "Cortico-thalamic organization and memory," *Proc. Roy. Soc. Lond. B.*, vol. 159, pp. 466–478, 1964.
260. D. S. Touretsky and G. E. Hinton, "Symbols among neurons: Details of a connectionist inference architecture," in *Proceedings IJCAI*, (Los Angeles, CA), pp. 238–243, 1985.
261. R. D. Traub and R. K. S. Wong, "Cellular mechanism of neuronal synchronization in epilepsy," *Science*, vol. 216, pp. 745–747, 1982.
262. N. Tsukahara, "Synaptic plasticity in the mammalian central nervous system," *Annual Review of Neuroscience*, vol. 4, pp. 351–379, 1981.

263. A. M. Uttley, "The informon in classical conditioning," *Journal of Theoretical Biology*, vol. 49, pp. 355–376, 1975.
264. L. G. Valiant, "Learning disjunctions of conjunctions," in *Proceedings 9th International Joint Conference on Artificial Intelligence*, pp. 560–566, August 1985.
265. L. G. Valiant, "A theory of the learnable," *Communications of the ACM*, vol. 27, no. 11, pp. 1134–1142, 1984.
266. J. von Neumann, *The Theory of Self-Reproducing Automata*. Urbana, Illinois: University of Illinois Press, 1966.
267. A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang, "Phoneme recognition using time-delay neural networks," Technical Report TR-1-006, ATR Interpreting Telephony Research Laboratories, Japan, 1987.
268. D. Walters, "Response mapping functions: classification and analysis of connectionist representations," in *1st International Conference on Neural Networks*, IEEE, June 1987.
269. D. Walters, "Selection of image primitives for general-purpose visual processing," *Computer Vision, Graphics and Image Processing*, vol. 37, pp. 261–298, 1987.
270. D. L. Waltz and J. A. Feldman, *Connectionist Models and Their Implications*. Ablex Publishing Corporation, 1988.
271. D. L. Waltz and J. B. Pollack, "Massively parallel parsing," *Cognitive Science*, vol. 9, pp. 51–74, 1985.
272. R. L. Watrous and L. Shastri, "Learning phonetic features using connectionist networks: an experiment in speech recognition," in *1st International Conference on Neural Networks*, pp. IV–381, IEEE, June 1987.
273. in *Memory Systems of the Brain*, (N. M. Weinberger, J. L. McGaugh, and G. Lynch, eds.), Guilford Press, 1985.
274. P. Werbos, *Beyond Regression: New T*. PhD thesis, Harvard, August 1974.
275. B. Widrow and M. E. Hoff, "Adaptive switching circuits," in *1960 Wescon Convention Record Part IV*, pp. 96–104, August 1960.
276. B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Prentice-Hall, NJ, 1985.
277. T. N. Wiesel, "Postnatal development of the visual cortex and the influence of environment," *Nature*, vol. 299, pp. 583–591, 1982.
278. G. V. Wilson and G. S. Pawley, "On the stability of the traveling salesman problem algorithm of Hopfield and Tank," *Biological Cybernetics*, vol. 58, pp. 63–70, 1988.

279. H. R. Wilson and J. D. Cowan, "Excitatory and inhibitory interactions in localized populations of neurons," *Journal of Biophysics*, vol. 12, pp. 1–24, 1972.
280. S. Wolfram, *Theory and Application of Cellular Automata*. World Scientific, 1986.
281. M. K. Wong and H. W. Chun, "Toward a massively parallel system for word recognition," in *Proceedings IEEE International Conference on Acoustics Speech and Signal Processing*, pp. 37.4.1–37.4.4, April 1986.
282. Y. Yeshurun and E. L. Schwartz, "An ocular dominance column map as a data structure for stereo segmentation," in *1st International Conference on Neural Networks*, IEEE, June 1987.
283. D. Zipser and R. A. Andersen, "A back propagation programmed network that simulates response properties of a subset of posterior parietal neurons," *Nature*, vol. 331, p. In Press, 1988.

Part III

**ASSESSMENT OF NEURAL NETWORK
TECHNOLOGY**

Edited by Thomas J. Goblick

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS	v
LIST OF TABLES	vi
III ASSESSMENT OF NEURAL NETWORK TECHNOLOGY	
1. OVERVIEW	1
1.1 Objectives of Technology Assessment Panel	1
1.2 Approach Taken in the Comparison Study	1
1.3 Caveat Emptor	3
2. METHODOLOGY	5
2.1 Neural Networks as Information Processing Systems	5
2.2 Artificial Intelligence: A Brief Review	8
2.3 Neural Networks as Knowledge-Based Systems	11
2.4 Measures Used In Comparing Neural Networks With Other Technologies	12
3. COMPARISON OF NEURAL NETWORKS WITH OTHER TECHNOLOGIES	15
3.1 Neural Network Applications Used In The Comparison Study	15
3.2 The Traveling Salesman Problem	17
3.3 Associative Memories	18
3.4 Pattern Classification Applications	22
3.5 Computational Maps For Robotic Control	29
3.6 Signal Processing Applications Of Neural Networks	30
3.7 Speech Applications	37
3.8 Machine Vision	39

4. CONCLUSIONS CONCERNING THE ASSESSMENT OF NEURAL NETWORK TECHNOLOGY	49
4.1 Current Status of Neural Network Technology	49
4.2 Expectations For Neural Network Technology	51
4.3 Conclusions Regarding Neural Network Technology	54
REFERENCES	56

LIST OF ILLUSTRATIONS

Figure No.		Page
3-1	Hamming Network	20
3-2	Heartbeat from Noise Demonstration Using the Backpropagation Network	32
3-3	Prediction Accuracy for the Backpropagation Network	35

LIST OF TABLES

Table No.		Page
2-1	Comparison of Several Types of Implementations of Information Processing Systems	7
3-1	Neural Network Applications Used in the Comparison Study	16

1. OVERVIEW

1.1 OBJECTIVES OF THE TECHNOLOGY ASSESSMENT PANEL

The DARPA Neural Network Study attempts to answer the basic questions concerning the technology of neural networks, such as “Should we be interested in neural networks, and if so, why?” and “What should be done in order to reap the potential benefits of neural networks?”

To answer such questions, it is necessary to determine what it is that neural networks do, how well they can do these things, and even more specifically, how well they can do these things compared to the standard ways of doing them. It is necessary to put neural networks in context with today’s available technology by comparing and contrasting neural networks with the current technological alternatives. In the case of a new and immature technology, it is critically important to attempt to project into the future to determine its potential benefits, even though what is currently achievable may not be very interesting. If, on the basis of current assessments and future projections, neural networks appear to offer significant advantages, then goals can be set for the new technology and a program can be planned to achieve those goals in some reasonable time-frame, commensurate with available support.

Therefore, the objectives of Panel 1 of the Neural Network Study, *the Technology Assessment Panel*, (originally titled in the Study’s Terms of Reference ‘Intelligent Systems – Status and Expectations’) are these:

- To provide a reading of the current status of neural networks relative to other technologies, and
- To try to make projections of what might be expected from neural networks in the future, given some investment in research and development efforts.

1.2 APPROACH TAKEN IN THE COMPARISON STUDY

In order to compare neural networks with other technologies, the Panel first put neural networks in the context of information processing technology, and then compared neural networks with alternative approaches to information processing. The competing technologies considered here include:

- Signal processing,
- Communication and information theory,
- Adaptive control systems,
- Pattern recognition/classification,
- Artificial intelligence (AI),
- Computer science, and

- Optimization theory.

There are many applications and many different approaches to information processing that must be considered in evaluating the worth of neural networks. The Panel has found applications where neural networks are superior to standard approaches and other applications where they are not competitive. Thus, at this stage in the development of neural network technology, it does not appear to be possible to draw general conclusions regarding the capabilities of neural networks relative to other approaches to information processing. The stance taken in this Study was, therefore, to undertake a case-by-case evaluation of neural network applications, comparing them with standard approaches in an effort to assess the current capabilities of neural networks in each of the application areas and to make projections on how well they might do in the future.

An important constraint on this approach was the need for quantitative performance data for both neural networks and alternative approaches to a problem, where the experimental conditions and data were the same for all approaches. It proved to be difficult at this time to find papers with such careful comparison data. In a time-constrained Study such as this one, the Panel found it impossible to do detailed analysis to generate its own comparison data for applications. The neural network applications studied in this section are therefore largely determined by the availability of comparison data.

Comparisons made in the context of applications must necessarily be done at the “black box” level – i.e., with respect to macroscopic performance measures and characteristics – because comparison at a detailed level with such a diversity of information processing technologies would not be practical or useful. Hence, in comparing a neural network with some other system for a particular application, the relevant questions are:

- How well does the neural network perform with respect to the competition, using the standard performance measures applicable to the specific application context being considered? That is, does the neural network get the right answers? Often enough?
- How quickly does the neural network get its answers? What is the response time of the neural network compared to the alternative approaches?
- How reliable can the neural network implementation be? Will it be suitable/unsuitable for applications involving stressing physical environments for equipment? Will it require specialists to maintain the equipment regularly?
- How quickly can neural network systems be developed, including obtaining information on which to base system designs, writing software, formulating and analyzing models, designing hardware?
- What physical constraints (size, weight, power consumption) can neural network systems meet?
- How much will neural networks cost relative to other approaches?

Given the Panel's interest in assessing the current capabilities as well as the potential benefits of neural networks in the future, the comparison studies should measure the differences between neural networks and other approaches to assess where neural networks are today, but also indicate how much neural networks would have to improve to overtake the other approaches.

1.3 CAVEAT EMPTOR

The assessment of future capabilities of a new field of technology has been attempted many times in the past, and almost always with abysmally poor results – especially in fields which are lacking in a sound overall theoretical framework. The development of digital circuit technology progressed from solid state technology to MSI, LSI, and on to VLSI technology much more rapidly than anyone projected, while the field of artificial intelligence (AI) progressed much more slowly than workers projected in the mid-1960s. The field of superconductivity moved very slowly since the first discoveries in the 1950s, but recently has exploded in an astounding manner with discoveries that were once believed impossible (and perhaps not pursued because of that belief).

This Part of the Neural Network Study's Technical Report must be viewed in the context of technology assessment and projection, with the acknowledgement that this is a very risky business. And in this brief examination, generalizations and projections must be made on the basis of a limited set of applications for which there exist sufficient quantitative data for careful comparison of neural networks and other approaches. But such projections are an essential part of any effort which attempts to determine the value of a new technology and how to reap its benefits. The work of the Technology Assessment Panel is not marketing and sales – that is, merely to provide hyperbole to maintain interest in the rest of the Report – but to actually let the cards fall where they may in making an objective assessment of neural networks and their potential for DARPA and the nation as well.

2. METHODOLOGY

2.1 NEURAL NETWORKS AS INFORMATION PROCESSING SYSTEMS

Neural networks have been defined in the beginning of this Report in a manner which clearly distinguishes them from other technologies. Neural networks are, among other things, information processing systems, and it will be useful for this Panel's comparative analysis here to discuss them in this context.

An information processing system (IPS) processes inputs to extract information from them, and produces outputs corresponding to the extracted information.

The functional specification of the transformation between inputs and outputs of an IPS is given by what is called an *algorithm*. The sense in which 'algorithm' is used here is somewhat broader than the computer science usage in that any type of specification is accepted, as long as it is well-defined. This would include mathematical formulas, sets of equations and constraints, rules, a computer program, etc.

The physical realization of the processing mechanism that runs the algorithm is referred to as the *implementation* of the IPS. For purposes of this comparative study, it will be useful to discuss IPSs on these two levels – the algorithmic level and the implementation level.

From the definition of an IPS, it is clear that an alternative view of an IPS is as a mapping from the space of all possible inputs to the output space. This view is sometimes helpful in providing insights into what is happening during the processing operations. This view is especially useful in characterizing pattern classifiers (see the discussion of multi-layer perceptrons in *Part II, Chapter 4* of this Report). Associative memory problems can also be viewed this way. A learning system can be viewed as one which formulates the mapping function from the training example presented to it.

IPSs are implemented in a variety of technologies. To illustrate:

- Signal processors may be built of analog and/or digital circuits, or even optical components, and frequently include programmed microprocessors;
- Pattern classifiers, optimization algorithms and AI systems are most often realized in the form of a computer program running on a digital computer;
- A robotic manipulator/controller can be realized as a digital computer, as an analog control system, or as a hybrid;
- An adaptive controller is often realized using analog computer techniques.

Although neural networks are implemented most frequently today as programs running on digital computers, they can also be realized using digital circuits and chips, and more recently, as analog circuits and chips, as well as with optical processing. Biological organisms represent yet another type of implementation of neural networks, in so-called *wetware*.

An interesting progression can be seen in the development of computers, starting with analog computers, then to programmable general-purpose (uniprocessor) digital computers, to parallel digital computers with

a flexible interconnection network between processing elements (PEs), to data flow machines which match topology to algorithm for maximum concurrency (for a particular class of processing problems).

To put some of the basic concepts in perspective, and thereby give a better feeling for the “style” of information processing performed by neural networks, Table 2-1 compares parallel digital computers, analog computers, biological systems and an analog realization of a neural network. The key differences that emerge from this Table are that neural network implementations are based on:

- A small number of different types of simple processing elements (PEs);
- Simple representation and communication of data in terms of physical quantities, such as voltages, charges, concentration of certain molecules, etc.;
- Adaptive PEs that each can store a small amount of data;
- Fine-grain parallelism in a network of many simple PEs;
- A network that directly implements the algorithm and thus matches the algorithm exactly, giving maximum parallelism;
- Training the network to adjust algorithm processing parameters to give the desired responses as well as to compensate for inaccuracies and faults in the hardware arising during fabrication.

Briefly, neural networks embody (a) the use of simple representations and PEs as in analog computers, and (b) the matching of topology to algorithm as in analog computers and data flow machines – resulting in fine-grain parallelism on a massive scale. However, the use of adaptive elements to simultaneously realize a computational capability integrated with a distributed data storage system is unique. The adaptation of the processing with experience is also not exploited in other forms of information processing to the extent that it is, or could be, in neural networks.

One can view neural networks as massively parallel analog computers implemented in analog VLSI, but using simple, adaptive functional elements.

Neural networks incorporate a combination of certain features of various types of implementations of IPSs together with some unique features – the sum total of which results in significant advantages for information processing. Massively parallel algorithms can provide rapid processing if implemented on an appropriate processing architecture, especially one which is matched to the algorithm. The use of only a few different types of simple processing elements and simple representations facilitates fabrication of such massively parallel systems. The adaptive feature of the PEs also allows element inaccuracies to be compensated for during a training phase. Massively parallel algorithms that deal with noisy inputs are usually tolerant of internal “noise” caused by fabrication imperfections. Training the system to adjust algorithm parameters and to compensate for fabrication imperfections also provides very important advantages in dealing with the detailed knowledge necessary to build the system; this issue will be discussed in some detail following a quick description of artificial intelligence.

	PARALLEL COMP.	DATA FLOW MACH.	ANALOG COMP.	NEURAL NETWORK
MATCH OF MACH. ARCH. & ALGORITHM	PROBLEM	GOOD	PERFECT	PERFECT
DATA REPRESENTATION	DIGITAL	DIGITAL	ANALOG	ANALOG
TYPE PROCESSING ELEMENT	ALU	ALU	ANALOG	SIMPLE ANAL.
SINGLE vs MULTI-FUNCTION PEs	MULTI	MULTI	SINGLE	SINGLE
TIME: CONTINUOUS vs DISCRETE	DISC.	DISC.	CONT.	EITHER
PARALLELISM: SCALE	~ 1000s	~ 1000s	~ 100s	1 TO 10**6
GRAIN SIZE	TASK	OPER.	OPER.	FINE GRAIN OPS.
DATA STORAGE	BANK	BUFFERS	ANALOG	IN EACH PE
COMMUNICATION: TOPOLOGY	GEN'L	SPECIAL	ALGO.	ALGO.
LINKS	MSGS	ARGS	VALUES	VALUES
ELEMENT ADAPTABILITY	NO	NO	NO	YES
ALGORITHM SETUP	PROGRAM	GRAPH	WIRE UP	WIRE & TRAIN

Table 2-1.

Comparison of Several Types of Implementations of Information Processing Systems

2.2 ARTIFICIAL INTELLIGENCE: A BRIEF REVIEW

Artificial intelligence is commonly described as a field that seeks to make computers more useful by having them solve problems which require “intelligence.”

2.2.1 What Is AI?

It is a higher goal of AI to understand intelligence itself in humans as well as in other organisms, and to be able to exploit such understanding to build “intelligent systems” – i.e., systems that do intelligent things. As a discipline, then, artificial intelligence consists of a collection of techniques, models, algorithms, and programs; these have been developed since the mid-1950s.

Among some of the particular aspirations of AI researchers are:

- Medical diagnosis by computer, with prescription of therapy;
- Machine translation of natural languages;
- The “voice typewriter” – a machine that converts spoken language directly into typewritten form;
- Discovery and demonstration of proofs for new mathematical theorems;
- Machine vision – interpretation of scenes based on images produced by a camera or other sensor.

Clearly, understanding how to build systems which accomplish the tasks described above would lead to many valuable, practical applications, and they have been aspirations of AI for many years; they remain today, however, areas of research rather than accomplished feats.

2.2.2 The Methodology of AI

The problems discussed above are clearly very difficult and involve far more than the ability to program a computer. A great deal of effort has been expended in trying to understand how people accomplish these tasks. Psychology, cognitive science, neuroscience, and biophysics have all made important basic contributions to the efforts of AI workers to solve problems like the ones cited above. AI has also benefitted from the work of electrical engineering, computer science, mathematics, and system theory. These fields have all provided the technology base on which AI has attempted to build new approaches to difficult problems.

In the early work, problems were characterized as systems of constraints and effort was focused on finding solutions satisfying the constraints by searching the solution space. The size of the solution space grew so rapidly with the complexity of the problems, that this approach was engulfed in “combinatorial explosion.” Pruning the search tree was attempted by use of heuristics – rules that applied to specific situations in the search process that could eliminate many search paths without having to evaluate them.

Indeed, so useful became the widespread application of rules to guide the search of the solution space that the emphasis in AI methodology gradually shifted from an effort to find general principles of problem solving effective for many problems to the use of domain-specific knowledge in the form of rules or packets of knowledge called *frames*. Successes with this approach have spawned a subfield in AI called “expert systems” in which the problem-solving behavior of human experts in narrow problem domains is studied to capture domain-specific knowledge that allows a computer to perform as well as the human expert.

AI today does not have a theoretical framework or models that guide the problem-solving activity. It is largely experimental in that most AI projects result in the writing of a computer program to demonstrate certain approaches to problems.

2.2.3 What Is AI Working On Today?

The list of problems that are currently being addressed in AI is very long; what they share is the need for intelligence to derive a solution. Among present AI interests are:

- Computers to aid physicians in medical diagnosis;
- Continuous speech recognition, independent of the speaker;
- Machine translation of natural languages;
- Computers that are capable mathematical assistants to engineers;
- Machine vision systems for robots and interpretation of sensor data;
- Machine reasoning and planning systems, including automatic computer programming systems;
- Expert systems for a wide variety of applications, including –
 - Decision-making systems,
 - Control systems, and
 - Tutoring systems;
- Hardware and software tools to facilitate development of AI systems.

These are certainly closely related to the initial list of AI aspirations noted earlier, the continued presence of these problems does not mean that workers in AI have not accomplished anything. On the contrary, each of these problem areas is a field of research in its own right involving accomplishments on many levels before the claim can be made that the field is no longer of research interest.

For instance, in the case of the “voice typewriter,” a machine commercially available today can convert spoken words to typewritten form with greater than 95% accuracy, and for vocabularies of many thousands of words. But the words must be isolated – i.e., spoken with an interval of silence between each one – and,

furthermore, each system can “understand” the speech of only one person. That is, speaker-independent recognition of continuous speech has not yet been achieved except for very limited contexts, like spoken digit strings. But a great deal has been accomplished in the area of the voice typewriter.

To further illustrate, consider yet another example. Machine vision technology is not currently able to analyze and interpret a complete scene from images, except in special limited contexts. But object recognition by machine is advanced enough to allow mobile robots to avoid obstacles while navigating to an objective. Similarly, robots in manufacturing plants can find and control manipulators to pick up desired objects (e.g., parts on an assembly line).

Similar experiences can be cited for all of the other major problem areas in AI, all of which would indicate that tremendous progress has been made since the birth of AI.

2.2.4 How AI Relates to Neural Networks

The history of neural networks has been intertwined with AI since the inception of both fields. In the late 1950s, when a group of scientists turned their attention to attempting to build intelligent systems, two different approaches emerged: one focused on **how the brain did things**, the other concentrated on **what the brain did irrespective of how it was accomplished biologically**.

In those early days, progress came slowly to both approaches, but the second approach – later dubbed *artificial intelligence* – became favored over the first approach – *neural networks* – for several reasons:

- Rapid advances in computer technology provided flexible and powerful tools for testing models and system concepts in software. This capability was put to good use by those working on the artificial intelligence approach because . . .
- There was a paucity of knowledge about the brain and how it works, and obtaining the necessary detailed knowledge was difficult; meanwhile, psychology was expanding into new and interesting areas of cognition and fed the appetite for knowledge to encourage the AI approach.

The neural network approach was, of course, pursued continuously over the last 30 years, but at a lower level of effort (and funding) compared to AI. In the early 1980s, several key papers – most notably those of Hopfield, Kirkpatrick, Hinton, Grossberg, and Rumelhart – spurred a revival of interest in neural networks. This interest has continued to expand through the 1980s and has led to the Study whose findings are delineated in this Report.

It is clear that both AI and neural networks are addressing the same sort of problems, and this should come as no surprise given their common roots. But it is also abundantly clear that AI and neural networks still approach these problems from very different perspectives.

AI seeks to exploit knowledge in many forms to solve complex problems – sometimes even acquiring this knowledge by observing the problem-solving behavior of human beings. The AI approach, largely experimental, implements systems in software, usually on powerful LISP-based workstations. The topic of learning has not been considered in the mainstream of AI research until recently.

Workers in the neural network arena, meanwhile, still seek to build structures inspired by biological systems – employing many simple, neuron-like processors operating in parallel – and adapting their behavior to improve system performance. Advances in neuroscience and computers fuel the neural network approach to developing intelligent systems, which has gained tremendous popularity in the last five years. The neural network approach to systems considers learning and parallel implementations from the outset, which seems to lead more naturally to special-purpose hardware implementations. The prognosis concerning the outcome of the “competition” between AI and neural networks is one of the major focal points of this Report.

2.3 NEURAL NETWORKS AS KNOWLEDGE-BASED SYSTEMS

Consider an AI-based information processing system (IPS) that is used to do medical diagnosis. The symptoms of a patient and the results of laboratory tests are the inputs that are entered into the system, and it produces a list of probable diseases. This “expert system” implementation of AI techniques uses rules of the form “*IF ... , THEN ...*”, which aid in establishing a diagnostic strategy as well as recognizing significant medical conditions.

The fact that such a system contains knowledge that it uses in processing its inputs is clear because of the explicit knowledge representation in the form of rules. There is no question that such a system can perform medical diagnoses much better than humans who have not acquired any more than the average amount of medical knowledge of the general public. Such a medical diagnostic system is certainly an IPS, but with embedded knowledge that is needed for it to perform its intended function. We refer to such systems as “knowledge-based” systems.

When a knowledge-based system has more knowledge than a human in a certain area of expertise, the human may consider the system to be “intelligent” – hence the usage of the term “intelligent systems” has recently arisen from the work of AI. If the human studies the system until he/she understands how to process its inputs to obtain the correct outputs, then he/she has acquired the knowledge contained in the system. (He/she may no longer consider the system to be “intelligent,” even though the system itself has not changed in any respect; the only changes that occurred were, in fact, the acquisition of certain knowledge by the human being.)

What is the nature of knowledge that is embedded in these “intelligent,” or knowledge-based, systems? The human being without the expert knowledge cannot produce the correct response to inputs – that is, there is uncertainty in the mind of the human before he/she has acquired the expert knowledge, and no uncertainty about how to process inputs correctly *after* acquisition of the expert knowledge. This reduction in uncertainty corresponds to an “information gain” on the part of the human observer. Knowledge thus is equated to information in the usual information theoretic sense of “that which reduces uncertainty.”

Consider another IPS which is hard-wired to process a vector input to determine which of a number of classes it belongs to. The vector could be composed of samples of the output of a communication receiver containing signal-plus-channel noise, or a feature vector in a pattern classification context. The processor now consists of digital circuits implementing certain logic functions to find, in the first case, the most probable transmitted waveform, and in the second, the name or label of the class to which the input belongs.

Is this a knowledge-based system? If a casual observer cannot do the classification task, then the system clearly uses knowledge or information that the observer does not. The knowledge in this case is in the *structure* of the hard-wired hardware implementation, and it is not so explicitly recognized as it was in the case of the expert system, but it is there nevertheless.

In building “intelligent,” or knowledge-based systems, the acquisition of the knowledge is a critical part of the job. In certain fields, such as system theory or information theory, experimental data from the problem domain is studied and models are formulated and analyzed, leading to system designs which are then implemented. The knowledge has been acquired by the engineers and scientists and embedded in the system implementation. On the other hand, the class of problems addressed by expert systems were impractical to pursue in this same manner because of the difficulty of modeling and analysis. Instead, the problem-solving process of human experts is observed in an attempt to “capture” the knowledge of these experts. Then this domain specific knowledge is coded in the form of rules or frames and put into a computer program which can make use of it in this form to solve problems in this domain.

Biological systems take yet another approach – that of passing knowledge of the general structure of a successful IPS to offspring via *genes*, and also by acquiring knowledge from *direct experience* in learning to survive in their natural environment.

Trainable neural networks are very similar to biological systems from the point of view of knowledge acquisition. The approach taken to acquire the knowledge to build a neural network is usually very different from the approach of system theory or AI, and it can have a marked effect on the total effort needed and the final cost of the system. One important issue that the Technology Assessment Panel includes in its comparisons of neural networks with other conventional technologies is that of knowledge acquisition and embedding.

2.4 MEASURES USED IN COMPARING NEURAL NETWORKS WITH OTHER TECHNOLOGIES

The comparative study undertaken by the Technology Assessment Panel will consider certain problem domains, or application areas, for which neural networks appear to offer some advantages. It is important to differentiate between several different types of neural network implementations – such as an algorithm running on a digital computer, a simulation of a neural network hardware design, an application-dependent hardware design implemented in digital, analog or optical technology – and a biological *wetware* implementation. For each application area considered, one or more neural network approaches for a particular application will be compared to the best known conventional approach to this application. The measures used to compare the different approaches are as follows:

- Accuracy (How well does each system do the job?)
- Response Time (How long does each system take to produce answers, responses?)
- Knowledge Acquisition/Embedding (How long does it take to acquire the knowledge to design, build, adjust, and test the different systems?)

- Reliability (How robust are the different systems to changes in the environment and/or the hardware implementation?)
- Physical Characteristics (size, weight, power consumption, etc.)

At this stage in the development of neural networks, it is not possible to assess reliability in the sense intended above because there are so few systems actually implemented in hardware. Furthermore, at this point the robustness of various processing algorithms to element defects and failures has not been studied theoretically in any detail. Therefore, assessment of reliability would amount to assessing the reliability of the various digital computers used to run neural network algorithms.

A similar point should be made relative to physical characteristics. This Panel is not interested in assessing the compactness of the various digital computer designs used to run neural network algorithms. The Panel will, therefore, limit its comments about physical characteristics to pointing out those applications for which compact special-purpose hardware has been demonstrated or is close at hand. Reliability will not be discussed at all.

Another item of interest is the *cost* of a fielded system. Since there are few neural network systems that have been manufactured, and many of the examples to be discussed are far from manufacture, cost can only be estimated very roughly on the basis of the size and complexity of various implementations of neural networks, and will thus not be a prime measure in the following comparisons.

3. COMPARISON OF NEURAL NETWORKS WITH OTHER TECHNOLOGIES

3.1 NEURAL NETWORK APPLICATIONS USED IN THE COMPARISON STUDY

Neural networks have been applied to a fairly wide variety of information processing problems, with those applications related to processing sensor data being the most common. This may be due to the fact that there is more biological inspiration here, since sensory systems are perhaps the best understood parts of biological nervous systems. The Technology Assessment Panel's set of neural network applications for study and comparison thus include many in the areas of pattern recognition, signal processing, machine vision, and speech. In addition, motor control applications from robotics are also included for their novel and generally simplifying approach to manipulator control.

The most mathematical applications are in the area of optimization theory, where models from physics have been found to be useful in finding good approximate solutions to problems involving minimization of a cost function under a set of constraints. Rarely have neural networks been used to solve problems (of any complexity) requiring reasoning and inference or complex mathematical calculations, such as testing very large numbers for primality.

In one sense, a comparison study is easy since any example applications that are demonstrably better than conventional techniques will establish the value of the new technology. The difficulty is finding examples that have been studied thoroughly enough to provide the data that establishes relative performance clearly and objectively – especially in a field where workers are still trying to prove the worth of their work.

One view holds that if the new technology were clearly superior to conventional approaches, then the workers would go to great pains to generate the data which would unequivocally demonstrate their claims. Such papers are very difficult to find in the field of neural networks today. In fact, the most difficult part of this comparison study has been finding papers and reports with enough data to allow quantitative comparison with the competing approaches. This has limited the Panel's choices of examples to present here, but in a sense, it puts the current state of the neural network field accurately in context. There are few examples available today that are treated in enough detail to make an accurate assessment of the technology with respect to conventional approaches.

The neural network applications used in this comparative study are listed in Table 3-1. Each of the neural network applications noted in this Table will be compared with alternative, more conventional approaches to the same problem.

- OPTIMIZATION PROBLEMS
 1. Traveling salesman problem (TSP)
 2. Associative memories
- PATTERN RECOGNITION/CLASSIFICATION
 1. Tactical ground target recognition using laser radar imagery
 2. Sonar discrimination problem
 3. Smart weapon application
- ROBOTIC CONTROL APPLICATIONS
 1. Learning nonlinear mappings for manipulator control (CMAC)
- SIGNAL PROCESSING
 1. Estimation of the shape of an unknown waveform corrupted by noise
- SPEECH APPLICATIONS
 1. Text-to-phoneme translation for speech synthesis
- MACHINE VISION APPLICATIONS
 1. Early vision: image segmentation
 2. Learning a color model from examples
 3. Visual motion processing

Table 3-1.

Neural Network Applications Used in the Comparison Study

3.2 THE TRAVELING SALESMAN PROBLEM

The traveling salesman problem (TSP) is an easily stated but difficult-to-solve optimization problem that has received considerable attention from computer scientists and operations research specialists.

The problem: Given a list of cities, and the distances between each pair of cities, find the minimum-length tour that visits each city exactly once. A tour is simply an ordered list of the set of cities, specifying the order in which the cities are to be visited so that total distance traveled is minimized. The problem of finding the minimum-distance tour is an NP-complete problem, that is, it has been established that the number of steps in any solution must have a non-polynomial (i.e., exponential) dependence on the size of the problem, i.e., the number of cities [20].

In a paper by Hopfield and Tank published in 1986 [8], it was shown that the TSP could be stated as an energy minimization problem, and, as such, that a Hopfield neural network could be used to solve it. It was also implied that such neural networks might be valuable in solving other difficult optimization problems. The solutions generated by the neural network were not optimum – i.e., not *the* minimum-distance solutions, but close approximations – and were only for problems with up to 30 cities, which in this problem context is a very small problem size.

Since it is worthwhile to put this application of neural networks in perspective, note that computer scientists and mathematicians had generated many different algorithms for the minimum-distance and for approximate solutions to the TSP. In a recent paper [11], David Johnson of AT&T Bell Laboratories compared the Hopfield and Tank results with an algorithm developed by Lin and Kernighan, termed here the L-K algorithm [14]. The L-K algorithm could run the TSP for up to 50,000 cities and get within 2% of the minimum-distance tour. The Hopfield network only ran up to 30 cities and never got within 17% of the optimum solution. The L-K algorithm is also fast, taking less than one second on a VAX 8550 for the 30-city problem. The VAX 8550 time for the Hopfield network is not available (perhaps because it was run on a different computer using a different programming language, which would make the results incomparable). A 400-city TSP took 20 seconds with L-K, while a “simulated annealing” algorithm on this problem took over six hours. The Hopfield network has not solved a TSP this large.

These results clearly show that the Hopfield network running on a digital computer is not competitive as an optimization algorithm for the TSP. However, if the Hopfield network were implemented on a VLSI chip instead of a digital computer, the chip implementation might be able to find solutions very quickly. If the circuit elements could respond in less than a microsecond, then the hardware implementation might settle to a solution in only a few tens or hundreds of microseconds. So if an autonomous vehicle like a planetary explorer or a submersible explorer needed to solve path optimization problems of similar complexity to the TSP in very short time, then a neural network implemented in VLSI might be the best way of realizing this capability.

Wilson and Pawley of the University of Edinburgh were interested in exploring just this possibility, and made detailed studies of the Hopfield network solutions to the TSP [36]. In the course of doing many computer simulations, they could not duplicate the published results of Hopfield and Tank even on a 10-city TSP. In their detailed investigations, Wilson and Pawley discovered basic problems with the algorithm. They found, in fact, that the Hopfield network got legal solutions (i.e., tours that visited each city exactly once) only 8% of the time with only 10 cities. They concluded that “Hopfield and Tank

were very fortunate in the limited number of TSP simulations they attempted. Even at the value $N=10$, it transpires that their basic method is unreliable and does not offer much hope for improvement.”

In conclusion, one of the early and exciting accomplishments of a neural network was to (supposedly) have solved an NP-complete problem. Careful review of the work revealed that the Hopfield network did not produce optimum solutions to the TSP at all, but only approximate ones – and poor approximations at that – compared to standard computer optimization algorithms like Lin-Kernighan. The Hopfield network also was limited to solving problems of uninterestingly small size; moreover, it was shown to be generally unreliable in producing solutions that satisfied the constraints. It appears unworthy at this time to pursue this algorithm to a hardware implementation, which might have produced faster response times that might be of use in certain applications. The initial excitement was not justified in this case, as neural networks have contributed little to the solution of the TSP. This cannot be taken to imply that neural networks will never be useful in providing approximate solutions to any mathematical optimization problem, but at present, this is not an area to which neural networks contribute very much.

Comparison Assessment Summary: The Traveling Salesman Problem

Application: Approximate solutions to the TSP using a Hopfield network.

Accuracy: Hopfield network approximations are not as accurate as conventional techniques, and very limited in the size of the problem that can be handled.

Response Time: Not directly compared on the same computer in the published results.

Knowledge Acquisition/Representation: Connection strengths are distances input in advance.

Projection: Cannot judge success on other types of complex optimization problems at this time.

3.3 ASSOCIATIVE MEMORIES

One of the first applications of neural networks was in associative memories. Consider a simple associative memory which has stored in it M binary vectors of length N , often called memories. Given a binary input vector \mathbf{u} of length N , the function of the associative memory is to produce as its output the stored memory vector with the smallest distance to the input \mathbf{u} . The distance measure used most commonly for binary vectors is the Hamming distance – i.e., the number of places in which two binary vectors differ. For vectors of real numbers, the distance metric is usually Euclidian distance. In some associative memories, the values of the input vector components can have three values, 1, 0, and *, where * is a “wild card” symbol denoting that the Hamming distance computation should exclude the components where the input has a *. Or the numbers can even be likelihoods, real numbers between 0 and 1. This capability allows one to specify which parts of the input vector are to be matched by the associative memory lookup operation, which parts may be ignored, or which parts are more reliable.

The problem: Given a specified distance function or metric for pairs of vectors, and a set of memories or code vectors, find the memory or code vector with the minimum distance to an input vector. The metric

may be changed for different inputs, as in the case of wild card symbols. Also, the associative memory may be required to list all code vectors within a prescribed distance from the input vector.

The problem of decoding a binary error-correcting code used on a noisy channel can be stated to correspond exactly to the function of the associative memory. One of M code words (binary sequences) of length N is transmitted over a noisy channel and the receiver produces a noisy binary vector corresponding to the received word. The decoder now must find which of the M code words is closest to the received word in the Hamming distance sense, which corresponds to the maximum *a posteriori* probability decoding decision for a certain class of noisy discrete binary channels. The Euclidian distance metric applies to decoding for a channel with continuous additive white Gaussian noise.

A similar, but not identical, problem that the associative memory function can do is encoding of the output of an information source to compress the data that must be transmitted over a communication channel. Consider a source which produces binary digits that are to be reproduced at a receiver only to within some average number of errors. That is, transmitting all of the source digits to the receiver without error is *not* required. This relaxed fidelity criterion allows the source data to be compressed as follows: An encoder with $M < 2^N$ binary code words of length N maps an input vector (of length N) from the source into the nearest code word, nearest being used in this case in the Hamming distance sense. There would be fewer than all possible source output sequences or vectors in the code set of M words, and this implies that, at most, only M of the source output sequences could be matched exactly by a word in the code set, with some number of errors being introduced by transmitting the nearest code word rather than the exact source output. Once again, an associative memory as defined above could perform the source encoding function. This is often called “vector quantization.”

Applications for associative memories can be foreseen in data compression, as described above, in the near term, but according to John Moody, who was a member of Panel 2, *the Adaptive Knowledge Processing Panel* of this Study, “The long-term applications of associative memories, as pure memories, have yet to be defined in any detail.” They do appear to be relevant to the problem of searching large databases, and to searching large sets of rules in AI expert systems. Given that associative memories have been studied since the early 1970s, with no significant practical application having yet materialized, their importance may be limited.

A digital system could perform the associative memory task with a bank of M storage registers, each N bits long, and a means of computing the distance between an input sequence and the contents of each memory register. The memory would require $M \cdot N$ bits. No fewer bits could perform the task of finding the best match in M words, so the ideal measure of hardware efficiency E could be taken as the number of bits in the memories over the total number of storage elements required to implement the system. In the case of the digital system,

$$E = (M \cdot N)/(M \cdot N) = 1$$

Hopfield’s first paper on neural networks treated an associative memory example. There were problems in accuracy in that the Hopfield associative memory did not always produce as its output the nearest stored memory to the input vector. But its capacity was shown to be limited to $M < N/(4 \log N)$ if perfect

reproduction was desired while it had to store N^2 bits. Its efficiency was thus

$$E = \frac{1}{4 \log N}.$$

A new associative memory, called the unary or Hamming network, has been designed by John Moody, *et al.* [4], and is described in *Part II, Chapter 5* of this Report. This associative memory can take an input, with wild card symbols interspersed within it, and produce the nearest memory stored in it using only $M \cdot N$ bits of storage, giving the ideal efficiency of unity. This design also appears to be compatible with VLSI implementation. A schematic diagram of the Hamming network is shown in Figure 3-1. In *Part VI* of this Report, a Hamming network on a VLSI chip, which was designed and fabricated by AT&T Bell Laboratories, is described. This chip computes the Hamming distance between a binary input vector and 46 vectors stored on the chip in less than 100 nanoseconds. The vectors are 46 bits long.

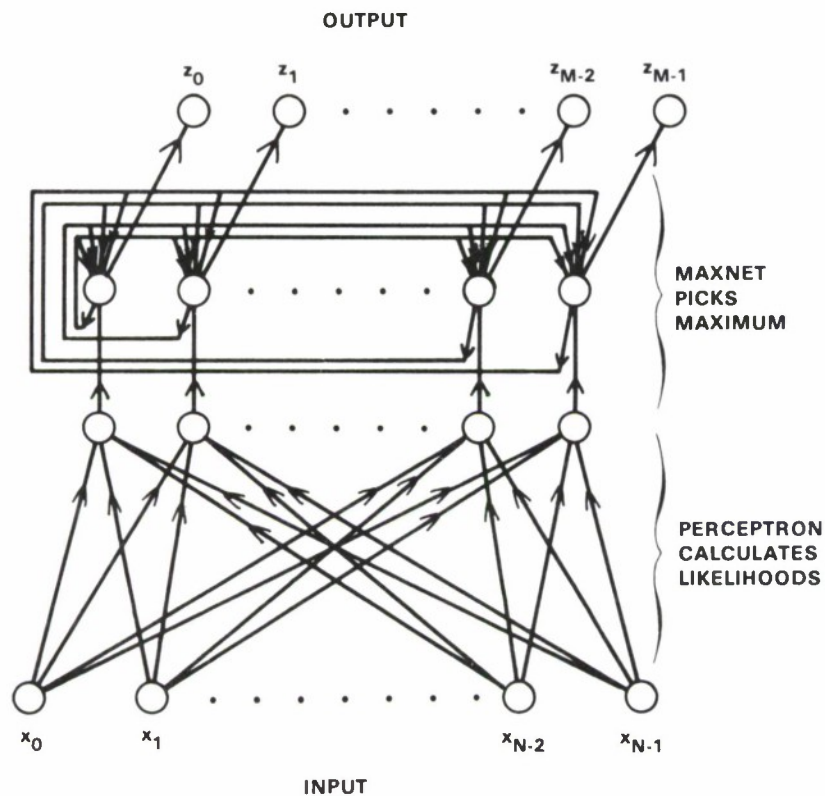


Figure 3-1. Hamming Network.

A more general form of the associative memory problem has been addressed in which the designer would like to assure that a certain neighborhood of inputs be mapped into a prescribed memory vector. The designer would like to be able to specify a “domain of attraction” for each memory vector. This requirement amounts to building an associative memory with a rather arbitrary mapping function from inputs to memories. A simple metric may not suffice for this problem. Very little work has been done on this problem and there is, at present, no general way to solve it.

As an aside, the channel decoder device, discussed earlier in the context of communication theory, was intended for use with very long block codes. For instance, a block code of length $N=128$ bits would not be unusual. The rate of a block code is defined as $\log M/N$, where M is the number of code words. A rate $1/4$ code would thus have rate $\log M/128 = 1/4$, or $\log M = 32$, and this implies $M = 2^{32} \approx 4 \cdot 10^9$. The total storage required by the code would thus be

$$M \cdot N = 2^{32} \cdot 2^7 = 2^{39} \approx 0.5 \cdot 10^{12}.$$

This was deemed impractical for a codebook-type of decoder, which would simply store the entire code and check the Hamming distance of an input to each codeword. It is still impractical even using the AT&T Hamming network chip, since many millions of chips would be needed to decode such a long code.

Associative memories do not provide a new solution to the noisy channel decoding problem or the source encoding problem. They simply propose a method of doing the computations in parallel, if it is ever feasible to wire up such a large amount of storage on chips or wafers. This is another example where neural networks were applied to a problem that had been studied years earlier, and the proposed solution was based on brute force parallelism, which communication theorists considered impractical in earlier years. There is no contribution to communication theory here, and it remains to be seen if VLSI implementations can be realized that will provide the required parallel system in a practical size.

The approach taken by communication theorists in the 1960s and '70s was to look for "systematic" codes that could be generated from many fewer bits than $M \cdot N$, using an algorithm, and proving that such systematic codes could provide good performance. Another point that was attacked, with notable success, was the development of decoding schemes that could make decoding decisions with many fewer computations than $M \cdot N$ to determine the nearest codeword. Parity check codes and convolutional codes which are in practical use today are the results of these early efforts in communication theory. The simple exhaustive search of a codebook was considered an impractical solution to decoding long codes for performance approaching theoretical ideal.

Recovery of stored information by means of associations with related data is a problem of great interest in AI. The relationships of interest in AI are very arbitrary and general ones, such as other objects with similar characteristics. This associative recall problem was addressed in AI by Quillian [24] using what he termed "semantic networks," which amount to graph-like data structures. The nodes of the graph denote objects or parts of objects, and the edges or connections between nodes represent relationships of various types. Semantic networks can be used to form hierarchical data structures that can include the inheritance of properties among related objects. An important drawback of neural network associative memories to date has been the simplicity of their structure, which limits them from being applied to the more general problems of storing and retrieving information, such as semantic networks.

In summary, the associative memory problem has been precisely stated as a minimum-distance mapping or decoding problem with, in its most general form, a specified distance metric. The newest formulations of neural network associative memories are efficient in terms of the number of elements to store and recover arbitrary sets of memories (code words). The minimum distance decoding problem is a key problem of communication theory that has received a great deal of attention over the last three decades. The complexity of the problem for long codes has led to the development of systematic codes to reduce the number of elements in the decoders. The neural network approach does not appear likely to offer practical new

solutions to the decoding problem in communication engineering. The storage and retrieval of information on the basis of complex relations like inheritance in a hierarchical data structure has been studied in AI for some time. The present associative memory models are too simple to handle this problem. Other specific applications of associative memories have not yet been defined, and so constraints on code sets and the form of the applicable distance metrics are not available. Thus while hardware implementations of associative memories do offer the potential for speed, their utility in practical applications will have to await more detailed evaluation in some specific applications.

Comparison Assessment Summary: Associative Memories

Application: Associative memories for various uses, including vector quantization.

Accuracy: Hamming networks perform adequately for simple metrics.

Response Time: Custom VLSI chips can provide very rapid response times for small sets of memories, but probably not fast enough or large enough for noisy channel decoding.

Knowledge Acquisition/Representation: Learning sometimes used, but specified set of memories usually handled.

Physical Characteristics: Custom VLSI for small sets of memories (see *Part VI* of this Report).

Projection: Not possible until more specific application evaluations appear. In present form, will not be useful as noisy channel decoders, or in implementing semantic networks.

3.4 PATTERN CLASSIFICATION APPLICATIONS

The application of neural networks to pattern classification has perhaps the best potential for practical systems. Pattern classifiers are mappings that define partitionings of feature space into regions corresponding to class membership. A theorem by Kolmogorov states that an arbitrary nonlinear mapping can be expressed in a form that can be implemented by a multi-layer neural network with, at most, three layers. Lippmann has shown that many standard pattern classification algorithms can be implemented using current neural network architectures [15]. New pattern classification algorithms, such as the “reduced Coulomb energy (RCE)” networks of Nestor, Inc. [25] have also emerged from the work in neural networks. This trend will most likely continue, since neural networks do seem to be well-suited to applications of trainable pattern classifiers.

The application of neural networks to three different pattern classification problems is considered in this section:

- Target recognition using imagery data,
- Target discrimination using sonar signatures, and
- A pattern classifier for smart weapon application.

3.4.1 Tactical Target Recognition Using Laser Radar Imagery

Captain Dennis Ruck, USAF, in his Ph.D. thesis (at the Air Force Institute of Technology (AFIT), Dayton, OH) [26], attempted a careful comparison of a standard statistical pattern classification technique with a multi-layer neural network for recognizing tactical targets using imagery produced by a laser radar. Ruck's work (supervised by Matthew Kabrisky) addresses the problem of automatically detecting and recognizing such tactical targets as M60 tanks, POL tankers, jeeps, and 1.25- and 2.5-ton trucks. Two different approaches to the classification problem were taken:

1. A nearest-neighbor classifier algorithm, and
2. Use of a multi-layer perceptron with the backpropagation learning algorithm.

The feature set used in all of the approaches was either the Zernicke moment invariants (a total of 22 used) – which are invariant to position, scale, and rotation of the target object in the image plane – or a reduced feature set derived from the moment invariants using the method of Fisher linear discriminants. A database of laser radar images available at Air Force Wright Aeronautical Lab (AFWAL) was used in this work.

The laser radar produced range and Doppler data – i.e., for each pixel for which a radar return was detected, a value of range and Doppler was provided by the laser radar. The recognition system included the following sequence of processing steps:

1. Image segmentation: Partitioning the image into disjoint regions on the basis of relative range and Doppler;
2. Target region detection: Selection of “interesting” regions on which to attempt recognition;
3. Feature extraction: Calculation of the Zernicke moment invariants;
4. Reduction of the feature set dimensionality: Using Fisher linear discriminants; and
5. Target recognition or classification: Using either a nearest-neighbor classifier or a multi-layer perceptron.

The training set used in these experiments had only 57 images of four different vehicle types, which included 47 tanks, six jeeps, four POLs, four 2.5-ton trucks, and no 1.25-ton trucks. The training set was further subdivided into 11 different target classes, considering each different viewing aspect for each target as a separate class. The nearest-neighbor classifier used a set of 10 features derived from the Zernicke moment invariants.

It correctly classified 76.5% of the tanks in the test set. It did poorly on the other types of vehicles, perhaps because they were under-represented in the database.

The multi-layer neural network was quite simple to implement, and was trained using the backpropagation learning algorithm. Classification experiments were run with both the raw Zernicke moment

invariants as features and with the Fisher reduced set. For the 22 raw Zernicke moment invariants, the number of neurons in the first and second layers were equal to 200 and 60, respectively. For the Fisher reduced feature set of four features, these numbers were 100 and 30, respectively. The number of output units was equal to the number of classes, which were considered to be just four in these experiments, since only four different vehicles appeared in the training set. The total number of training iterations to train the neural network were 26,650 for the 22-dimensional feature vector and 3,110 for the four-dimensional input. The multi-layer perceptron correctly recognized 86.4% of the tanks in the test set when using the 22-dimensional raw feature set, and 59.1% using the reduced four-dimensional feature set.

In summary, the work by Ruck was a serious attempt to quantify the relative performance of a standard nearest-neighbor pattern classifier and a multi-layer perceptron in a target recognition application.

The database was not adequate for a careful measurement of the separability of all of the target classes. Even with this qualification, the performance results showed the neural network approach to be competitive with the nearest-neighbor classifier. The performance of the nearest-neighbor classifier should have been evaluated using the 22-dimensional raw features, for fair comparison with the neural network which achieved its best performance with this set (on tank recognition).

Another aspect of interest in the comparison of these two methods is that of knowledge acquisition – i.e., deriving information from the training set to set up the algorithms. In the case of the nearest-neighbor classifier, the decision rules must be determined from the training set, which, under a Gaussian noise assumption, simply involves the estimation of a mean vector for each class. In the case of the neural network, the knowledge acquisition process involves training the network using the backpropagation learning algorithm. The number of training iterations given above indicates that the training is more time-consuming than setting up the classifier decision rules.

It appears that the neural network approach compared favorably in performance (tank recognition) to the standard pattern classifier approach, but was more time-consuming to set up. There was no surprising discovery in this work of a radically improved approach to target recognition, although Ruck's work is somewhat encouraging with regard to further comparison studies of this type. The work is valuable in that it does a careful comparison of two approaches using the same database, and it does indicate that such a careful comparison takes considerably more effort than merely demonstrating that a neural network can do the recognition task, but without any reference points to put the neural network performance in perspective with respect to existing techniques.

3.4.2 Discrimination Between Two Different Sonar Targets

Gorman and Sejnowski [6] have applied a multi-layer neural network to the problem of discriminating between sonar returns from two different targets on a sandy ocean bottom – a metal cylinder and a rock, both about five feet in extent. The objective of the study was to compare the neural network approach to the performance of a standard nearest-neighbor pattern classifier and to that of humans, and to try to correlate the hidden layer units to features that were useful to the humans. The database used was comprised of a selected set of 208 recorded sonar returns (selected from a larger database of 1,200 returns on the basis of signal-to-noise ratio), and included returns from different aspect angles for both targets. The experiment

tested the ability of the various approaches to discriminate between the two targets on the basis of only a single sonar return.

The transmitted sonar waveform was a linear frequency modulated (FM) chirp signal. The sampled returns were filtered using a bank of 60 bandpass filters. The feature set used in these experiments consisted of spectral estimates of the returns made by smoothing the outputs of these filters. The spectral estimates were staggered in time to correspond to the FM chirp of the transmitted waveform. A three-layer neural network with continuous valued units was used, with the input layer of 60 units clamped to sample values of the sonar signal. The number of units in the hidden layer was varied from 0 to 24 in a series of experiments. The two output units were used to represent the two possible targets. The network was trained using the backpropagation learning algorithm and 300 presentations of the entire training set were used for each experiment. The test sets were always different from the training sets. In one set of experiments, no attention was paid to the aspect angle of the targets, picking samples for the training and test sets at random. In another experiment, care was taken to include a sampling of all aspect angles for both targets in both the training and test sets. For each experiment, the network was trained 10 times with different (random) initial values used for the weights each time to try to average out the effects of the initial state of the network.

The nearest-neighbor classifier used a set of labeled returns to which the feature vector of an unknown input was compared in the Euclidian distance sense. If the two nearest neighbors of the stored set belonged to the same target, then the classifier announced that target as the decision on the input. The stored data set used 104 returns in this algorithm, so the distance computations were considerable compared to the multi-layer network. A useful reference point is provided by a result of Cover and Hart that states that the asymptotic error rate of a nearest-neighbor classifier is lower-bounded by that of an optimum Gaussian classifier, which can be no more than a factor of two lower than that of the nearest-neighbor classifier.

Experiments were done with human subjects that were trained on the sonar data. The sonar returns presented to the subjects were of longer duration than those used for the network and the nearest-neighbor classifier, and were also processed differently for the human listeners. Only three subjects were trained and tested in these experiments, with 100 of the total of 208 data samples in the database used in the training sessions. The subjects were required to announce a classification decision as each training sample was presented, with immediate feedback of the correct answer being provided.

For simplicity, only the results for the aspect-dependent experiments are discussed here. The neural network achieved correct classifications at around the 90% range for this series of experiments. This performance did not improve significantly for more than six units in the hidden layer. The nearest-neighbor classifier achieved 82.7% correct classifications using the same overall database. This implies that an optimum Gaussian classifier could achieve as high as 91.4% correct on this data. Only one human subject addressed the classification problem for the aspect-dependent data, achieving 88% correct results, but this is clearly too small a sample from which to draw conclusions.

In summary, it appears that the multi-layer network performed about as well as the small set of human subjects, although the experiments differed in detail. The nearest-neighbor classifier was not as good as either the neural network or the human subjects, although the optimum Gaussian classifier bound was quite close to the neural network performance. From the point of view of accuracy, there were no major breakthroughs in this problem.

From the point of view of knowledge acquisition, the training of the neural network involved considerable effort, but it could be automated (programmed) using a digital computer. The experiments with human subjects also involved considerable training effort, probably taking more time to present and record the subjects' decisions than that taken with the neural network. The nearest-neighbor classifier was more straightforward to set up, once programmed, but the computation time to classify each input was significant, and would be an important consideration in fielding an operational system, since it would dictate the size of the computer needed.

Both the neural network and the nearest-neighbor classifier could be implemented in special-purpose hardware to realize a compact system with fast response time. This particular problem was not of great interest in this regard because of the limited scope of the experiments. A practical system would have to deal with a greater variety of target types and ocean bottoms, as well as other acoustic propagation effects in water that could affect classification performance. The knowledge acquisition issue would be much more significant and the response of a nearest-neighbor classifier might also be an important issue. Special-purpose hardware implementations of a classifier might well be considered for a practical sonar target recognition system if adequate performance could first be demonstrated using a digital computer implementation.

It appears that it would not be unreasonable to consider a practical sonar target recognition system based on the use of multi-layer neural networks. One advantage of this approach is that it appears that formulating a detailed model of the targets, the sensor noise, and the distortions of the propagation medium might not be of primary importance in such an effort. In a manner very much like that used in applying statistical pattern classification techniques, a classifier algorithm is programmed on a digital computer and tested on a database. Unlike the pattern classifier, the neural network may require considerable effort to train the system. But the neural network then would be easier to implement in special-purpose hardware than something like the nearest-neighbor classifier, which might require a digital computer for accuracy and considerable computation to classify inputs. A significant development effort would be required to establish the feasibility of the neural network approach to a practical sonar target recognition/classification system, but the work of Gorman and Sejnowski is not at all discouraging.

3.4.3 Smart Weapons Applications

The concept of "smart" weapons is to transfer more of the problem-solving capability from human operators to the machines themselves in the hope of getting the job done effectively but with much less risk to the human operators. Examples of such applications are abundant. Autonomous homing weapons, the so-called "fire and forget" weapons, are strongly preferred over weapons that require constant monitoring and guidance all the way to the target by the operator. Of course, this type weapon puts more of the functionality in the weapon itself, which means more equipment in a small space, a high "G" environment, and a package which is not going to return. The basic issue is whether or not a good guidance package can be designed to give adequate performance at all in the presence of real-world conditions like noisy sensor data, variable target characteristics and, perhaps most critical, enemy countermeasures designed to deliberately try to foil the guidance system. If a guidance system can be designed to cope with all these effects and still do the job adequately, then one faces the challenge of building an expendable package to fit the size, weight and power constraints.

One easy-to-understand function that is essential in many smart weapons is that of determining the exact instant of detonation of the warhead of the weapon to assure maximum damage to the target. Some missiles in the past were completely controlled from the ground, including this detonation function. However, in fire and forget weapons, a device on board the weapon, called the proximity fuze, performs this function. The design of a proximity fuze for an anti-aircraft missile is a challenging problem when one considers noisy sensor data, variations of target signature with different aspect or approach angles, and countermeasures.

There are a number of technical alternatives to this problem, such as a deterministic decision tree, a rule-based expert system, a conventional pattern classifier, or even a neural network. Whatever approach is chosen, if it can be made to work, perhaps by demonstrating performance in simulation studies, then the hardware implementation must also be considered in evaluating its feasibility. Rule-based systems run on digital computers, usually in high-level languages which require large memories and lots of processing to execute. Implementations based on the use of general-purpose digital computers are not feasible for this application. Some of the approaches require complete analysis of the problem to formulate a solution, which can then be implemented in special-purpose hardware that might meet the physical constraints of the vehicle, especially if custom VLSI were employed. However, an important consideration is the changeable nature of the problem, as new target types are deployed and new countermeasures are developed. A customized, special-purpose circuit might soon be obsolete.

A trainable system appears to have important advantages for this problem. One should thus be interested in trainable pattern classifiers or neural networks for the proximity fuze function. Many pattern classification schemes can be implemented in the form of neural networks, so these two approaches are quite similar. One study done in a government laboratory pursued this chain of reasoning and experimented with a simple multi-layer neural network for this function and found that, for a particular missile application, the network could be trained to do this job adequately. Moreover, it also appeared that the network could be implemented with analog neurons that could fit on a single VLSI chip. It remains to design the chip, fabricate it, and test it in detail to establish the feasibility of the neural network approach for this application. Given that the neural network is small, the size of the chip may not be a problem. The feasibility of this approach may be more critically dependent on whether the circuit technology will allow adaptation of the network weights to compensate for optimization of algorithm performance as well as for element inaccuracies. The experimental work on implementing neural networks in VLSI chips discussed in *Part VI* of this Report is relevant to this point.

If this approach can be made to work, then chips would be fabricated and trained using a standard data base, before being installed in equipment on board a missile. When new target types appear or new countermeasures were to be included, the data base would be augmented and used to train the same type of chips for the expanded problem. The size and processing speed of the neural network chip offers perhaps the best chance of meeting the physical constraints of the smart weapon problem. The production cost of custom neural network chips is difficult to estimate at present, since new circuit technology may be involved and production runs may not be very large.

This approach would seem to be appropriate for other smart weapon functions as well. The output of an on-board sensor might be processed using neural networks to recognize the target, to estimate target parameters, and to perform the tracking function for an evasive target. It may be feasible some day to have

such a target tracker driving the missile control system which uses a neural network to perform efficient, coordinated maneuvers for pursuit of an evasive target. A number of such simple functions appear to be very useful in the context of smart weapons. If such functions could be done in relatively small neural network modules that were implemented in custom chips, then there would be a reasonable chance of meeting the stringent constraints on hardware imposed in such applications as smart weapons and other autonomous vehicles and robots.

3.4.4 Summary of Pattern Classification Applications

Neural networks have demonstrated good classification accuracy when carefully compared to standard classifiers, but not significantly better performance in this regard – although this cannot be ruled out in future work. As trainable systems, they have dealt effectively with the knowledge acquisition/embedding problem. For some applications – such as “smart” weapons (e.g., the proximity fuze problem) – implementation in VLSI of simply-structured neural networks is not far off, and the physical constraints of this application make neural networks the only known way of compactly implementing a classification algorithm that can do the job with adequate accuracy. It appears almost certain that neural networks will provide new solutions to important pattern classification problems, and this may occur in the near future – e.g., in the time it takes to design, fabricate and test a neural network chip with adaptive synapses. According to the work of the Neural Network Study’s *Advanced Implementation Technology Panel*, Panel 5, this may be within four years.

Comparison Assessment Summary: Pattern Classification Applications

Application: Pattern classification, trainable classifiers.

Accuracy: Comparable to standard pattern classification algorithms, but not significantly better.

Response Time: Custom VLSI chips offer fast response time for realtime system applications, faster than computer implementations.

Knowledge Acquisition/Representation: Depending on the learning algorithm used, training a neural network usually takes longer than training statistical pattern classifiers, but the knowledge acquisition problem is more straightforward than modeling and analysis for subsequent design of a fixed algorithm.

Physical Characteristics: Custom VLSI chips offer the smallest classifier realization for space-, power-, and weight-constrained applications.

Projection: Neural networks appear well-suited to pattern classification problems, allowing straightforward implementation of existing algorithms or new variations of classifier algorithms, with the possibility of compact, realtime hardware using VLSI implementations, for space-constrained applications such as autonomous vehicles and smart weapons.

3.5 COMPUTATIONAL MAPS FOR ROBOTIC CONTROL

One important problem in control theory is the design of a controller which uses the present state of a system and determines the optimal control inputs to a system of actuators to get the system to a desired new state. The standard approach to this problem is to model the elements of the system, do a mathematical analysis to determine the control law that produces the optimal actuator inputs, and implement this control law by doing realtime computation. Control of robot manipulators is an especially complex task because it usually involves many actuators which interact with each other, many degrees of freedom, and many ways to carry out a desired task; the challenge is to determine control inputs that result in “coordinated movements.” Manipulator control usually involves approximate models of physical devices, complex coordinate transformations, and tensor calculus.

A new approach to this problem was pioneered by James Albus in the 1970s, inspired by his Ph.D. studies of the role of the cerebellum in motor control in living organisms [3,2]. His approach, which he has termed the Cerebellar Model Articulation Controller (CMAC), is based on a neural network architecture which determines actuator control inputs by using what amounts to a lookup table with feedback variables from a manipulator. In CMAC, the values of the desired control functions are learned from training examples, stored and used with an interpolation scheme that allows a degree of generalization to handle situations similar, but not exactly the same as, the training examples. It is essentially a trade of computational complexity for storage, which is advantageous given the availability of inexpensive storage devices.

In the CMAC approach, there is no need for detailed mathematical modeling, analysis or complex realtime computations to realize a robot controller. An arbitrary mapping function from the present and desired states of the manipulator to the actuator control inputs can be learned from training examples. Many feedback variables from an array of sensors can be accommodated with only a linear growth of required memory with the number of variables. The resolution of the mapping function can be chosen by the designer for the required degree of accuracy of control. The feedback sensors need not be linear or simple – only repeatable. These are obviously advantages that would be very important in biological control problems. It appears that this is the style of control imposed by the cerebellum of animals.

An interesting application of the CMAC concept to the control of a robotic fork lift is currently being carried out by Jennings at Martin Marietta [10] (*Part IV* of this Report includes a brief description of this application written by the investigator). In an experimental system, a human operator can control the fork lift to align it with a pallet and pick it up, while a number of optical and acoustic “radar-like” sensors on the tines of the fork provide data on the relative position of the fork and the bottom of the pallet. The CMAC system thus receives the sensor inputs together with the human operator’s control inputs, which can be used to learn the mapping between the two to eventually control the system without the human operator. This system is in an experimental stage at present and various learning strategies are being tried. No quantitative performance comparison can be made with a conventional robotic control approach to this problem, but the CMAC approach clearly offers several advantages which are important to this class of problems.

In another study carried out at the University of New Hampshire by Miller, Glanz and Kraft, a torque controller was realized by using a CMAC module in place of a dynamic model of the robot [18]. In this

simulation study, a controller was trained to follow desired trajectories by providing it with a sequence of positions along the desired trajectories. Experiments were carried out with various CMAC memory sizes, and performance was found to be relatively insensitive to memory size. Although analytical evaluation of performance is difficult for this type of control system, it does appear to be straightforward to apply the CMAC concept to such a control problem, it needs only relatively inexpensive hardware to provide better performance than fixed gain controllers, and it is fairly insensitive to changes in the system characteristics.

In summary, the use of computational maps has several important advantages in robotic control applications, and will probably find application in practical systems even at their present state of development. This approach is not appropriate for all control problems, of course. One area that would increase the utility of this approach is manipulator motion planning with constraints on the trajectory due to obstructing objects and difficult gripping surfaces. This more complex problem has just begun to receive some attention.

Comparison Assessment Summary: Computational Maps For Robotic Control

Application: Learning mappings for simple robotic manipulator control.

Accuracy: The CMAC approach can provide accuracy comparable to conventional control approaches.

Response Time: Response time is adequate for realtime applications, with less computational power than conventional approaches.

Knowledge Acquisition/Representation: Learning computational maps is simpler than modeling control system components and doing system analysis and optimization, at least for simple control problems.

Physical Characteristics: Currently available memories allow fairly compact realizations of practical computational maps that compare favorably with conventional approaches.

Projection: Computational maps will probably find practical application in the immediate future in simple manipulator control systems. More complex control requiring planning and obstacle avoidance will require further research.

3.6 SIGNAL PROCESSING APPLICATIONS OF NEURAL NETWORKS

3.6.1 Recovery of Noise-Corrupted or Distorted Waveforms

A classical signal processing problem is that of recovering an analog signal after transmission over a noisy or dispersive channel. In many cases, there may very little knowledge about the statistical characteristics of the signal, noise, or dispersion. Some examples are:

- Dispersion over telephone channels, time-invariant or very slowly varying;
- Noisy sensor data, such as electrocardiograms, where the shape of the waveform of interest varies slowly with time and is corrupted by measurement noise.

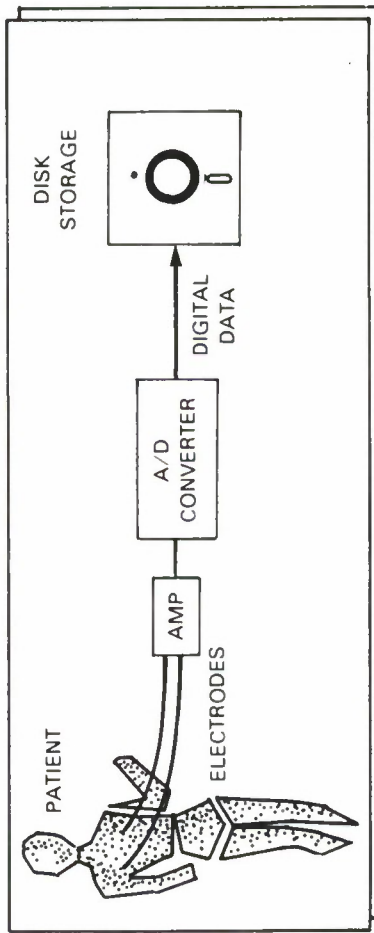
The standard approaches to this problem include filtering for noise reduction and channel equalization to reduce the dispersion, and the application of estimation theory to form an optimal estimate of the desired waveform. The lack of knowledge of relevant statistical characteristics hinders the use of estimation theory, in that data gathering and analysis may be first needed to formulate the requisite models, which could then be used to find the optimal signal estimator.

In some cases, it may be possible to specify the desired processor output for a particular noisy or distorted input. For instance, in the case of the dispersive telephone channel, a known signal can be sent by prior arrangement, so the receiver has both the distorted channel output and the desired output waveform. Then these waveforms can be used to “train” an adaptive filter – i.e., solve the optimal filter problem in real time by adjusting the filter parameters to give the desired output waveform. There is a version which does not use a training waveform but minimizes dispersion at sampling times. This approach to equalization of dispersive telephone channels has been highly successful, and the so-called “adaptive equalizer” is now the standard means of reducing dispersion on these channels [35]. Each adaptive equalizer is a linear finite impulse response (FIR) filter which operates on a window of successive samples of the channel output, and the output is simply a weighted sum of the input samples. An adaptive algorithm, usually the Widrow-Hoff Least Mean Square (LMS) algorithm, is used to adjust the weights to provide the desired (undistorted) channel output [34]. The adaptive equalizer is really a single, linear “neuron,” i.e., without the usual sigmoidal nonlinearity. There are probably many thousands of such adaptive equalizers currently operating in digital telephone transmission systems throughout the country, demonstrating the utility and practical value of adaptive filtering techniques for certain types of problems.

Adaptive filters are also used to recover waveforms from noisy channels. Douglas Palmer of Hecht-Nielsen Neurocomputer, Inc. has attempted to compare the filtering capabilities of a multi-layer neural network with an adaptive linear filter using the LMS algorithm [19]. An artificial problem was created in that essentially noise-free EKG samples were available, and noisy channel outputs were simulated for use in training by adding random noise to these noise-free samples. Both the noise-free and noise-corrupted waveforms were then used to adapt a linear FIR filter and also to train a three-layer neural network using the backpropagation learning algorithm. The neural network included a linearly weighted combination of the samples in the input window as well as the output of a multi-layer network with neurons that had a sigmoidal nonlinearity. The neural network system thus included the linear filter as well as nonlinear terms (see Figure 3-2). The hope was that the nonlinear terms would allow the neural network to mimic the desired output waveform more accurately than the linear filter alone. The graphs of output waveforms indeed show that the neural network produces waveforms that appear less noisy than the linear adaptive filter, but quantitative measurements of output signal-to-noise ratios were not provided in Palmer’s paper.

In an attempt to quantify the advantage of the neural network approach relative to the adaptive linear filter in this problem, Widrow and his student, Rodney Winter, did a computer simulation of a linear least squares adaptive FIR filter and an FIR filter that included nonlinear terms as well [33]. The nonlinear terms were simply the squared values of some of the samples in the input window. This was not at all the same as the multi-layer neural network used by Palmer. But it provided an alternative set of quantitative data on nonlinear adaptive filtering. Since Palmer’s EKG data was not available to Widrow, Winter used ideal triangular pulses as the signal and added noise from a random number generator to produce “noisy” data. The output signal-to-noise ratios of the linear and the nonlinear adaptive systems were then estimated.

DATA GATHERING



DISPLAY OF DATA

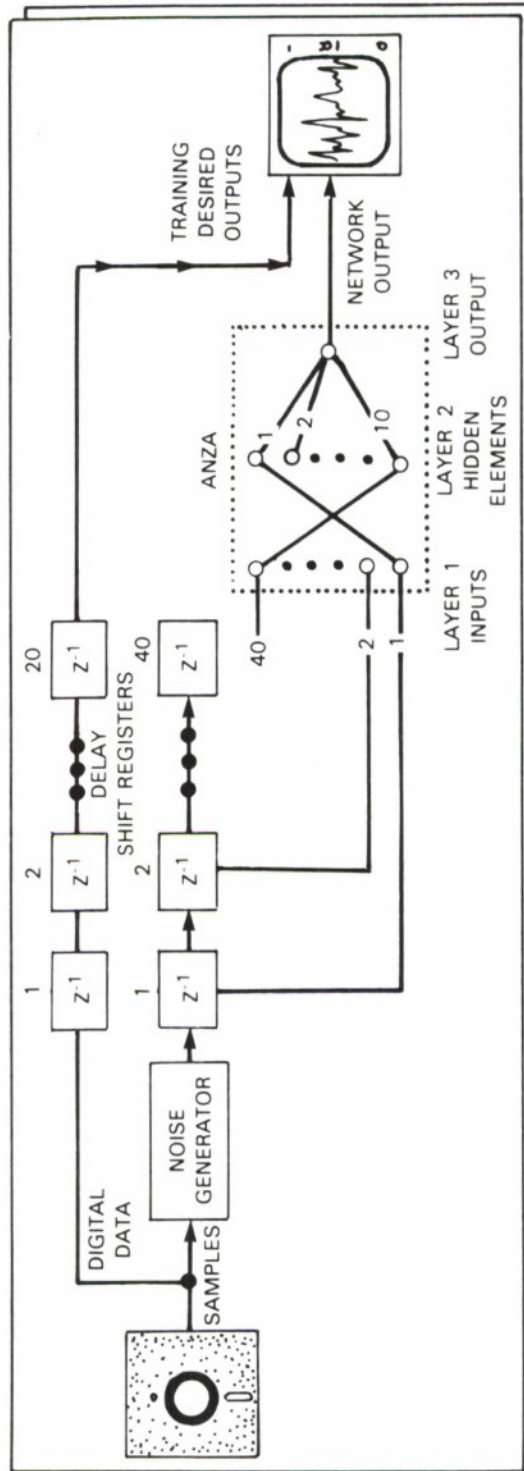


Figure 3-2. Heartbeat from Noise Demonstration Using the Backpropagation Network.

The results showed that the nonlinear system produced output signal-to-noise ratios approximately 2.5 dB higher than that of the linear system.

It is difficult to draw sharp conclusions from these two experimental studies. Both used data which were artificial in the sense that noisy measurements were simulated by adding noise to the signal recordings which were essentially noise-free, and neither had a theory to explain the performance gains.

It is desirable to compare these experiments with analytical results, such as those which might be provided by estimation theory. However, as mentioned earlier, the lack of good models for the signal variations and channel effects seriously hinders this approach. Moreover, the solutions provided by estimation theory are very sensitive to the exact state of knowledge at the receiver concerning the signal and channel. For instance, if a periodic waveform whose shape (and period) was fixed for all time were to be recovered from data in which this signal was corrupted by additive, stationary noise, the optimum estimator would produce waveform estimates which would steadily improve with time to provide an estimate with arbitrarily high accuracy. In the two experiments described above, the waveform could just as easily have been quasi-periodic, with intervals between the pulse waveforms that varied randomly. As long as the actual noise-free waveform was available for training, the systems could still adapt to provide the desired outputs. In the case of the estimation theory approach, a pulse train with random jitter between pulses would lead to a composite estimation problem, having to simultaneously estimate time of arrival as well as the shape of the pulse waveform. This problem leads to nonlinear estimation equations for the optimum estimate of these quantities which would have to be solved by some approximate methods. One way to realize approximate solutions would be to cast the estimation equations in the form of an energy function so that finding the maximum-likelihood estimator would correspond to finding the minimum of an energy function [17]. Then simulated annealing or some other optimization technique could be used to obtain approximate solutions to the estimation equations. Thus a neural network that implements a simulated annealing algorithm could be used to provide an approximate, iterative method of solving the estimation equations for the best (maximum likelihood) estimate of the quantities desired in this problem.

In summary, adaptive filtering techniques have proved valuable in signal processing problems where training data is available. The widely used adaptive equalizer for dispersive telephone channels is an example of this. The studies discussed here which attempted to compare linear adaptive filters, neural networks, and nonlinear adaptive filters addressed artificial problems. However, the results showed some advantage for the nonlinear systems, although without accompanying theory. Certainly the results have no clear implications for the claimed applications of neural networks in detecting arrhythmias in EKGs or to separating fetal EKGs from the mother's.

Estimation theory, on the other hand, could provide highly accurate estimates of an unknown but invariant waveform with known time of arrival, which is not unlike the knowledge implicit in the availability of the desired output for training the adaptive systems. In composite estimation problems where the waveform and its time of arrival have to be simultaneously estimated, exact solution of the (nonlinear) estimation equations is difficult, but a neural network may be used to provide what amounts to iterative approximations to the solution of such equations. It appears that if a great deal of training data is available, then the data could be used to formulate good statistical models for the signal and channel to apply estimation theory, which might, however, present difficulties because of the nonlinear equations. If only a limited amount of training data is available, then the adaptive system approach may be more suitable. But

a neural network system which learns inefficiently may not adapt adequately with limited training data either.

At present, it is not possible to say which approach is best-suited to building a signal processor when limited data on the signal source and channel are available. The problem of optimal use of available data, whether for formulating statistical models for system design or for training adaptive systems, is deserving of detailed study to try to quantify the basic issues of optimal use of limited information.

3.6.2 Prediction Of Time Series

A problem that occurs repeatedly is that of prediction of future sample values of a time series. The predicted tracks of aircraft are often needed for a number of applications like estimation of potential for mid-air collision. Prediction of the coordinates of aircraft position at some time in the future on the basis of measured positions (coordinates) provided by a radar sensor can be viewed as a problem in prediction of time series. Prediction of sunspot activity can also be viewed in this light, and, of course, there is always a great deal of interest in schemes for the prediction of future values of common stocks on the basis of past stock prices and various market composite indices. The waveform estimation problem discussed above is very similar – but not quite exactly the same as – time series prediction, since it involves estimation of the latest sample value on the basis of past noisy measurements.

Alan Lapedes and Robert Farber have attempted to do prediction of time series using neural networks [13]. In their work, they use a multi-layer perceptron to predict values of a nonlinear dynamical system that exhibits chaotic behavior. The scheme is based on the familiar sliding window, which takes N consecutive samples of the series and maps them into a value that represents the prediction of a sample value at some time interval ahead of the window. Chaos in dynamical systems mimics the behavior of pseudo-random number generators, which appear to be random in many statistical tests, and, hence, might be expected to be difficult to predict.

The particular dynamical system studied by Lapedes and Farber is the Glass-Mackey nonlinear differential equation, which has an infinite dimensional phase space – i.e., an infinite number of values are needed to describe the initial conditions for this system. They used a network with two hidden layers, with the number of neurons being determined by the required degree of accuracy of the predictions. They typically used a window size of four samples, and trained the network using 500 input/output pairs – i.e., 500 sets of four samples each of the time series were used as inputs together with the computed value of the sample to be predicted (ahead of the window). Prediction intervals of up to 400 samples ahead of the last input value were used.

The accuracy of the predictions was normalized to the dynamic range of the function (over an unspecified time interval). The plot of prediction accuracy (ordinate) versus the prediction interval (abscissa) is shown as curve “D” in Figure 3-3. A recursive prediction method was used with the neural network and its performance is shown as curve “E” in this figure. The normalized accuracy of the neural network predictors was compared to the conventional Widrow-Hoff linear adaptive filter [34] and polynomial curve-fitting predictors. The Widrow-Hoff predictor performance is given by curve “B” of Figure 3-3 and non-recursive and recursive polynomial predictors are given by curves “A” and “C”, respectively. The neural network clearly gives superior predictions for this particular dynamical system.

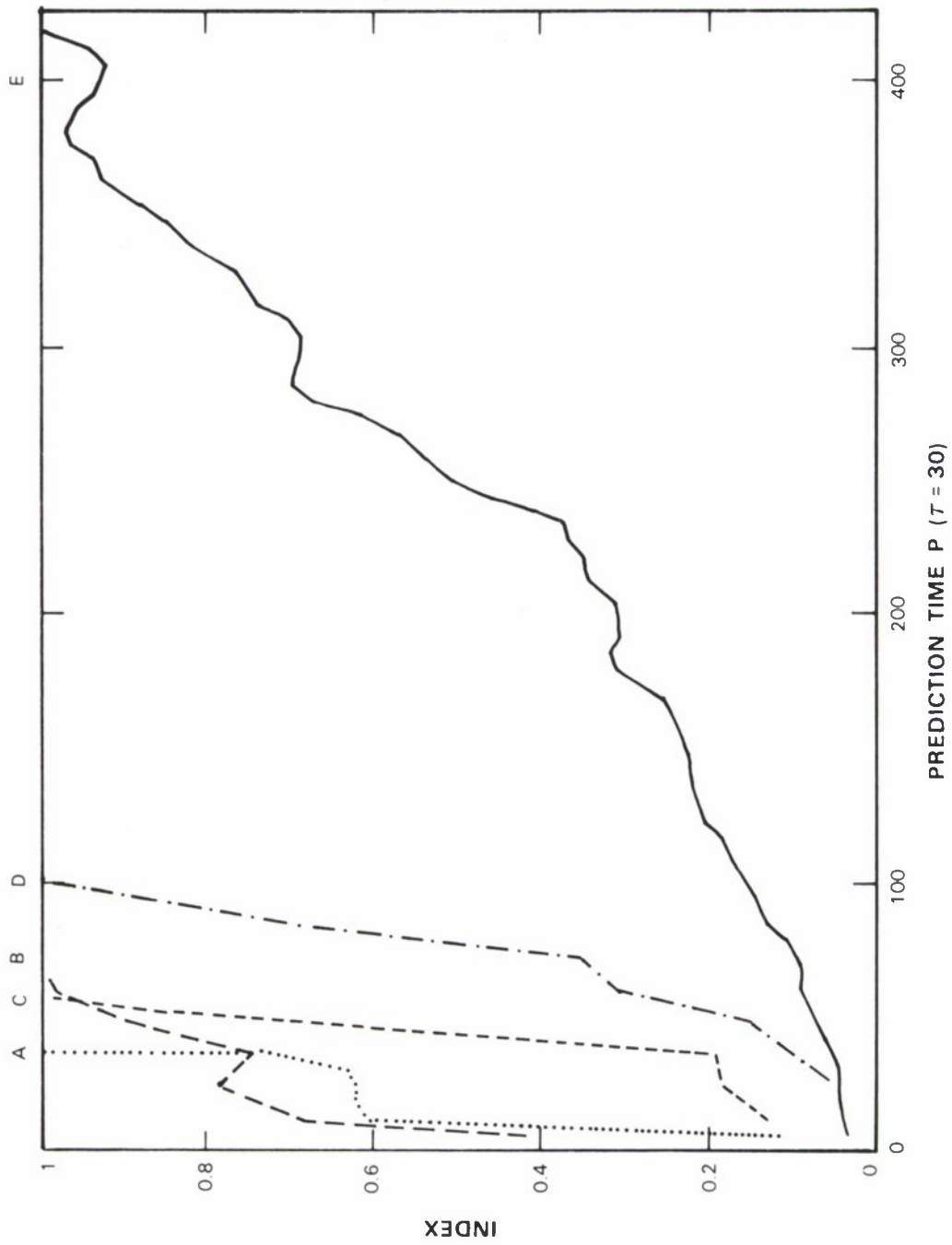


Figure 3-3. Prediction Accuracy for the Backpropagation Network. A = iterated polynomial, B = Widrow Hoff, C = non-iterated polynomial, D = multi-layer neural network (non-iterated), and E = iterated multi-layer neural network.

Lapedes and Farber make the point that neural networks are seldom touted as providing a high degree of numerical accuracy in calculations, yet in this application, the neural network gave the highest accuracy. Some confusion may result from this statement, since in this Study a neural network implemented as an algorithm running on a digital computer is differentiated from a neural network as implemented directly in hardware. The neural network time series predictor of Lapedes and Farber is, in fact, an algorithm running on a digital computer using the floating point capability of the computer. This statement should not be confused with the discussions in other parts of this Report concerning implementation of neural networks in hardware using analog processing elements, where an important issue is tolerance of the network to neural element fabrication inaccuracies and defects. Their claim is simply that the multi-layer neural network algorithms applied to this time series prediction problem can provide the requisite accuracy when the algorithm is done in floating point math on a digital computer.

The paper by Lapedes and Farber discusses the multi-layer network as providing a functional approximation of a mapping of R^n into R^m , which is then used for interpolation or extrapolation to obtain predictions. Their discussion points out that only two hidden layers are needed for arbitrary nonlinear functions (invoking Kolmogorov's result), and the required accuracy of the function determines the number of neurons needed in each layer. The use of a chaotic function as an example for time series prediction required the functional approximation of what might be considered a difficult mapping. However, no general conclusions can be drawn regarding the efficiency of the approach for other classes of functions. This application of neural networks again illustrates the basic property of neural networks as implementing mappings that can be defined from a series of examples.

3.6.3 Summary of Signal Processing Applications

The application of neural networks to problems of estimating and predicting waveforms has been examined. The examinations discussed here are exploratory studies. They indicate that in problems where sufficient data is available for training, it may be useful to develop a trainable system architecture and optimize its performance by training using the currently known learning algorithms for neural networks. The use of neural networks introduces nonlinear processing that adds more flexibility to the system capabilities, which may lead to better performance than conventional linear adaptive systems in certain applications. The applications considered here show some performance advantage, but not a major advantage. Even so, research will probably continue in the exploration of new classes of filters, nonlinear signal processors, and learning algorithms for adaptive systems for many applications. It is impossible to predict the degree of success that will be achieved in these efforts, but it is likely that the extra flexibility inherent in these systems will lead to some number of practical applications for which neural network signal processors will offer better performance in a smaller, cheaper package.

Comparison Assessment Summary: Signal Processing Applications

Application: Signal processing for waveform estimation and prediction in the presence of channel noise and distortion.

Accuracy: Limited study of somewhat artificial examples shows performance of current neural network models to be comparable to conventional adaptive linear filters.

Response Time: Computational power needed depends on the waveform bandwidth and time constraints, with the development of VLSI implementations becoming cost-effective over serial or parallel computers at some point.

Knowledge Acquisition/Representation: The tradeoff is modeling-analysis versus training to optimize system performance. It is not clear which approach is advantageous at present. With abundant data available for training, if an adequate system architecture can be formulated so that it can be adequately adjusted by training, the adaptive system approach would have an advantage.

Physical Characteristics: VLSI implementations would have advantages for space-constrained applications.

Projection: Although there is no hard evidence at present, it is likely that nonlinear adaptive systems will find utility in signal processing applications in the future because of the greater flexibility of such systems and the difficulty of analytic approaches for nonlinear systems.

3.7 SPEECH APPLICATIONS

3.7.1 Text-To-Speech Synthesis

The problem of synthesizing speech from text is usually treated in two separate stages: first, translating the text into a representation for sounds; and second, using this representation to control a device that can generate the sounds. The text, a sequence of alphabetic characters, is first translated to a sequence of symbols representing elemental sounds or phonemes. There is a long history of research on this problem which is summarized quite completely in a recent article by Klatt [12]. There are a number of commercial products available, some of which produce speech of remarkable quality. The best may be DECtalk, a system sold by Digital Equipment Corp. (DEC) which is based heavily on the work of Klatt. A sample recording of over 30 speech synthesizers included with Klatt's article shows DECtalk to generate clearly intelligible speech at a rate of 300 words per minute.

DECtalk took approximately 15 years of development and refinement, most of which were spent on the speech generator stage of the system to produce high-quality synthetic speech. The first stage of translation of alphabetic to phonemic representation is done using acoustic-phonetic rules, and took approximately three years, according to Klatt.

Sejnowski and Rosenberg have published papers on a system called NETtalk, which they describe as "a parallel network that learns to talk" [30,29]. In this system, they used a three-layer neural network to do the first stage; the second stage was done using the speech generator portion of DECtalk. A seven-character sliding window is used to provide inputs to the neural network, and the output is one of 21 characters representing a set of articulatory features. A unary representation is used for the input text characters as well as the output phonemes, so the input layer has a total of 203 neurons and the output layer has 21 neurons. The number of units in the hidden layer was varied from 0 (zero) to 120, to test the sensitivity of the performance to this parameter. The network was trained using the backpropagation learning algorithm.

The network was trained using a data base of 20,000 English words together with their correct translation to phonemes. The actual training set consisted of the 1,000 most commonly-used words in this dictionary, each word being presented to the network individually – i.e., each word slid through the input window of seven characters by itself. The learning curve for this network varied with the numbers of units in the hidden layer. The set of 1,000 words was presented 30,000 times during the training phase; correct translation of text was done more than 98% of the time on the training set. When tested on the full dictionary of 20,000 words, the network translated 77% of the words correctly. This falls far below the level achieved by the approach used in DECTalk. In the opinion expressed by Klatt in his article, the structure of NETtalk in operating only on seven contiguous letters is probably too restrictive to ever be competitive with DECTalk, even for much larger training sets. The entire set of NETtalk experiments was carried out during a summer, which was considerably shorter than the time taken by a human expert to formulate acoustic-phonetic rules of DECTalk, but the performance of NETtalk also fell considerably short of that of DECTalk.

The NETtalk example is presented here because it is considered one of the highlights of recent neural network accomplishments. A detailed comparison between the commercial DECTalk system, regarded as the state of the art in text-to-speech synthesis, and NETtalk is not possible because the data are insufficient to compare the time for knowledge acquisition for both approaches to achieve the same level of performance, or to compare the performance for the same amount of time invested in knowledge acquisition. Thus the Panel can conclude very little, on the basis of the NETtalk experiments, with respect to the efficiency of neural network knowledge acquisition or about ultimate performance in this application.

Comparison Assessment Summary: Text-to-Speech Synthesis

Application: Text-to-speech synthesis.

Accuracy: The current NETtalk design is inferior to commercially available systems and the architecture is believed by speech experts to be inadequate for significant performance improvements. There appears to be no interest in pursuing new designs to make NETtalk commercially competitive.

Response Time: The simple NETtalk system could be implemented adequately using a personal computer.

Knowledge Acquisition/Representation: NETtalk was demonstrated in an amazingly short time because of the learning, although it is not clear that the same could be achieved with a neural network system that achieved performance levels competitive with commercial products.

Physical Characteristics: A VLSI implementation of a NETtalk-like system might be more appealing for certain applications, like portable aids for the handicapped, if it were implemented in custom VLSI, which is certainly feasible for simple multi-layer networks.

Projection: The future of text-to-speech synthesis by neural networks is uncertain, needing performance improvements and perhaps even VLSI implementation to produce an interesting product.

3.7.2 Speech Recognition Using Neural Networks

The application of neural networks to speech recognition is covered in detail in *Part II, Chapter 9* of this Report.

3.8 MACHINE VISION

Machine vision has evolved from a branch of artificial intelligence into an independent field of research. The objective of machine vision is to enable machines to “see” – that is, to use visual sensors to avoid obstacles, to detect and recognize objects of special interest, to monitor scenes, etc.

This field is still an active research area, although there is a considerable amount of commercial activity connected with industrial robotics applications. The techniques currently used in machine vision are the result of a combination of various disciplines, which at the very least include physical optics, signal processing, computer science, biophysics, neurophysiology and cognitive science. David Marr first espoused the treatment of vision as an information processing problem and this view is now widely accepted. More recently, the vision problem has been approached as an “inverse optics” problem, which is an ill-posed problem in that it is under-determined without imposing additional conditions or constraints on the allowable solutions. In biological systems, it is the knowledge embedded in these systems which allows them to solve the inverse optics problems and extract useful information out of noisy and partial information contained in scenes. Thus, machine vision systems are information processing systems which have to solve under-determined problems with the use of specialized knowledge.

The work in machine vision can be broken into “early vision” and high-level vision. Early vision was described by Marr as going from images to surfaces. The job of early vision is to extract information about surfaces from input images, such as boundaries of different surfaces, as well as surface shape, motion, color, depth, and texture. Early vision can be regarded as consisting of a set of processing modules, each of which extracts certain information from images in a “bottom-up” fashion. High-level vision has the function of organizing and controlling the flow of information from these modules and combining this information with “high-level knowledge” to analyze, understand, and use visual inputs in other tasks.

It is impractical to present a complete description here of the state of research in machine vision, including an assessment of the overall impact that neural networks have had on the field (for more information, see *Part II, Chapter 8* of this Report). Suffice it to say here that neural networks are indeed having a significant effect on machine vision; the Technology Assessment Panel will try to illustrate this effect by discussing only a few different areas of machine vision.

3.8.1 Image Segmentation

The purpose of image segmentation is to partition a two-dimensional array of image samples into disjoint subsets, each of which represents a different region or surface in the scene. This partitioning can be done on the basis of some property or characteristic of the image samples, such as the intensity or perhaps Doppler velocity (which can be provided by some optical radar sensors), or as being part of a textured pattern in the image. Two basic approaches to image segmentation have been used:

- Finding the bounding contour of a region,
- Grouping image samples with a common property.

One approach works on the edges of regions and the other works on the interior of regions. The edge-based approach is a traditional method that has been in use for many years, employing spatial filters to enhance the parts of the image where the samples change character, and using nonlinear operations to define edges of regions or surfaces. The region-based approach uses techniques such as “region growing” and morphological filters to fill in the interior and smooth out the boundary of a region in the image. These approaches classify image samples (pixels) based on the use of local operations. Although many machine vision systems use these techniques to define surfaces or regions within an image, they are by no means standard or universally agreed upon as the best way to do segmentation.

Because of the inadequacies of the traditional segmentation techniques, another approach has emerged in the last four or five years: stochastic modeling, in which image samples are considered to be generated by a random process that changes its statistical properties from region to region. The random process that generates the image samples is a two-dimensional analog of a Markov process, called a Markov random field (MRF). Image segmentation is then considered as a statistical estimation problem in which the system calculates the optimal estimate of the region boundaries from the input image. The estimation problem includes simultaneous estimation of region properties as well as the boundaries of the regions, and results in a set of nonlinear estimation equations that define the optimal estimate of the regions. In this statistical formulation of the problem, the processor must find the maximum *a posteriori* probability estimate of the image segmentation.

Geman and Geman showed in 1983 that this problem can be recast into the minimization of an energy function, which, in turn, can be solved approximately by optimization techniques such as simulated annealing [5]. The interesting point is that simulated annealing can be implemented using a neural network with local connections, in which the network iterates or relaxes into a global solution using these local operations. The MRF provides local constraints on the image samples, but it is now well-known that these local constraints uniquely determine the class of probability distributions over the whole image. Thus the MRF actually imposes local constraints that imply global properties, and the estimation theory approach attempts to make best use of the constraints provided by the MRF model.

The MRF approach has not yet been developed to the point where very complex scenes can be modeled completely and processing algorithms and/or neural networks implemented to solve the segmentation problem for all time. There remains more research to do on modeling images and implementing efficient and accurate approximations to the optimal estimators. But the connection between early vision, statistical estimation theory, approximation techniques, and neural networks has been firmly established, and there is a rapidly growing literature on this new direction of research in machine vision. The work on simulated annealing provided the original connection between solving a difficult nonlinear estimation problem and using a neural network to produce approximate solutions.

The segmentation problem can also be treated as an ill-posed problem which can be solved as a variational problem with constraints on the set of allowable solutions [24]. Variational problems can also be recast as energy minimization problems, again leading to neural networks which can provide approximate solutions to the minimization problem. The construction of an energy function which constrains the image

segmentation process to produce good solutions is currently the subject of research as an alternative, but related, approach to that of using MRFs.

In another approach, Grossberg and Mingolla have formulated a neural network model that combines boundary processing with region-based processing to arrive at an overall solution to the segmentation problem [7]. Their model involves the solution of a large set of nonlinear partial differential equations (PDEs), which is a time-consuming task on current digital computers. However, the model could someday be directly implemented in analog hardware to operate in real time. There is still more research to do to optimize the many parameters of the model before making the investment to go to hardware. Energy functions are used in this work to establish the stability of the dynamical system defined by the PDEs, reinforcing the notion that this type of analysis is basic to neural networks and to early vision.

It is clear from this very brief summary of current directions in one area of early vision – segmentation – that neural networks are inseparable from machine vision, and are actually in the mainstream of current research, being viewed as someday becoming part of massively parallel processing architectures for realtime machine vision systems.

3.8.2 Learning A Color Algorithm From Examples

The observed brightness or irradiance of a surface element in an image is the product of the illumination intensity falling on that element and its reflectance. When the illumination is not uniform across a surface, the observed brightness varies across the surface and cannot readily be separated from a variation of reflectance across that surface. That is, the problem of separating surface reflectance from illumination is ill-posed, or under-specified. Yet biological organisms do this separation all the time. The explanation is that they use knowledge about the nature of natural illumination and surfaces that allows them to separate these two functions under certain conditions.

Poggio and Hurlbert have formulated a model for color perception that produces color constancy in the presence of varying illumination [21]. It is similar to other “retinex” models, in particular to one presented by Edwin Land, that had been based on physical optics considerations. The question that they then addressed was whether such a color model could be derived purely from examples. The point was not to establish that biological systems actually adapted themselves during a certain period of their development, but rather to demonstrate that there exist mechanisms that could be implemented in biological systems that could solve ill-posed problems by using examples rather than physics.

In the Poggio/Hurlbert color constancy model, logarithms are used to express the log of the observed brightness as the sum of the log reflectance and log illumination. The question posed was how to separate these two quantities from observing only their sum. This is clearly an ill-posed problem. One approach is to constrain the allowable types of functions that the system could produce as outputs. To do this, it is assumed that the surfaces had constant reflectance within their boundaries and that the illumination had a smooth variation across the surfaces. A variational problem is then formulated to minimize the mean squared error in approximating a brightness input corresponding to a Mondrian figure by constructing it as the sum of a smoothly tapering function (the illumination) and a function consisting of jumps between constant values (the reflectance).

Poggio and Hurlbert showed that such an ill-posed problem with a mean squared error constraint leads to a linear filter as the best solution. When such a linear operator is postulated, the parameters of the linear operator can be estimated from a set of examples for which the input brightness function is given as well as both the reflectance and illumination.

To simplify calculations, Poggio and Hurlbert used a single scan line of a grey-scale Mondrian for each example. They then used many such examples to solve for the linear operator that best separated the inputs. And then they tested the operator on test data to see how it worked. They also examined the form of the operator to compare the processor derived from the training examples with the models derived from physics. They found that, indeed, the linear filter derived from training “was qualitatively the same as that which results from exploiting spatial constraints on reflectance and illumination.” They only needed to assume the correct form of the operator to solve this class of ill-posed problem.

In summary, this work on learning a model for color constancy from examples has not led to a new model. However, it does establish that (a) interesting processing problems can be solved by evolving a processing operator from a large set of training examples for which the correct solution is known for each input, and that (b) the “learned” solutions can, under certain conditions, turn out to be the same as those which exploit special knowledge about the inputs in order to solve an ill-posed problem.

Restated differently, it does appear feasible to extract the specialized knowledge to solve an ill-posed problem by using a learning algorithm together with a large set of training examples. In the words of Poggio and Hurlbert, their results suggest that, “evolution may recover and exploit natural constraints hidden in the physics of the world.” This work illustrates the influence of neural networks, and notably the idea that such networks can learn, on the thinking of workers in the field of machine vision.

3.8.3 Visual Motion

The movement of objects in our visual field is a very important source of information for many tasks – such as detecting threats, navigating to avoid objects, and determining the three-dimensional structure of objects from observing their motion, to name only a few. The motion of objects can be estimated from sequences of images using two basically different approaches. One relies on the spatial and temporal gradients of image intensity, and the other is based on image-to-image tracking of a set of extracted features of objects.

Both of these approaches have been pursued for some time to extract object motion from image sequences, and there is abundant literature on this work. However, new directions are currently emerging in both approaches which are based on processing image sequences using a locally-connected array of simple processors. One of these approaches offers the opportunity for direct analog implementation which could provide realtime extraction of motion parameters for use in higher-level processing. These implementations could also be compact processing systems which would be suitable for space- and weight-constrained robotic applications. One such design for a “retina” has been implemented by Carver Mead [16], providing a concrete example of what these new approaches offer for the future, when implemented in appropriate technology.

The intensity gradient approach has been pursued in Poggio’s laboratory, most notably by Christof Koch [22,9]. There, they begin with the observation that all early vision problems that attempt to recover

properties of visible three-dimensional surfaces from two-dimensional intensity arrays are ill-posed problems [23]. This includes edge detection, binocular stereo, and estimation of the velocity field from optical flow. Thus, the estimation of object motion from optical flow involves the solution of a variational problem with constraints based on physical considerations, or the determination of a maximum *a posteriori* (MAP) estimator for a Markov random field (MRF) model which embodies knowledge in the form of *a priori* probability distributions. The solution of these difficult nonlinear problems can be approached by means of minimization of appropriate energy functions, as was discussed earlier in this section on machine vision.

Koch, *et al.*, have recently shown that optical flow can be obtained using analog resistive networks to process image intensity data [9]. They also show that smoothness constraints to solve the ill-posed problem do not provide good definition of velocity at object boundaries. To offset this, they adapted the approach of Geman and Geman by defining an energy function which included terms that encouraged the insertion of sharp intensity edges at object boundaries [5]. They then showed that this form of the energy function leads to good velocity estimates within object boundaries and clearer definition of velocity discontinuities for velocity segmentation of the image. This non-quadratic optimization problem can be solved using simulated annealing, as Geman and Geman did. But Koch shows that, again, an iterative approximation procedure can be used to obtain solutions which are qualitatively similar to those obtained from simulated annealing, and this iterative procedure can be implemented using an analog resistive network.

Even with the approximations used in these computations, it takes considerable computational effort to obtain velocity estimates for image pairs, even on a multiprocessor computer such as the Hypercube. For this reason, direct implementation of these networks in analog VLSI is of great interest to simultaneously realize realtime processing speeds and compact processors. An ordinary CMOS process can be used to realize a compact resistive network using transistors running in the subthreshold range. A similar network has already been built by Mead to implement a 48-by-48 pixel “silicon retina” on a single chip that attempts to emulate the first stages of processing of the vertebrate retina [16]. This chip contains integrated photoreceptor and processing elements that generate realtime outputs which correspond to retinal ganglion cells. The retina chip uses light as an input, and processes this input to produce a representation which is (relatively) faithful to a biological retina. The chip is one quarter of a square centimeter, consumes 100 microwatts of power, and operates in real time.

An entire visual system would be composed of many stages of processing, the first of which could be implemented (in the near future) in the form of a chip such as this one. The efforts of Mead and co-workers illustrate two points:

- Early stages of biological processing appear to be reasonably well understood, and
- Biological processing, when understood, can be implemented quite successfully in hardware – even when using technology that is very different from the wetware of biological systems.

The development of a complete visual system in special silicon chips or wafers, however, will require a deeper understanding of the higher levels of visual processing than is now available. The pursuit of this approach to information processing systems will have to rely on the neurosciences to increase present

understanding of how such biological systems function. Perhaps the study of invertebrate imaging systems will help.

The feature-tracking approach to obtaining object motions from a time sequence of images has a number of difficulties associated with it. First, the extracted features must be robust – i.e., must be reliably extracted in image after image as the object moves and changes aspect. Then the features must be correctly associated from image to image. In spite of these complexities, feature tracking can provide quantitative velocity information about motion of objects.

To illustrate the principles involved in this approach, consider the problem of estimating the velocity of a set of point targets from a sequence of images or frames. Each image is a binary image with only a few pixels that are “on” (say, white), and successive frames show the positions of the targets as they move.

This is similar to the problem of tracking multiple objects in space using an optical sensor, or for tracking individual features that have been extracted and displayed in a feature map for several complex moving objects in a scene. If the targets (points) move farther in successive images than the average separation between points in any one image, it will be difficult to tell which pairs of points in two successive frames correspond to a single target or object. Similar difficulties arise when some of the points are occasionally missing in some frames and when some “extra” points appear in some images at random locations.

This problem was attacked by workers at TRW in the context of optical tracking of space objects [1]. They have formulated the problem as an energy minimization problem by constraining the range and rate of change of velocities of targets, and have used a Hopfield network to do the frame-to-frame association of targets. This is similar to the approach that would be taken in treating this as an ill-posed problem.

Recently, Waxman and Bergholm have devised a novel approach to this problem of visual motion estimation using dynamic features based on what they call “convected activation profiles” [32]. Each feature of a feature map causes an activation pattern that diffuses outward from the initial locations of the feature and also decreases with time. This spatio-temporal activation pattern is the activation profile of a single feature at a single instant of time. When the feature moves from frame to frame, each occurrence of the feature produces a new activation profile which is superimposed on the previous ones. The activation profile of a moving feature also moves with that feature and is thus said to be “convected.” The specific form of the activation profile that Waxman and Bergholm chose is a Gaussian shape, centered on the feature providing the activation, with fixed variance and with a Gaussian decay over time. The activation profile of a feature map is just the convolution of the feature map with this Gaussian kernel.

The activation pattern produced by a moving feature is just a moving pattern with the past decaying away over time. The activation provides a means for a feature to make itself “felt” at nearby locations and for a certain amount of time. A moving feature will produce a superposition of the kernel activation profiles which overlap if the spatial and temporal parameters of the kernel are chosen properly. This overlapping actually serves to solve the frame-to-frame feature association problem. It turns out that an estimate of the velocity of the moving feature can be obtained as a ratio of certain partial derivatives of the activation profile – namely, the ratio of the time rate of change of the gradient of the activation and the local curvature of the activation at any point. Of course, the accuracy of the velocity estimate depends on the relationship between the velocity and the parameters of the activation kernel (spatial and temporal decay constants). Accurate velocity estimates for a wide range of velocities would require a bank of different

activation profiles, which could be realized in this model as an activation which diffused over image space with time. Sampling this time-varying activation at various times would correspond to velocity filtering.

Multiple features closely spaced will produce activation patterns that interact with each other, influencing the accuracy of velocity estimates. If different features are separated by more than the spatial extent of the kernel, there will be little interaction in any single frame. Again, the parameters of the kernel must be carefully chosen to extract velocity accurately. Slowly moving features could be processed by narrow kernels, limiting the velocity to small values to achieve accurate estimates.

This scheme for obtaining velocities of features is very new, and there are many issues that must be worked out to establish it as a practical approach to visual motion processing. Some of these issues are:

- The effects of noise,
- The robustness of feature detection, and
- Optimal choice of kernel parameters for discrete spatial sampling and discrete time sampling.

Nevertheless, it appears to be very promising, achieving frame-to-frame feature tracking without explicitly solving the feature association problem. The motion of features is derived from the dynamics of the activation profiles, involving simple convolutions with feature maps. The activation profile can be implemented by driving a diffusion network with a feature map, either continuously in time or with time-discrete inputs. This same diffusion network could provide a bank of velocity filters by sampling it at different times as it continues to diffuse, even ascribing several different velocity interpretations to a single feature. Such a diffusion network could be implemented in analog circuits for realtime processing speeds, which would provide a significant step toward realizing a realtime image motion analysis system – which presents an insurmountable computation load to conventional processing schemes. The use of diffusion networks has recently been applied to feature extraction in an object recognition system, and the approach looks very promising in this context also [28,27].

3.8.4 Summary of Machine Vision Applications

Even though machine vision is one area that clearly needs massive parallelism for fast processing, it may be somewhat surprising that neural networks are in the mainstream of the new directions in this field. Problems of early vision – such as edge detection, image segmentation, motion, and stereo – are being formulated as ill-posed problems which can be solved using variational methods which lead to energy minimization problems.

Another new approach is the use of statistical models, such as Markov random fields, which lead to maximum *a posteriori* probability estimators that are nonlinear and also involve minimization of an energy function. It has been shown that these problems can be solved using resistive networks which can be implemented in analog VLSI for realtime solutions.

Similarly, in the area of deriving object motion from image sequences, diffusion networks have been shown to be extremely useful in extracting velocity information from feature maps. These networks also can be implemented in the form of locally-connected neural networks.

An important feasibility demonstration of early vision processing functions in analog VLSI is provided by Mead's silicon retina, which detects moving objects in an image plane. This retina, implemented on a single chip, can perform its processing functions in realtime, and establishes the value of this approach to providing modules for front end processing in machine vision systems. It is very likely that one of the earliest significant contributions of neural networks will be early vision modules implemented in analog VLSI, and used in combination with conventional techniques in a machine vision system.

High-level vision functions are much less well-understood and the prognosis for solutions of any type are harder to foresee. Basic problems remain in the areas of representations, modular processing architectures, top-down system control, and matching techniques for object recognition. The biological systems are not well-understood at these levels yet. Pattern classification methods may be the first approach to these high-level vision problems, until neuroscience can furnish more knowledge to guide the pursuit of new concepts.

In summary, the value of compact, fast, robust processing modules for robotic vision applications is clear. The field of machine vision has diverged from conventional AI to meet these unique needs and, as a result, is currently pursuing techniques that are more in the spirit of neural networks than of AI. Analog VLSI hardware has been demonstrated as a workable technology for implementation of early vision functions, and neural network modules could be embedded in the near future in machine vision systems using standard technology. More fundamental work is needed to solve the problems of higher-level vision functions.

Comparison Assessment Summary: Machine Vision Applications

Application: Low-level vision processing.

Accuracy: Neural network approaches have not yet surpassed conventional methods in performance, but they have been under development for only a short time.

Response Time: The neural network approaches are aimed at VLSI implementation to achieve processing speed through massive parallelism, but only Mead's hardware system has achieved this promise to date.

Knowledge Acquisition/Representation: Early vision processing modules do not usually incorporate learning, and the algorithm development process is more like conventional analytical approaches.

Physical Characteristics: The intended VLSI implementations of modules will produce compact processors with very high information processing rates

Projection: Although there is very little neural network hardware today that exploits massive parallelism in early vision processing, the current directions in algorithm development are driven by the desire for practical VLSI implementations to achieve compact and realtime processing for vision systems. The algorithms are not yet mature, but they appear to be very promising, and experimental VLSI chips and/or wafers could start appearing regularly in two-to-four years. A second generation of VLSI processors could appear in four-to-six years. This second generation might be adequate for use in a practical vision processing module in a real vision system in, say, six or more years.

High-level vision is not well enough understood at this time to make projections relative to the contribution of neural network processors.

4. CONCLUSIONS CONCERNING THE ASSESSMENT OF NEURAL NETWORK TECHNOLOGY

4.1 CURRENT STATUS OF NEURAL NETWORK TECHNOLOGY

Neural networks represent a unique “style” of information processing that is viewed by many as promising very important benefits, such as:

- Processing speed through massive parallelism,
- Learning as a means of efficient knowledge acquisition and embedding,
- Robustness with respect to fabrication defects, and
- Compact processors for space- and power-constrained applications.

In the course of trying to compare neural networks with conventional approaches to information processing, it became clear that the volume of commercial business in neural network systems is quite small at this time, and that there are few practical neural network systems functioning in the field right now. Currently, almost all of the activity in this field is aimed at trying to understand neural networks and to develop and demonstrate their capabilities in simple applications.

Almost all applications of neural networks are implemented at this time in the form of an algorithm running on a digital computer, which does not provide much information concerning the processing speed, robustness, or compactness of a hardware implementation. In fact, there is very little neural network hardware operating at this time, and all of it is experimental.

One outstanding example of hardware which does indeed lend credibility to the “promise of neural networks” is the silicon retina built by Mead and his associates, which does realtime processing of moving objects with a single chip plus ancillary circuitry that fits on a single circuit board. This chip is, however, experimental, has a very small retina, and is not ready for use in a practical machine vision system at this time.

All other neural network hardware in existence at this time is experimental, consisting of silicon chips and optical implementations of some of the standard neural network configurations, such as Hopfield networks, multi-layer perceptrons, feature maps, etc. In the area of signal processing, the adaptive equalizer – which consists of a single adaptive, linear neuron – is in widespread use, but the Technology Assessment Panel does not really consider such a simple adaptive linear system to be a neural network in the context of this Report. About a decade ago, a small company marketed a word recognizer that was really a neural network designed by Tom Martin in his Ph.D. thesis [31]. This recognizer outperformed all others on the market at the time, but was surpassed by the new designs of the 1980s, and it appears that this company is no longer in business.

Thus, there are no possibilities of neural network hardware demonstrating the benefits of processing speed, robustness, and/or compactness in practical systems at this time. And there appears

to be much research and development remaining in order to realize these benefits of neural network hardware implementations.

Of what value are neural network algorithms running on a digital computer (referred to here as computer simulations)? The few commercial products on the market today are of this form. These are mainly in the area of pattern classifiers, where neural networks have been shown to provide performance competitive with traditional pattern classification techniques. Among these are trainable neural network classifiers for loan underwriting, hand-printed character recognition (including Japanese Kanji characters), and recognition of protein sequences in studies of DNA.

Computational maps for control of robot manipulators require neural networks of modest size and processing speed; these have been successfully demonstrated using digital computer implementations, but again, these are in experimental systems at the present time.

Applications of neural networks to signal processing problems have been in the form of computer simulations, and some interesting performance improvements have been demonstrated in the area of adaptive, nonlinear prediction of chaotic behavior. In the more traditional context of waveform estimation, the advantages of adaptive, nonlinear signal processors using neural networks have not been outstanding relative to adaptive linear systems, but there seems to be promise in the area of signal processing.

Neural networks are also being studied for application to machine vision systems – in particular, for early vision processing functions. Promising algorithms for image segmentation, object motion, and stereo have been developed and demonstrated on digital computers, but the computational load associated with early vision processing is too large for considering practical machine vision applications in the form of computer simulations. In fact, the area of machine vision points out the need for powerful neural network simulation machines to speed the development and thorough evaluation of neural network algorithms suitable for hardware implementation.

The study and development of neural network algorithms are providing the impetus for new conceptual approaches to such problems as pattern classification, robotic control, speech recognition, and machine vision. New parallel, adaptive algorithms are emerging which may be useful in practical applications even when running on a general-purpose digital computer. Certainly, evaluation of new algorithms on computers is a necessary step in the development of algorithms – no matter how the final implementation is done in a practical system. **But the benefits of neural networks implemented as algorithms running on digital computers are limited compared to hardware implementations, which can provide fast as well as compact realizations of neural network processors.**

In summary, the current status of neural network technology is fairly immature, with only a few practical systems operating in the field, and these in the form of computer simulations. Parallel adaptive neural network algorithms are contributing new approaches to many information processing problems. **Neural network hardware is in the experimental stage, with at least another generation of hardware development needed before it is likely that neural network chips will be used in an operational system.** Thus, at the present neural networks have not demonstrated most of the benefits of greatest interest – i.e., massive parallelism, learning, robustness, and compact processors.

4.2 EXPECTATIONS FOR NEURAL NETWORK TECHNOLOGY

The application of neural networks to a variety of problems has been discussed in order to evaluate and illustrate their utility; in a significant number of cases, neural networks have matched or slightly exceeded the performance of conventional solutions to these problems. Given that many of these neural network solutions represent the first attempt to solve these problems with this new technical approach, their results constitute a rather encouraging achievement.

In many cases, the neural network solution was an alternative implementation of an existing solution. Even so, the neural network solution often offers the potential for compact implementations with rapid response time, which, in itself represents a “value-added” contribution to the existing solution. But one can reasonably expect that successive attempts at neural network solutions will ultimately lead to unique and more appropriate applications of neural networks to these problems – with potential for dramatically improved performance. The field of neural networks needs a period in which to get away from mimicry, to build on this early experience and develop its uniqueness, before being critically evaluated with respect to its future potential. Thus, although there are few practical neural network systems functioning in the field today that illustrate the true potential of this technology, the experience to date has been encouraging in many areas, and one should expect that the technology will only get better with more experience.

4.2.1 Pattern Classification

Perhaps the surest projection is relative to neural network pattern recognizers. Neural networks can be used to implement many standard pattern classification algorithms, and they have also led to new classification algorithms. Neural networks provide a means of realizing complex map functions using training examples, which is the essence of pattern recognition systems, so this success should not be surprising.

For simple pattern classification problems, neural network systems will continue to be implemented as computer simulations. But for more demanding applications, such as “smart” weapons, neural network hardware will offer compact processors in the next few years, especially for simple problems. As an example, a proximity fuze for a smart weapon can be realized by a multi-layer neural network, which in turn, can be implemented on a single chip within the next two years. More complex pattern classification problems may be addressed by neural networks, but the key issue here is the representation problem. When good representations can be found, good neural network solutions can usually also be found. A thorough understanding of the representation problem is essential to the success of neural networks, and there is a great deal of basic work to be done in this area.

4.2.2 Robotics

Neural networks will most surely appear in robotic manipulators in the form of computational maps for trajectory control. These problems require neural networks of modest size and speed, and they have been successfully implemented so far only as computer simulations, but neural network hardware for robots is definitely feasible and could appear in robots in two-to-four years.

The harder, more general problem of trajectory control under constraints (e.g., avoiding collisions with obstacles) has not yet been solved using neural networks, and the widespread and standard use of neural networks for robotic control problems of this level cannot be projected at this time.

4.2.3 Machine Vision

Although the general area of machine vision will require years of research before any clear cut solutions are established, the problems of early vision are well understood and are very important since they give rise to a huge information processing load which currently dominates the performance of the entire vision system. The recent focus on neural network algorithms for early vision functions that can be implemented as analog VLSI networks is aimed at relieving this bottleneck. Mead's experimental silicon retina demonstrates the basic feasibility of the analog VLSI approach to realizing compact realtime processors for early vision, although the design is not mature enough for practical systems at this time. A number of new algorithms under development will be approaching readiness for silicon implementation, and experimental chips for these early vision functions could start appearing in two-to-four years, with second generation designs coming in four-to-six years. These are massively parallel, non-adaptive processors that may require new techniques in analog VLSI for practical implementation. It does not appear feasible to expect neural network implementations in analog VLSI modules for early vision functions in less than six years, and it could take considerably longer if the analog VLSI technology required is more demanding than that used by Mead.

High-level vision functions are not well understood at this time and so no predictions can be made relative to any kind of practical implementations in this area.

4.2.4 Signal Processing

In the area of signal processing, neural networks have been tried with some success, although there are no practical systems in the field yet. The adaptive equalizer points out the usefulness of adaptive systems for certain signal processing problems.

Neural networks can provide an adaptive, nonlinear capability that will most assuredly be useful in some signal processing applications. Theoretical approaches to nonlinear filtering are very difficult, and the adaptive system approach may again prove, as in the case of the adaptive equalizer, to be the practical way to solve some class of problems. Therefore, even though no such signal processing applications are known at present, the capabilities of neural networks in providing adaptive, nonlinear signal processing will prove useful in some class of problem in the future. More demanding problems could also benefit from hardware implementation of a neural network solution.

4.2.5 Speech Applications

There are several different aspects of speech applications that must be considered relative to neural networks.

Text-to-speech synthesis. The area of text-to-speech synthesis systems has been discussed in some detail earlier. NETtalk was able to achieve an interesting level of performance in the text-to-phoneme translation stage of speech synthesis, and it was able to do this very rapidly compared to the time scale required to complete a rule-based system for acoustic-phonetic rules. However, it appears (to the experts in the speech synthesis field) that the architecture used in NETtalk is not rich enough to ever get to the performance level of the systems based on acoustic-phonetic rules, regardless of the size of the training set and training time. Practical text-to-speech systems are now available on the commercial market at reasonable prices, and so neural networks are not likely to contribute anything to this area, unless a fundamentally new approach is tried and proves capable of better performance than present systems. This not likely since the present systems perform quite well and a long development time would be required. That is, there is no market that is demanding better performance and no motivation to invest in radically new concepts.

Word recognition. Intensive effort is now focussed on the word recognition problem, with the current leader being the hidden Markov model (HMM) approach. The front end processing problem in speech is not nearly as demanding as that of vision, and so programmable digital signal processor chips provide compact solutions to this aspect of the problem. In speech, it is the higher-level problems that are receiving the research effort and reasonable progress is being made. The severe computational demands in word recognition will arise as systems capable of handling very large vocabularies are attempted. In the case of HMM systems, the number of word models that must be processed grows with the vocabulary size. Lippmann has shown that the HMM word recognizer, currently implemented using digital computers, can be implemented using neural networks, but it remains to be shown that practical neural processing elements can be built to carry out processing functions now done in floating point calculations without degrading performance. This evaluation effort will be carried out in the next two years. If neural network implementations prove practical for the HMM word recognizer, then experimental neural network hardware could be fabricated in two-to-four years for recognizers with vocabularies of hundreds of words; experimental recognizers with vocabularies of thousands of words could be expected in, say, four-to-six years, if this HMM approach is feasible. If this approach is not feasible, then a new concept for the application of neural networks to word recognition will have to emerge. There are several approaches being studied at this time, so such a new approach might emerge in two-to-four years, slipping the whole schedule for neural network hardware for word recognition by at least two-to-four years.

Continuous speech recognition. A much more difficult problem than isolated word recognition, continuous speech recognition is still in the basic research stage, with no approach mature enough at this time for hardware implementation. Neural networks may indeed provide a new viewpoint that ultimately leads to a good solution, but this is not apparent at present. The extent of the contribution that neural networks might make to this problem thus cannot be projected at this time.

4.2.6 Autonomous Neural Systems

What about a complete autonomous system constructed entirely out of neural networks, such as a submersible vehicle, or a land rover for planetary exploration? Such systems must include several major subsystems for basic functions – including sensory processing, data fusion, motor control, and higher

cognitive functions like reasoning and planning. A complete sensory perception system implemented entirely using neural networks is considerably farther in the future than the six-plus years projected for analog VLSI modules for early vision functions. Some effort has gone into applying neural networks to cognitive functions, but there were no careful comparisons presented to this Study for detailed consideration.

The area of planning, which has been an active (and somewhat successful) area in AI, has received slight attention in the neural network field, although there appears to be no fundamental incompatibility in applying neural networks to planning problems. Thus, it appears that new applications of neural networks in higher cognitive functions, such as reasoning and planning, would have to achieve some degree of success before such a project could be expected to succeed.

There is so much more study and experimentation to do in neural networks that it is impossible to estimate exactly when an autonomous system constructed entirely of neural network components might become a reality. It certainly appears to be longer than 10 years, putting it into the 21st century – too far in the future to project with any accuracy, and essentially irrelevant with regard to influencing the details of a near-term neural network research program.

4.3 CONCLUSIONS REGARDING NEURAL NETWORK TECHNOLOGY

On the basis of this comparison of neural networks with other information processing technologies, the following conclusions appear to be a fair assessment of neural network technology:

- Neural networks offer significant potential benefits for information processing, such as knowledge acquisition through learning, fast processing speeds, robustness to implementation defects, and compact processors.
- The prime candidates for early neural network applications are expected to be in the areas of pattern classification, simple computational maps for robotic control systems, early vision, signal processing, and speech recognition.
- Neural network technology is not mature enough at present for widespread practical applications, since computer simulations presently are the primary method of implementing neural networks while hardware implementations remain in the experimental stage.
- The first hardware implementations will undoubtedly be functional modules for inclusion in systems using conventional technologies.
- Realizing the potential benefits of neural network technology will require basic research to advance the technology on several fronts, including:
 - Theory, including representations, efficient learning algorithms, stability;
 - Modular architectures, overall system control; and
 - Implementation techniques for silicon and optics.

If a government-sponsored program for the development of neural network applications were to be implemented, the following recommendations are suggested for such a program to realize the potential of neural networks:

- The prime thrust of the initial phase of such a program ought to address research and development efforts to advance the general technology as well as several important generic application areas – rather than focusing on a large, complex neural network system which would have high visibility and rigid schedules and milestones.
- The generic application areas to be pursued should be those where success from the unique neural network approach is likely and would have an important impact. This comparison study indicates that these areas are pattern classification, early vision, speech recognition, signal processing, and robotic control.
- The program should establish a methodology that allows measurement of progress toward goals by providing specified performance criteria, benchmark problems and databases, and review of unsuccessful as well as successful projects, to make best use of experience from the program.
- Periodic review of progress should be carried out, perhaps in one-or two- year intervals, to determine when goals can be adjusted or the program focus can be changed to implementation of practical applications.
- A neural network program should assure good coupling to other branches of information processing, neuroscience, and cognitive science to take advantage of conceptual breakthroughs in the difficult application areas such as vision and speech and in the understanding of particular simple biological neural networks, notably invertebrates.
- Advanced implementation technology should be addressed early in the program in order to understand the problem of matching algorithms to technology, as well as the constraints of silicon and optics, and to support the early development of experimental neural network hardware.

REFERENCES

1. See *Part V, Chapter 3* of this Report.
2. J. Albus, "A new approach to manipulator control: the cerebellar model articulation controller (cmac)," *Journal of Dynamic Systems, Measurement and Control, Transactions of the ASME, series G*, vol. 97, September 1975.
3. J. Albus, *Theoretical and Experimental Aspects of a Cerebellar Model*. PhD thesis, University of Maryland, December 1972.
4. E. Baum, J. Moody, and F. Wilczek, "Internal representations for associative memory," *Biological Cybernetics*, 1988.
5. S. Geman and D. Geman, "Stochastic relaxation, gibbs distributions and the bayesian restoration of images," *IEEE Transactions on PAMI*, vol. 6, pp. 721–741, 1984.
6. R. Gorman and T. Sejnowski, "Analysis of hidden units in a layered network trained to classify sonar targets," *Neural Networks*, vol. 1, pp. 75–89, 1988.
7. S. Grossberg and E. Mingolla, "Neural dynamics of perceptual grouping: textures, boundaries and emergent segmentations," *Perception and Psychophysics*, vol. 38, pp. 141–171, 1985.
8. J. Hopfield and D. Tank, "Computing with neural circuits," *Science*, vol. 233, pp. 625–633, August 1986.
9. J. Hutchinson, C. Koch, J. Luo, and C. Mead, "Computing motion using analog and binary resistive networks," *submitted to IEEE Computer*, August 1987.
10. V. Jennings, "Trainable and adaptable neural networks for robot control," January 1988. Presentation by Martin Marietta to DARPA Neural Network Study, MIT/Lincoln Laboratory.
11. D. Johnson, "More approaches to the traveling salesman guide," *Nature*, vol. 330, December 1987.
12. D. Klatt, "Review of text to speech conversion for english," *Journal of Acoustic Society of America*, vol. 82, pp. 737–793, September 1987.
13. A. Lapedes and R. Farber, "Nonlinear signal processing using neural networks," in *Conference on Neural Information Processing Systems – Natural and Synthetic*, IEEE, November 1987.
14. S. Lin and B. Kernighan, "An effective heuristic algorithm for the traveling salesman problem," *Operations Research*, vol. 21, pp. 498–516, 1973.
15. R. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Magazine*, pp. 4–21, April 1987.
16. C. Mead and M. Mahowald, "A silicon model of early visual processing," *Neural Networks*, vol. 1, pp. 91–97, 1988.

17. M. Miller January 1988. personal communication.
18. W. Miller, F. Glanz, and L. Kraft, "Application of a general learning algorithm to the control of robotic manipulators," *International Journal of Robotics Research*, vol. 6, pp. 84–98, Summer 1987.
19. D. Palmer, "Removing random noise from ekg signals using a back propagation network," November 1987. informal paper from Hecht-Nielsen Neurocomputer, Inc.
20. C. Papadimitriou and K. Steiglitz, *Combinatorial Optimization*. Prentice-Hall, 1982.
21. T. Poggio and A. Hurlbert, "Learning a color algorithm from examples," Memo 909, M.I.T., Artificial Intelligence Laboratory, April 1987.
22. T. Poggio and C. Koch, "Analog networks: a new approach to neural computation," Memo 783, M.I.T, Artificial Intelligence Laboratory, 1984.
23. T. Poggio, V. Torre, and C. Koch, "Computational vision and regularization theory," *Nature*, vol. 317, pp. 314–319, 1985.
24. M. Quillian, "Semantic memory," in *Semantic Information Processing*, (M. Minsky, ed.), MIT Press, 1968.
25. R. Reilly, L. N. Cooper, and C. Elbaum, "A neural model for category learning," *Biological Cybernetics*, vol. 45, pp. 44–69, 1987.
26. Ruck, D.W., Capt. USAF, *Multisensor Target Detection and Classification*. PhD thesis, Air Force Institute of Technology, Computer Science Department, 1987.
27. M. Seibert and A. Waxman, "A neuromorphic vision system for invariant learning and recognition," Tech. Rep. LSR-TR-6, Boston University Laboratory for Sensory Robotics, March 1988.
28. M. Seibert and A. Waxman, "Spreading Activation Layers, Visual Saccades, and Invariant Representations for Neural Pattern Recognition Systems," Tech. Rep., Boston University Laboratory for Sensory Robotics, January 1988. to be published in *Neural Networks*, Fall, 1988.
29. T. Sejnowski and C. Rosenberg, "Nettalk: a parallel network that learns to read aloud," Tech. Rep. JHU/EECS-86/01, Johns Hopkins University, Electrical and Computer Science Department, 1986.
30. T. Sejnowski and C. Rosenberg, "Parallel networks that learn to pronounce english text," *Complex Systems*, vol. 1, pp. 145–168, 1987.
31. T. T. Martin, *Acoustic Recognition of a Limited Vocabulary in Continuous Speech*. PhD thesis, Submitted to Electrical Engineering Department, University of Pennsylvania, 1970.
32. A. Waxman, J. Wu, and F. Bergholm, "Convected activation profiles and the measurement of visual motion," in *Conference on Computer Vision and Pattern Recognition*, IEEE, June 1988.
33. B. Widrow January 1988. personal communication.

34. B. Widrow and M. Hoff, "Adaptive switching circuits," in *IRE WESCON Convention Record, Part 4*, pp. 96–104, 1960.
35. B. Widrow and S. Stearns, *Adaptive Signal Processing*. Prentice Hall, 1985.
36. G. Wilson and G. Pawley, "On the stability of the traveling salesman problem algorithm of hopfield and tank," *Biological Cybernetics*, vol. 58, pp. 63–70, 1988.

Part IV

SYSTEM APPLICATIONS

Edited by Michael Holz

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS	viii
LIST OF TABLES	xi
IV SYSTEM APPLICATIONS	
1. OVERVIEW	1
1.1 Introduction	1
1.2 Neural Network Approaches	1
1.3 Review of Panel Objectives	3
2. NEURAL NETWORK SYSTEM APPLICATIONS – STATE OF THE ART	5
2.1 Perspectives on the Neural Network Field	5
2.2 Applications Selection Criteria	6
2.3 Categorization of Applications	7
2.4 Overview of Invited Contributions	8
2.5 Neural Network Approaches	10
2.6 Description of Selected Applications	15
2.7 Historical Perspectives	17
2.8 Status of the Featured Applications	17
2.9 Near-term Application Potential	18
2.10 Unique Advantages of Neural Networks	20
2.11 Neural Network Attributes	26
2.12 Long-term Goals	27

3. ISSUES IN NEURAL NETWORK RESEARCH	29
3.1 What is a Neural Network?	29
3.2 Forms of Neural Network Research	30
3.3 Magic or Mathematics?	31
3.4 Qualifying Performance	33
3.5 Rating the Quality of Research	33
3.6 Real versus “Toy” Problems	34
3.7 Some Promising Areas	35
4. CONCLUSIONS CONCERNING NEURAL NETWORK SYSTEM APPLICATIONS	37
4.1 Conclusions	37
4.2 Recommendations	39
4.3 Closing Observations	40
APPENDIX A – TEACHING AN ADAPTIVE NETWORK WITH VISUAL INPUTS TO BAL- ANCE AN INVERTED PENDULUM	43
A.1 Introduction	43
A.2 Inverted Pendulum	44
A.3 Previous Work	46
A.4 Current Work	46
A.5 Summary and Future	48
REFERENCES	49
APPENDIX B – GTE PROCESS MONITOR	51
REFERENCES	53
APPENDIX C – SPEAKER-DEPENDENT ISOLATED WORD SPEECH RECOGNITION (E.G., INTEL’S iSBC 576)	55

APPENDIX D – LEARNED CLASSIFICATION OF SONAR TARGETS USING A NEURAL NETWORK	59
D.1 Introduction	59
D.2 Problem and Approach	59
D.3 Network Experiments and Results	61
D.4 Network Interpretation	62
REFERENCES	64
APPENDIX E – A NEURAL NETWORK DECISION LEARNING SYSTEM APPLIED TO RISK ANALYSIS	65
E.1 The Mortgage Origination Underwriter	66
E.2 The Mortgage Insurance Underwriter	67
E.3 The Delinquency Risk Processor	72
E.4 Computational Requirements	73
E.5 Overview of the Nestor Learning System (NLS)	73
APPENDIX F – TRAINABLE AND ADAPTABLE NEURAL NETWORKS FOR ROBOTIC CONTROL	81
F.1 Objective	82
F.2 Approach	82
F.3 Results	86
APPENDIX G – HUGHES PARAMETER NETWORK FOR AUTOMATIC TARGET RECOGNITION (ATR) IN TACTICAL IMAGERY	87
G.1 Problem Addressed	87
G.2 Previous Approaches	87
G.3 Hughes Neural Network Approach	88
G.4 Results	89

G.5 Conclusions	89
REFERENCES	92
APPENDIX H – IMAGE CLASSIFIER	93
H.1 Neocognitron Model Description	93
H.2 Results	96
H.3 Summary	97
REFERENCES	100
APPENDIX I – A MODEL RETINA WITH PUSH-PULL, COOPERATIVE RECEPTORS	101
REFERENCES	106
APPENDIX J – TRW INVESTIGATION OF SCAN-SCAN CORRELATION AND MULTI-TARGET TRACKING	107
J.1 Problem Area	107
J.2 Prior Approaches	107
J.3 Neural Network Approach Used	108
J.4 Results	109
J.5 Future Plan	113
REFERENCES	114
APPENDIX K – MULTIDIMENSIONAL IMAGE FUSION AND SEGMENTATION	115
K.1 The Context-Sensitive Processing of Noisy Multidimensional Image Data	115
K.2 The Boundary Contour System and the Feature Contour System	117
K.3 Emergent Segmentation: Boundary Completion, Noise Suppression, and Regularization	117
K.4 Interaction of Feature Contour System and Boundary Contour System: Fusion of Color-and-Form in Filled-In Regions	123
REFERENCES	128

APPENDIX L – PRESENTATIONS TO THE APPLICATIONS PANEL	129
L.1 List of Contributors	129
L.2 One Page Summaries of Contributions	136
L.3 Questionnaire to Forecast Neural Network Applications During the Next Quarter Century	208

LIST OF ILLUSTRATIONS

Figure No.		Page
2-1	Neural Network Associations	6
2-2	Development of Neural Network Applications	17
2-3	A Neural Network Taxonomy Including Featured Applications	19
2-4	Functional Organization of the Brain	25
4-1	Panel Assessment of Risk and Value for the Featured Applications	41
A-1	Inverted Pendulum	45
A-2	Realization of a Control System for the Inverted Pendulum	45
A-3	Two Examples of Macintosh-generated Pendulum and Cart Images	47
A-4	Learning Curve for a One-unit Network Using Two 5×11 Images as the Input	47
A-5	Generalization Curve for a One-unit Network Using Two 5×11 Images as the Input	48
B-1	Connectionist Learning for a Fluorescent Bulb Manufacturing Process Line	52
C-1	Matched Filter Word Recognizer	56
D-1	Schematic Diagram of the Network for Sonar Target Recognition	60
D-2	Response of Hidden Units to Narrow-band Input Patterns	63
E-1	Mortgage Origination Underwriter: System Agreement with Human Underwriters	67
E-2	Mortgage Insurance Underwriter: System Agreement with Human Underwriters	69
E-3	Mortgage Insurance Underwriter: Comparison of Quality versus Throughput	71
E-4	Delinquency Risk Processor: Delinquency Reduction	73
E-5	The RCE Network	74
E-6	Feature Space Representation of Cell Commitment in the Internal Layer	75
E-7	Threshold Reduction for Active Internal Layer Cells	76
E-8	Examples of Separable and Nonseparable Pattern Classes	77
E-9	An Example of Multiple RCE Networks Achieving Unambiguous Classification for Overlapping Feature Fields	78
E-10	The Nestor Learning System	79
F-1	Cerebellar Model Articulated Controller (CMAC)	83

Figure No.		Page
F-2	Robot Test-bed to Study Sensory-guided Object Acquisition	84
F-3	Method for Teaching by Observation of Human Task Performance	85
G-1	Generic Current-Generation ATR	88
G-2	A Neural Network Process to Recognize Shape	90
G-3	Parameters Networks for Recognizing Occluded Targets	90
H-1	Architecture of the Neocognitron	94
H-2	Mechanism of Shift Invariance in the Neocognitron	96
H-3	Input Patterns Used to Train the System	97
H-4	First S-layer Output Response on Planes 1-3	98
H-5	First S-layer Output Response on Planes 4-6	98
I-1	Photopigment Population-transition Model for the Iris Retina	102
I-2	Temporal Receptor Response to Light Steps	103
I-3	Intensity-dependent Spatial Coupling of Receptors	103
I-4	Intensity-dependent Adaption of Resolution	104
J-1	Input Frames for Multi-target Tracking	108
J-2	Probable Object Motion Distribution	109
J-3	Two-track Example	110
J-4	15-Track Example	111
J-5	Ambiguous Two-Track Example	112
K-1	Emergent Segmentation and Featural Filling-In for Processing of Noisy Multi-dimensional Images	116
K-2	Architecture of the Boundary Contour System	118
K-3	Illustrations of Emergent Segmentation	119
K-4	Examples of Emergent Groupings Discovered by Using the Boundary Contour System	120
K-5	Image Segmentation in the Presence of Changing Noise Distributions	121
K-6	Multiple-Scale Segmentation	122
K-7	Flow Chart of the Feature Contour System	124

Figure No.		Page
K-8	Boundary and Feature Contour System Interactions Giving Contextually Dependent Brightness Constancy	125
K-9	Simulation of the Craik-O'Brien-Cornsweet Effect	126
K-10	Simulation of a Mondrian Image	127
L-1	Level of Support Appropriate for Neural Network Applications Research	210
L-2	Development Time of Neural Networks as a Major Technology	210
L-3	Potential of Neural Networks for Novel Applications	211
L-4	Potential of Neural Networks for Enhanced System Performance	211
L-5	Promising Application Areas for Neural Networks	212
L-6	Promising Application Areas for Neural Network Subsystems	212
L-7	Performance Advantages of Neural Networks	213
L-8	Potential for Replacing Existing Technology	213
L-9	Share of Civilian versus Military Applications	214
L-10	Application Challenges	214
L-11	Potential Social Impact	215
L-12	Fundamental Limits	215
L-13	Limiting Factors	216
L-14	Neural Network Psychology	216
L-15	Applicability to Human Psychology	217
L-16	Human versus Neural Network Processing Potential	217
L-17	Speculative Phenomena	218

LIST OF TABLES

Table No.		Page
2-1	DARPA BAA Survey Results	8
2-2	Summary Chart of Application Areas, Status, and Invited Speakers	9
2-3	Neural Network Approaches of All Surveyed Applications	11
2-4	Potential Neural Network Approaches for the Featured Applications	12
2-5	Selected Innovative Applications I	13
2-6	Selected Innovative Applications II	14
2-7	Neural Network Advantages as Expressed by Interviewees	22
2-8	Neural Network Advantages versus Competing Technologies	23
D-1	Results of the Sonar Classifier for the Aspect-angle Dependent Series	61
E-1	Quality Comparison of the Nestor Mortgage Insurance Underwriter System with Human Underwriters	70

1. OVERVIEW

1.1 INTRODUCTION

“Neural Networks – what are they good for?” This central question guided the Neural Network Study’s *System Applications Panel* (Panel 4) in its survey of the field. A number of experts were invited to provide an overview and introduction, and 55 researchers reported on 77 potential neural network applications. Eleven contributions were selected as primary examples for highlighting the neural network state of the art, its range of applicability, and its future potential. The *functions* performed by these neural networks range from linear and nonlinear mappings, to optimization, to such specialized functions as a model retina, track interpolation, and automatic segmentation of general imagery.

Neural networks, in the strict sense, are to be understood as *electronic networks* which implement algorithms and attempt to emulate certain features of biological neural networks. They typically consist of a large number of highly interconnected but simple processing elements; by varying the interconnection weights according to a “learning rule,” the network is expected to provide a variety of benefits – such as adaptation to changes in the problem specifications, compensation for faults in the network, and improved performance with time. (See *Part II: Adaptive Knowledge Processing, Chapter 2: Background* and *Part I: Introduction, Appendix B: Neural Network Glossary of Terms* for further explication of the definition of neural networks.)

All of the applications and models reviewed by the Study’s System Applications Panel addressed some form of learning by mapping input spaces, usually of high dimension, into some interpretive output space. A practical taxonomy may start by dividing neural network models into feedforward versus feedback information flow classes; this can be further subdivided according to fixed, supervised, or unsupervised learning. At the present time *mature applications* – of which only two have been fielded thus far – tend to be *supervised feedforward* systems with restricted connectivity. Feedforward mapping or classification systems offer the shortest development time for a given problem complexity. This is not an accident; feedback systems of complexity sufficient for practical applications are much more difficult to develop.

1.2 NEURAL NETWORK APPROACHES

Viewed as algorithms, neural networks compete with alternative conventional techniques. (See *Part III: Assessment of Neural Network Technology* for comparisons of neural network and conventional approaches in specific application areas.) Researchers generally attempt to apply neural network techniques to problems for which “classical” approaches either are difficult or cumbersome, or have failed altogether to provide adequate solutions. Often, these are the kinds of problems which biological systems can handle easily. Neural network approaches seem advantageous for *data-intensive* problems and problems dealing with *difficult* and *changing environments*, as they appear in vision, robotics, speech, database and battle management. Specific problems might be, for example, the fusing of many simultaneous data streams in a laser radar or the correlation of targets from scan-to-scan in a scanning sensor system.

It is difficult to evaluate the performance of proposed neural network architectures. Many architectures appear to show “promise” at solving a given problem, yet may – upon further examination – be found to be inappropriate, or at least sub-optimal. It is quite easy to define a “toy” problem that will show excellent performance in simulations, while real data often present subtle quirks that impede performance. Whether being tested with simulated or real data from the field, the *quality of the test* is critical in evaluating the performance of a neural network. There does not appear to be any general method, other than the experience of the researcher or his/her reviewers, for determining in advance whether there are “hidden” problems inherent in a given approach.

Benchmark problems and *standardized databases* need to be developed in the generic areas. While neural network research should not be limited by such databases, their availability could enable comparison of such key metrics as network size, accuracy, computational load, and training requirements.

Speed, parallelism, and ease of implementation were most often cited by the researchers as reasons for using neural networks, although speed and parallelism are not advantages exclusive to neural network approaches. An example of ease of implementation is found in neural network classifiers which can be taught without recourse to specific assumptions or knowledge about the probability distribution inherent in the data.

Self-organization and *unsupervised learning* emerge as additional advantages unique to neural networks. *Small-scale* self-organization is exhibited, for example, by the common supervised learning approaches in the sense that they can, to a limited extent, organize their internal nodes as receptive fields. *Global* self-organization is a much harder problem. Without doubt, the most exciting future applications of neural networks will center on systems with some degree of global self- organization. Again, living systems provide ample evidence of fine-grained, yet global control structures.

What is not clear from the current state of neural network research is whether the kinds of circuits now being studied have capabilities anything like those of the human (or even a frog’s) brain. Researchers do *not* have complete and accurate models of the behavior of biological neurons in isolation or in a network. What they *do* have are a variety of models which can be described in mathematical terms, and which can then be simulated on a computer.

Neural networks – as algorithms – represent “simple,” or deterministic, systems – such as mapping transformers, feature extractors, pattern generators, and dynamical systems. But, in addition, neural network systems can also represent “complex” systems. In a certain sense, neural networks allow the proper realization of physical phenomena which are normally attributed to living systems. Complex systems, in particular, do not permit a complete syntactic description. They can be realized only as complete, autonomous entities for which an inherent distinction between hardware and software is not possible. “Complex system control structures” are at the core of long-term neural network research and are central to the successful development of autonomous sensor systems for reconnaissance and “smart” weapons. The time seems ripe for DARPA to broaden the scope of neural network applications and help drive this field to greater maturity.

1.3 REVIEW OF PANEL OBJECTIVES

As part of the DARPA Neural Network Study, it was the mission of the System Applications Panel (Panel 4) to assess the applications potential of neural networks for future military system needs as well as survey possible benefits in the civilian sector. In securing adequate sources of funding for a new technology area – such as neural network technology – concrete applications potential is of particular concern for DARPA.

In general, a sound assessment of a specific neural network applications potential requires two components:

- A thorough understanding of the neural network field – its foundations, capabilities, and limitations; and
- A complete and detailed appreciation of the needs and restrictions of any particular application.

The former component was the primary concern of Study's *Adaptive Knowledge Processing Panel* (Panel 2), the findings of which comprise *Part II* of this Report, and the *Technology Assessment Panel* (Panel 1), which presents its findings in *Part III*; the latter component, meanwhile, concerned the System Applications Panel (Panel 4) the findings of which are summarized below.

The System Applications Panel heard presentations from several domain experts who could provide an authoritative overview on areas where neural network technology appears promising – notably vision, robotics, speech, battle management, laser radar sensors, and database management.

In an effort to illuminate the issues in a comprehensive way, the Panel sponsored, in addition, a half-day meeting with Richard Feynman and conducted a working lunch with Carver Mead at the California Institute of Technology. These contributions were highly informative and can form the basis for a far-reaching assessment of future neural network applications, especially in realms for which neural network technology has not been applied so far.

For an assessment of present neural network applications, a survey of workers actively engaged in neural network activities seemed more effective. Accordingly, the majority of the speakers interviewed were researchers who have implemented practical neural network applications. Typically, these researchers have applied their neural network expertise to problems which are difficult or impossible to solve by present-day traditional methods. Concise summaries of all 77 contributors are collected in *Appendix L.2*, beginning on page 136. In addition, *Appendix L.3* [p. 208] reports the results of a questionnaire sent to all speakers to help the Panel forecast neural network applications in the longer term.

From the set of practical neural network applications, the Panel selected 11 contributions as primary examples that describe the neural network state of the art, its range of applicability, and its future potential. In-depth descriptions of these featured applications, written by the respective investigators, are provided in *Appendices A – K*, beginning on page 43.

Based on these featured neural network applications, the Panel has attempted to isolate unique neural network advantages in comparison with competing technologies and indicate short- and long-range neural network research and development goals.

2. NEURAL NETWORK SYSTEM APPLICATIONS – STATE OF THE ART

2.1 PERSPECTIVES ON THE NEURAL NETWORK FIELD

The field of neural networks has a special attraction, no doubt, because of its association with the human brain. It promises, it is hoped, to shed some light on the theoretical underpinnings of the workings of all those human faculties – reasoning, vision, speech, locomotion – which have confounded for many years the extensive efforts of many researchers, primarily in the field of artificial intelligence, but also in robotics and the behavioral sciences, for example. After the initial euphoria, a distinct disillusionment has settled in: the view is more prevalent that engineering emulations of such capabilities are not easily achieved and that successful solutions to these long-standing problems may even require a new outlook, perhaps a fresh approach.

Without exception, each researcher interviewed by the System Applications Panel expressed an inspirational fascination in one way or another. Most speakers also felt that research in neural networks will enhance rather than supplant the more traditional approaches. In this context, particularly, it is worth recalling that interest in neural networks was the driving force at the very beginning of applied research into intelligence (artificial and otherwise), and then fell into considerable disfavor; now it basks in the irony of the circle seemingly fully turned. Determining to what extent this now-renewed interest in neural networks is in fact justified, in terms of applications potential, is the main objective of this portion of the Neural Network Study.

Neural networks, collectively, embody and formalize renewed approaches and efforts aimed at solving difficult, anthropomorphically-inspired problems. As indicated in Figure 2-1, strong bidirectional links connect the field of neural networks with research on the central nervous system, artificial intelligence, and robotics. But neural network research also embraces theoretical efforts aimed at understanding, describing, and utilizing the properties of large networks of distributed processors or computing elements; this links it to computer science, automata theory, and to other formalized models of self-organization. *Self-organization* is, indeed, a large field in its own right: biology and the questions concerning development of the central nervous system are closely linked, as are physical systems which exhibit “order from chaos,” as studied in chemistry and physics.

With such diverse connections, it is not surprising to see the interest in neural networks as widespread as it is. One might even wonder whether “neural networks” exists as an independent research effort. This Study, however, firmly attests to the separate existence of the field of neural networks: a major factor in legitimizing neural networks as a distinct discipline is the unifying goal of utilizing knowledge drawn from diverse fields toward the solution of “hard problems” in practical applications.

Neural network research has resulted in new algorithms for data-intensive problems, and is expected to continue to provide more. Neural network research is also aimed at elucidating the problem of control structures for complex systems. This is a field in its infancy: standard control theory, although extensive and useful, typically restricts itself to applications where external perturbations can be regulated by linearized control approaches. “Complex system control structures” are at the core of long-term neural

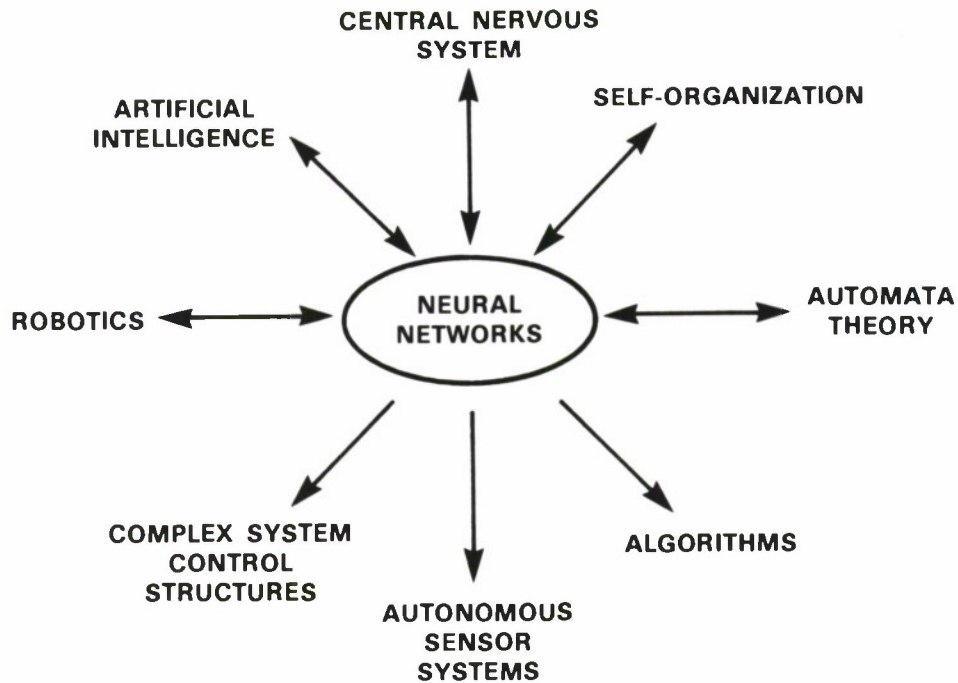


Figure 2-1. Neural Network Associations.

network research and are central to the successful development of autonomous sensor systems for reconnaissance and smart weapons.

2.2 APPLICATIONS SELECTION CRITERIA

Given the great interest in neural networks and their large applications potential, some kind of selection criteria seemed necessary in order to bound the number of people whom the System Applications Panel could interview in a limited amount of time. On the other hand, the major concern for this Panel was perceived as the development of an accurate and fair account of *all* neural network activities which might bear on future applications potential.

With these considerations in mind, it was decided to concentrate speaker selection primarily on workers in the neural network field who have experience with practical applications. In this way, the Panel simultaneously achieved an introduction to the diverse methodologies of neural networks as well as an overview of the neural network application potential. However, the Panel did *not* – according to some prior taxonomy for neural networks – choose selected speakers for their specific neural network approach.

At the outset of the Study, all members of the System Applications Panel submitted a list of suggested speakers; this formed the initial set of invitations. As the Study progressed and its exposure increased, the Panel also learned of many applications by word of mouth. In all cases, a concerted effort was made to follow up on every lead, and no one was rejected who wanted to be heard and could participate.

Notwithstanding these principles, it is not here claimed that the System Applications Panel has completely covered the entire neural network applications field. Several good applications were presented to other panels which are not included here. The Technology Assessment Panel (Panel 1), in particular, focused on the comparison of certain neural network applications with other technologies; discussions of several additional applications can be found in that Panel's findings in *Part III* of this Report.

Based on this open-door policy, maintained throughout the course of the Study, the System Applications Panel is confident in having achieved a valid and comprehensive representation of present-day efforts aimed at distilling current neural network capabilities into practical applications. Moreover, the Panel can now characterize with some confidence the main features of problems where neural network solutions seem advantageous:

- *Data-intensive* problems, as they occur in vision, for example, and
- Problems dealing with *difficult* and *changing environments*, such as found in robotics.

Generally, researchers attempt to apply neural network techniques to those problems for which standard engineering approaches either are difficult or cumbersome, or have failed altogether to provide adequate solutions. Often these are the kinds of problems which biological systems can handle easily. In this regard, the Panel held a two-day presentation session focused specifically on biologically-inspired application approaches.

2.3 CATEGORIZATION OF APPLICATIONS

In an effort to completely survey the field – including workers, primarily in industry, who may have started neural network research recently or who have not published at all in the literature – the Panel asked DARPA for access to the applications database generated by the DARPA-sponsored neural network Broad Area Announcement (BAA). As shown in Table 2-1, the majority of the applications-oriented BAA responses fall into eight categories: vision, speech, sonar, radar, signal processing, robotics, dynamical systems, and cognitive functions.

Accordingly, the contributions to the System Applications Panel have been organized along a similar applications-oriented structure. The categorization exhibited in Table 2-2 is quite self-evident. In this context, the Panel sees *signal processing* as a catch-all category for any applications which do not explicitly fall into the specific areas described by vision, speech, sonar, or radar. *Dynamical systems* include applications which explicitly deal with time-dependent situations and also cover biology-related topics. *Decision systems* include attempts to utilize neural network approaches for high-level functions, such as command & control or battle management – applications which are traditionally addressed by artificial intelligence techniques. *Theory* and *survey* are two additional categories found in *Appendix L.1* (p. 129) in order to accommodate all speakers. [Specific contributions will be referred to by category according to a three-letter mnemonic, for example, THE-1.]

Category	Number of Papers	
	Submitted	Selected
Vision	47	7
Speech/Sonar/Radar	14	4
Signal Processing	11	1
Robotics	14	1
Dynamical Systems	8	2
Cognitive Functions	82	1
Non-Applications	110	9
Totals	268	25

Table 2-1.

DARPA BAA Survey Results

2.4 OVERVIEW OF INVITED CONTRIBUTIONS

Table 2-2 lists the number of all speakers who addressed the System Applications Panel and their application areas. The Panel interviewed 55 speakers who they presented a total of 77 application descriptions.

The applications are separated into four groups:

- **Concept stage.** These are potential applications for which neural network techniques appear promising, but no verification has been accomplished or was detailed to the Panel.
- **Simulation with synthetic data.** This group forms the majority of the surveyed applications, particularly for the vision area.
- **Simulation using real-world data.** This considerably smaller group demonstrates, perhaps most clearly, the application potential of neural network techniques.
- **Hardware development.** Again, a smaller group which attests to the real dedication toward serious neural network implementations of a few workers, primarily in industry.

It is clear from a quick scan of the Panel's survey results that the field of *vision* stands out as embracing most of the neural network applications. As mentioned, the Panel solicited speakers solely for their application potential – independent of category; it is not unreasonable, therefore, to assume that the relative frequencies of occurrence of the individual application areas reflect the interest of neural network researchers as a group. A similar distribution also emerges from the BAA response (see Table 2-1).

In order to quantify realistic near-term potential, the Panel subdivided each category into three groups based on the estimated time to fielding. A completed, functional neural network system may be expected in:

STATUS	VISION		SPEECH		SONAR		RADAR		SIGNAL PROCESSING		ROBOTICS		DYNAMICAL SYSTEMS/BIOLOGY		DECISION SYSTEMS		TOTAL
No. OF SPEAKERS	19	3	2	6	3	6	7	6	3	6	7	6	6	55			
CONCEPT STAGE SIMULATION	5	1		3									2	11			
— SYNTHETIC DATA	21		2	5		5	3	5		5	3	4	4	40			
— REAL DATA	3	3	2	2	2	2	2	2	2	2	2	1	1	14			
HARDWARE DEVELOPMENT	4	2			1	3	2	1	3	2	2			12			
No. OF APPLICATIONS	33	6	4	10	3	8	7	3	8	7	7	7	7	77			
UTILITY <5 YEARS	5	1	1	3	1	5	6	1	5	6				22			
UTILITY <2 YEARS	4			1		2	1		2	1				8			
UTILITY NOW		1										1		2			
INNOVATIVE/PRACTICAL	4	1	1	1		2	1			1				11			

Table 2-2.

Summary Chart of Application Areas, Status, and Invited Speakers

- Less than five years,
- Less than two years, or is
- Fielded now.

As of the beginning of 1988, the last group yielded a total return of only two fielded systems.

Clearly, neural networks constitute an applications field in its infancy, and so the Panel chose to broaden our scope to also include applications which, although still in the development stage, seem to offer a high degree of novelty, usefulness, or near-term application potential. This last group and the group of presently fielded systems are highlighted as "innovative/practical" in the bottom-most row of Table 2-2. It is this group of 11 applications which are fully featured in the *Appendices A – K*.

2.5 NEURAL NETWORK APPROACHES

It is interesting to compile not only the application categories which can be solved by neural network methods, but also the neural network models which have been utilized in practice. Table 2-3 lists the response of all those interviewees who volunteered information about the nature of the model on which their application is based.

For those familiar with the literature, it is probably not surprising that the two most popular approaches are the fully-connected multi-layer perceptron using the backpropagation learning algorithm and the content-addressable memory (CAM) popularized by Hopfield. What is remarkable, perhaps, is the observation that the multi-layer backpropagation network is the only model which has been applied to every listed category, save biology.

In the Panel's selection of the 11 featured applications, the primary concern was with near-term applicability; another concern, however, was to select applications which can also serve as representative examples for the diversity of neural network approaches. Table 2-4 presents a cross-reference of the neural network models listed in Table 2-3 with the set of 11 featured applications. Each featured application stands for a distinct neural network model (●), but could also be addressed by other neural network methods (○).

A summary overview on the 11 featured applications is presented in Table 2-5 and Table 2-6. Categories for each application include name; purpose; special features; the neural network model used, its function, connectivity, the type of learning involved, and its information flow; and the status of the application.

As can be seen, only the *Word Recognizer*, using Intel-developed chips, and the *Risk Analysis* system for a large mortgage applications database, using the Decision Learning System developed and marketed by Nestor, are fielded systems as of the beginning of 1988. A third application, the *Process Monitor*, is a GTE-developed application for realtime yield assessment of a fluorescent tube manufacturing process, and will be coming on line in late 1988.

A brief description of each of these featured applications, status assessment, near-term prospects for fielded systems, and the unique advantages of neural networks are discussed in the sections that follow.

MODEL	VISION	SPEECH	SONAR	RADAR	SIGNAL PROCESSING	ROBOTICS	DYNAMICAL SYSTEMS/BIOLOGY	DECISION SYSTEMS	TOTAL
PERCEPTRON/ADALINE	6	1				1	1	1	3
MULTILAYER BACKPROP		3	3	3	1	1		1	18
AVALANCHE			1	1					2
SPARSE NETS/RCE	1			2			1		4
CMAC					2				2
PARAMETER NET	1								1
NEOCOGNITRON	2								2
BSB				1					1
CAM	4		2		1			1	8
COMPUTATIONAL MAPS	3				2		1	1	7
CELLULAR AUTOMATA	5				1				6
MASKING FIELD	1			1					2
BCS/FCS	2			1					3
ART	1			1				1	3
DYNAMICAL SYSTEM							3	1	4

Table 2-3.
Neural Network Approaches of All Surveyed Applications

APPLICATION	BROOM BALANCER	PROCESS MONITOR	WORD RECOGNIZER	SONAR CLASSIFIER	RISK ANALYSIS	FORK LIFT ROBOT	TARGET RECOGNITION	IMAGE CLASSIFIER	MODEL RETINA	TARGET TRACKING	IMAGE FUSION
PERCEPTRON/ADALINE	●	●	●			○					
MULTI-LAYER BACKPROP	○	○	○	●	○	○	○	○			
AVALANCHE				○							
SPARSE NETS/RCE		○	○	○	●	○	○	○			
CMAC	○					●	○				
PARAMETER NET		○		○	○	○	●	○			
NEOCOGNITRON							○	●			
BSB							○			○	
CAM							○	○		○	○
COMPUTATIONAL MAP				○		○	○	○		○	○
CELLULAR AUTOMATA			○	○				○	●	○	○
MASKING FIELD			○	○							
BCS/FCS								○		●	●
ART I/II							○	○			
DYNAMICAL SYSTEM									○	○	○

● APPLIED MODEL ○ COMPETING MODEL

Table 2-4.

Potential Neural Network Approaches for the 11 Featured Applications

APPLICATION	PURPOSE	FEATURE
BROOM BALANCER	SIMPLE TRAINABILITY BY EXPERT	CONTROLLER WITH VISUAL INTERFACE
PROCESS MONITOR	YIELD PERFORMANCE PREDICTION	MANY-SENSOR NON-STATIONARY DATA
WORD RECOGNIZER	HAND-FREE DATA ENTRY	MICRO-BASED, 0.5 s SYSTEM RESPONSE
SONAR CLASSIFIER	DISTINGUISH UNDERSEA MINE/ROCK	DISCOVERING CLASSIFIER STRATEGIES
RISK ANALYSIS	VERIFIABLE DECISIONS	LARGE DATA SET, RAPID LEARNING
FORK LIFT ROBOT	REAL-TIME ROBOT CONTROL	ADAPTABLE SENSOR INTEGRATION
TARGET RECOGNITION	IR SIGNATURE ANALYSIS	OCCCLUSION TOLERANCE
IMAGE CLASSIFIER	AUTOMATIC FEATURE EXTRACTION	SHIFT/ROTATION TOLERANCE
MODEL RETINA	ADAPTIVE SENSITIVITY/RESOLUTION	AC-COUPLED, SPACE/TIME INTEGRATING
TARGET TRACKING	SCAN-SCAN TRACK ASSOCIATION	SPARSE TEMPORAL DATA INTERPOLATION
IMAGE FUSION	MULTI-MODAL DATA INTEGRATION	CONTEXT-SENSITIVE SEGMENTATION

Table 2-5.
Selected Innovative Applications I

APPLICATION	NN MODEL	FUNCTION	CONNECTIVITY	LEARNING		INFORMATION FLOW			STATUS		
				SUPERVISED	UNSUPERVISED	FEEDFORWARD	RESONANT	SIMULATION	DEMONSTRATION	FIELD	
BROOM BALANCER	SINGLE ADALINE	LINEAR MAPPING	FULL	●		○		<input type="checkbox"/>			
PROCESS MONITOR	TD-UPDATED ADALINES	LINEAR REGRESSION	CUSTOM	●		○				▽	
WORD RECOGNIZER	MATCHED FILTER BANK	CLASSIFICATION	FULL	●		○				▶	
SONAR CLASSIFIER	3-LAYER BACKPROP	NONLINEAR MAPPING	FULL	●		○				△	
RISK ANALYSIS	COUPLED 3-LAYER RCEs	NONLINEAR MAPPING	SPARSE	●		○				▶	
FORK LIFT ROBOT	SINGLE CMAC MODULE	NONLINEAR MAPPING	LOOK-UP	●		○				△	
TARGET RECOGNITION	PARAMETER NET	OPTIMIZATION	CUSTOM	●		○		<input type="checkbox"/>			
IMAGE CLASSIFIER	NEOCOGNITRON	NONLINEAR MAPPING	LAYERED		●	○		<input type="checkbox"/>			
MODEL RETINA	3-STATE MEMBRANE	VISION FRONT END	LOCAL		N/A	○				△	
TARGET TRACKING	TEMPORAL BCS	INTERPOLATION	LOCAL		N/A			<input type="checkbox"/>			
IMAGE FUSION	INTERACTING BCS-FCS	GLOBAL GROUPING	LOCAL		N/A			<input type="checkbox"/>			

Table 2-6.
Selected Innovative Applications II

2.6 DESCRIPTION OF SELECTED APPLICATIONS

In attempting to isolate advantages unique to neural networks and to assess their short- and long-term potential, the Panel's considerations will be discussed with particular reference to the 11 featured applications listed in Table 2-5. Each selected application is comprehensively described in a separate *Appendix* written by the authors. Only a brief overview is presented here.

Three applications utilize simple one-layer neural networks. The *Broom Balancer* by Tolat and Widrow [*Appendix A*, p. 43; ROB-8, p. 171] exemplifies the capability of a *single* neuron – the Adaline – to learn a control problem by “looking over the shoulder” of a human teacher. In the second application, the *Process Monitor* by Sutton [*Appendix B*, p. 51; DYN-1, p. 172], a large number of such Adalines are utilized to monitor and ultimately control a manufacturing line for fluorescent tubes. To be fielded during 1988, this application is considered a substantial improvement over current control techniques in tube production. It demonstrates the capability of neural networks to process non-stationary data in a continuous fashion. Finally, the *Word Recognizer* by Hoff [*Appendix C*, p. 55; SPE-2, p. 153] presents an example of Adalines configured as matched filters for speaker-dependent word recognition. This Intel-developed, single-board digital implementation represents a complete voice-controlled data entry system fielded since 1983 in diverse manufacturing applications.

Of the many potential applications of fully-connected, multi-layer networks which are adapted by the backpropagation learning algorithm, the Panel chose to feature the work of Gorman and Sejnowski on the *Sonar Classifier* [*Appendix D*, p. 59; SON-1, p. 155]. Of particular interest here is the interpretation of the “hidden layer” as self-organized feature detectors which are reminiscent of the clues exploited by human operators. As detailed and novel as this work is, it represents but a first step on the long road toward a fieldable sonar classification system.

As an alternative to backpropagation networks for supervised learning, the Panel offers the Nestor-developed learning system as applied by Collins *et al.* to mortgage underwriting – *Risk Analysis* [*Appendix E*, p. 65; DEC-(2,3), p. 183]. This system is an example of a “network of neural networks” – an approach that seems ultimately necessary for most real-world applications. A central controller automatically assigns and adapts several linked network modules (three-layer “RCE” networks [p. 74] with reduced connectivity) and allocates resources dynamically. A number of different pattern recognition problems have been successfully solved by the Nestor Learning System [VIS-(11,12,13,14,15), p. 145; SPE-3, p. 154]. In the mortgage application, a wide array of evaluation criteria are processed on a large data set of historical loan cases; the system is fielded and has shown superior performance as compared to human loan evaluators.

Yet another approach to the implementation of nonlinear mapping functions by supervised learning is the “Cerebellar Model Articulation Controller” (CMAC) model developed by Albus [ROB-1, p. 164]. Among its features are simplicity (it is essentially a form of look-up table, which results in fast response) and efficient learning with user-controllable generalization (i.e., a single learning example influences nearby points in state space, which allows a flexible system to be taught by few examples). These properties are particularly attractive for realtime robot systems for which the *Fork Lift Robot* by Jennings [*Appendix F*, p. 81; ROB-5, p. 168] is the Panel's featured example. In this application, a CMAC network performs an autonomously guided acquisition task involving several degrees of freedom of robot motion.

CMAC provides the calibration for an array of simple laser-diode-based reflection detectors to map intensity into distance. In effect, the diodes operate as proximity sensors. As with the *Broom Balancer*, this system is initially “taught” by storing the mapping sequences acquired from executing trajectories repeatedly by a human operator.

The remaining applications are primarily vision-related. One approach, *Target Recognition* by Oyster [Appendix G, p. 87], addresses the problem of how to recognize a target hidden in a noisy infrared image by use of *parameter networks*. Such networks provide a convenient approach to specifically connect a multiplicity of features distributed in a multi-level classification hierarchy. Explicit programming techniques similar to those used for rule-based systems establish the network connectivity. A network relaxation cycle achieves winner-take-all voting, which minimizes the impact of noisy or missing portions of the image and provides robust classification decisions. Although in an early development stage, this approach seems to offer near-term applications potential.

All applications discussed so far fall under the category of *supervised* learning systems. *Unsupervised* learning is much harder to achieve. One approach is Fukushima’s neocognitron model, which Menon applied in the context of a tactical *Image Classifier* [Appendix H, p. 93; VIS-6, p. 141]. Although strictly a feedforward system, the neocognitron model represents a technique to self-organize several layers of a hierarchically-structured classification system such that limited shift- and rotation-tolerance is achieved.

Successful vision systems may require an integrated approach from light reception to image interpretation. An example of a biologically-inspired design for an adaptive vision front end is represented by the *Model Retina* developed by Brill *et al.* [Appendix I, p. 101; VIS-1, p. 136]. Cooperative receptors are coupled in such a way as to provide an automatic iris – a light-flux-dependent integration of pixels in space and time.

Scan-to-scan correlation is a problem important in many areas besides vision. Kuczewski developed a *Target Tracking* approach [Appendix J, p. 107; RAD-4, p. 160] based on an extension of Grossberg’s boundary contour system into the temporal domain. Objects whose dynamical constraints are expressed as probabilistic “receptive fields” occupy locations in a space spanned by a cellular automata-type network. The network relaxes toward most likely track assignments in time scales relatively independent of the number tracks present and can naturally accommodate new updated information.

Context-sensitive segmentation of imagery is a problem central to vision. A biologically-inspired model system of considerable complexity for *Image Fusion* has been developed by Grossberg [Appendix K, p. 115; VIS-4, p. 139]. Emergent – i.e., unsupervised – segmentation and filling in of features are carried out by the nonlinear interaction of two parallel modules, the boundary contour system and the feature contour system. In a series of simulations, this coupled system has been shown to automatically accomplish a variety of tasks, such as segmentation, completion, or noise suppression. Efforts are underway at several laboratories to apply this system to the fusion of multi-dimensional real-world data. Numerous parameters must be carefully adjusted to achieve and maintain stability of the system, which is balanced by several feedback loops at different scales. The intricacies of such a system are a powerful reminder of the way that nature has found it necessary to weave a multitude of simple feedback cycles into delicate and yet robust structures which form the basis of autonomous systems adaptable to changing environments.

2.7 HISTORICAL PERSPECTIVES

Considering the long history of the neural network field, dating back a full half-century, the paucity of fielded neural network applications is, perhaps, surprising. To put this question in the proper context, the historical development of the featured applications is traced in Figure 2-2. It can be seen, on a time scale stretching from 1965 to 1990, that most neural network models have their conceptual origins a decade or two before the start of the particular application. With the exception of the *Broom Balancer* and the *Word Recognizer*, all applications shown do not start until 1985, while half again as many applications were not begun until 1986.

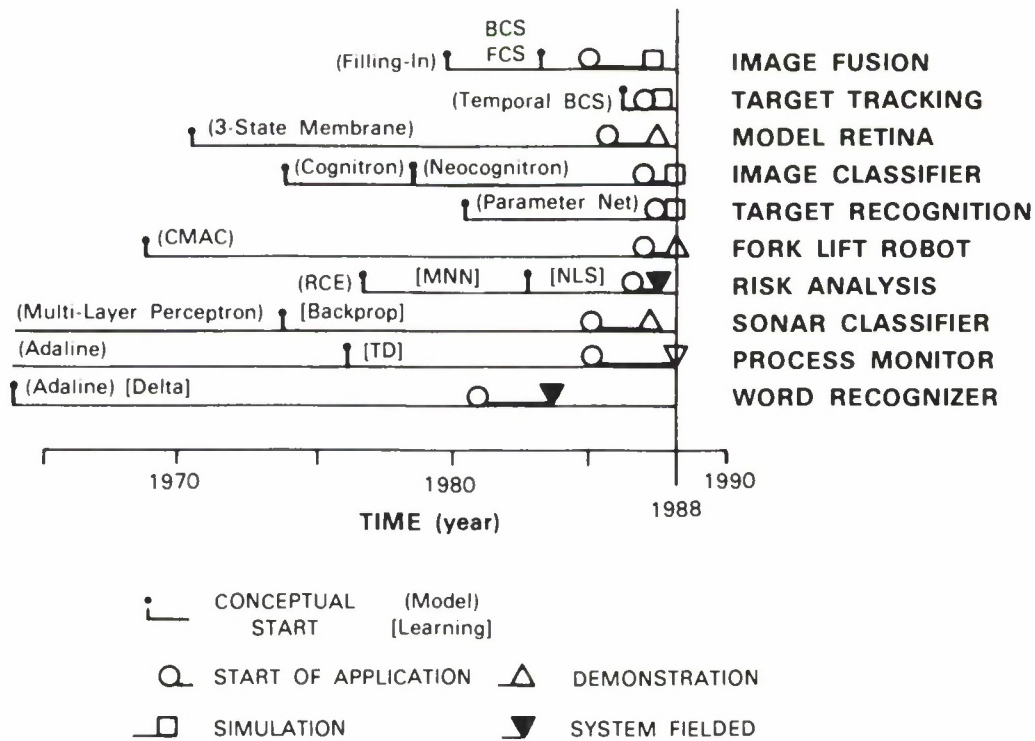


Figure 2-2. Development of Neural Network Applications.

From this point of view, it is not surprising at all to find the number of “finished” – that is, fielded – applications so small. The time seems ripe for DARPA to broaden the scope of neural network applications and help drive this field to greater maturity. Near-term application potential for the featured applications will be discussed in Section 2.9.

2.8 STATUS OF THE FEATURED APPLICATIONS

Taking a global view of the information collected in Table 2-6, it is clear that, in increasing numerical order, the neural network models change from simple to complex. The functions performed by neural networks range from linear and nonlinear mappings, to optimization, and finally to specialized functions, such

as a model retina, track interpolation, and automatic segmentation of general imagery. Meanwhile, the connectivity of the models shifts from full connections, to sparse connections, to local interconnections. In other words, the numerical order reflects, in a sense, a general trend toward increasing problem complexity and specificity. Another important trend is the correlated shift from supervised to unsupervised learning modules or systems, followed by task-independent or autonomous modules. Almost concurrently, the information flow column reflects a change from feedforward to feedback or resonant systems.

This line of reasoning can lead to a practical taxonomy for neural networks. The neural network models are divided into two classes of information flow: feedforward versus feedback. Each of these is, in turn, subdivided according to three types of learning: fixed, supervised, and unsupervised. A graphical representation of neural network models considered in this Study, along with the featured applications, is shown in Figure 2-3. Clearly, most of the applications – and the fielded or nearly fielded systems in particular – are clustered in the *feedforward-supervised* column.

Of practical import for the present status of neural network applications is the observation that *mature applications* tend to be *supervised feedforward* systems with restricted connectivity. The near-term potential of individual applications is discussed in the next section.

Another point can be drawn from Table 2-4 (p. 12), which displays alternate neural network approaches for the featured applications. For the majority of applications, a solution can typically be attempted by more than four alternative neural network models. What neural network model is most suitable for a given application? With the exception of a comparative analysis for classifier networks in the speech domain by Lippmann (IEEE Neural Network Conference, November 8-12, 1987), the Panel remains unaware of other applications-specific neural network model comparisons; such studies seem highly desirable for a mature advancement of the field.

2.9 NEAR-TERM APPLICATION POTENTIAL

In general, one would suspect that the gestation period for neural network ideas would be the longest, while the actual time spent developing an application to maturity is considerably shorter. A case in point is the *Word Recognizer* [p. 55], developed by Intel as a sellable product in less than three years (see Figure 2-2).

Another noteworthy case is the *Risk Analysis* [p. 65] application, which, once its generic NLS (Nestor Learning System) had been developed, was completed by Nestor in less than a year. The development of this system, composed of multiple, complementary networks controlled by a central controller, traces its beginnings to the idea of the reduced Coulomb energy (RCE) network in 1977. Here again, once the foundations were laid, successful applications appeared rapidly. Indeed, the NLS system has been fielded in the context of at least five other applications: a Japanese [VIS-12, p. 146] and other character recognition systems [VIS-13, p. 146; VIS-15, p. 147], a speech recognition system [SPE-3, p. 154], and an industrial parts inspection system for camshaft sorting and automatic transmission stator identification [VIS-11, p. 145].

The Nestor case is, perhaps, typical for all supervised mapping problems: once an adequate learning algorithm has been found, such mapping networks provide a rapid solution to a given problem. This is

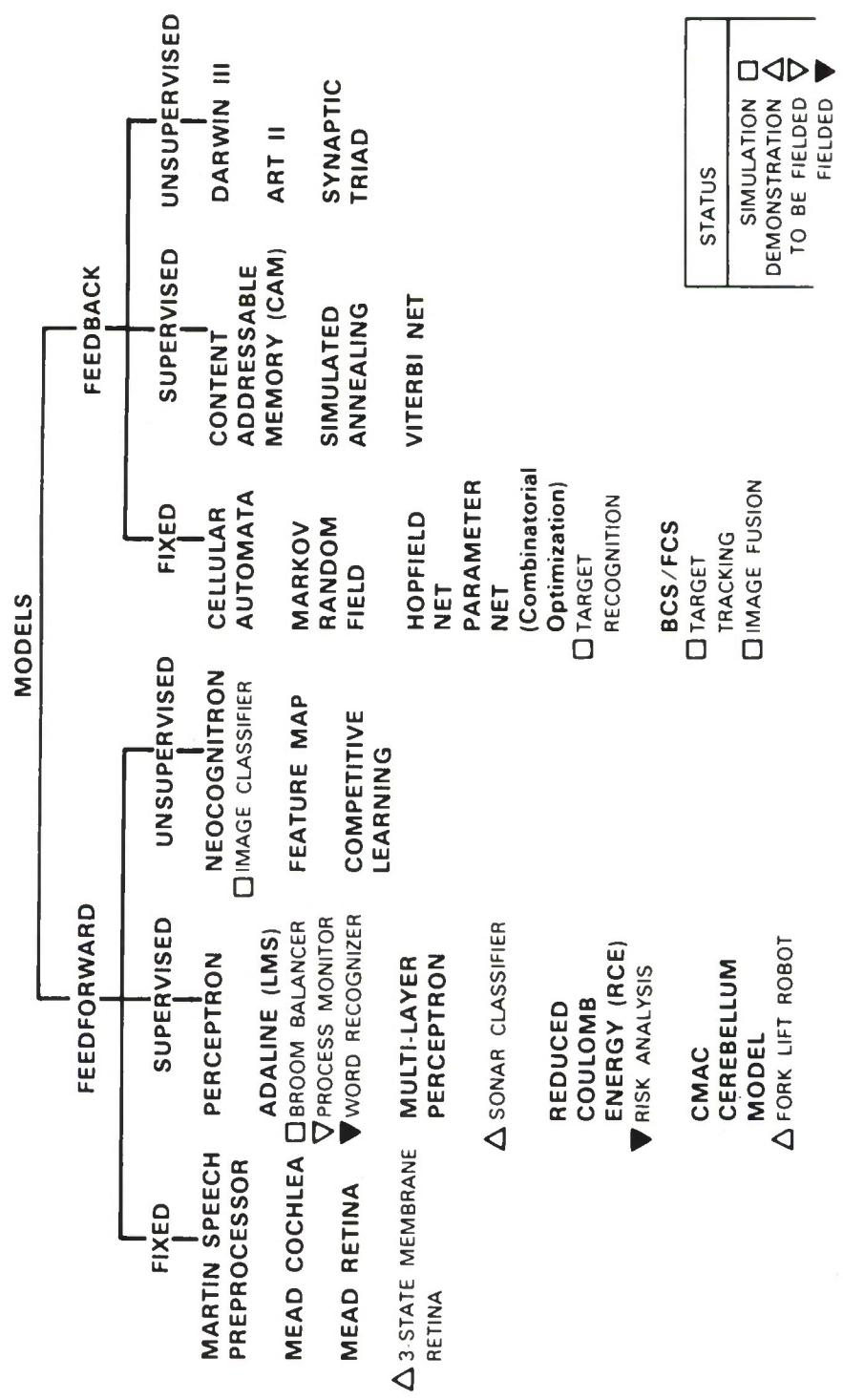


Figure 2-3. A Neural Network Taxonomy Including Featured Applications.

the basis for the great popularity of the multi-layer perceptron, which combines generality with apparent ease of use, thanks to the backpropagation learning algorithm.

The work of Sejnowski (e.g., *Sonar Classifier, Appendix D, SON-1*, p. 59; or “Shape from Shading,” VIS-16, p. 148) is a premier example of the potential flexibility and power of the “backprop” approach.

The development times for fully functional systems of practical interest, however, may be not so short. In the hierarchically structured NLS system, the model can adapt fully to the structure of the problem, and rapid learning is automatically assured. For a backpropagation network, on the other hand, the representation of the problem, the training set, and its size and sequence must be carefully chosen to ensure convergence.

The Cerebellar Model Articulation Controller (CMAC) approach to mapping, developed by Albus [ROB-1, p. 164] in the late 1960s, is another approach to the implementation of general, nonlinear mapping functions. As applied in the *Fork Lift Robot* [p. 81], CMAC offers instantaneous learning, realtime response, and sufficient generalization to adapt a multi-dimensional, uncalibrated sensor array to a flexible object acquisition task. As a hardware-based demonstration of a specific robot task, the *Fork Lift Robot* is completed; but the goal of a multi-purpose, flexible (multi-jointed) robot controller will likely require research efforts well beyond the immediate near-term.

Another application with near-term applications potential is the automatic *Target Recognition* [p. 87] system proposed by Hughes. In this case, a parameter network is utilized; it offers many of the features of traditional rule-based systems and therefore should allow for rapid development time. In contrast to the standard expert system approach, the parameter network approach also includes a relaxation phase to allow for combinatorial optimization of the decision. The need to find a globally stable parameter network may lead to highly application-specific development times, however.

It is worth noting that, in comparing information flow and neural network field status, as shown in Table 2-6 (p. 14), a clear correlation is evident: all applications which have been fielded possess a *feed-forward* information flow. This is no accident; feedback systems of complexity sufficient for practical applications are much more difficult to develop.

From a programmatic standpoint, it seems safe to assert that feedforward mapping or classification systems offer the shortest development time for a given problem complexity. Such problems have been and should be vigorously pursued, primarily by those with specific industry-oriented applications in mind.

The remaining applications noted on the chart will require longer development times for full system applications, since they all belong to the class of advanced research. This work must be pursued if the longer-term potential of neural networks is to be exploited fully.

2.10 UNIQUE ADVANTAGES OF NEURAL NETWORKS

An important component in the assessment of neural network approaches for system applications is the existence of competitive advantages with respect to alternative technologies (see Section 3.3, p. 31). For a young and immature field such as neural networks, it is typically difficult to win head-on against the conventional, established competition. A thorough comparison study of several specific application areas

is included *Part III: Assessment of Neural Network Technology* of this Report; here will be delineated the perceived advantages of neural networks as they were reported by neural network researchers to the System Applications Panel.

A list of neural network advantages, as cited by the Panel's interviewees and ranked according to the total number of quotations, is found in Table 2-7. Speed, parallelism, ease of implementation, and supervised learning rank solidly at the top of the list.

Speed and parallelism by themselves, however, are not advantages exclusive to neural network applications. In Table 2-8, the same list of advantages found in Table 2-7 are presented as they might apply to the 11 featured applications. Each stands for a distinct neural network advantage (●), but entails other perceived advantages (○) as well. In addition, this matrix is contrasted against the advantages offered by "conventional approaches." Clearly, most entries are shared and need to be checked in the final column. Three items, however, emerge as advantages unique to neural networks: *ease of implementation*, *self-organization*, and *unsupervised learning*.

The first, *ease of implementation*, is a matter of judgement, but has been consistently stressed by the very people who ought to know – the workers in the field who have, presumably, struggled with conventional approaches as well. The latter two entries, *self-organization* and *unsupervised learning*, should be considered to genuinely belong to the field of neural networks. Indeed, if another so-called non-neural network approach existed with these attributes, it should probably be seriously examined for inclusion in the neural network field.

Ease of implementation appears to be a feature consistently cited in favor of all supervised learning approaches. Conversely, all these approaches have conventional competition, such as statistical or rule-based pattern classification techniques. The important aspect of neural networks here is the possibility of "programming by example." Of course, as has been discussed before, the notion of easy application development has to be generally qualified, particularly with respect to fully-connected networks (see Section 3.5, p. 33).

Unless the problem has a near-trivial scale, a serious implementation of neural networks will require care, judgement, and intuition – as with any other important problem (see also Section 3.6, p. 34). However, the fact that neural network classifiers, for example, can be taught without recourse to specific assumptions on the nature of the problem, such as a common probability distribution, is a clear-cut advantage of neural network approaches for such applications as mapping transformers and pattern classifiers.

Self-organization harbors a connotation usually reserved for living systems. But the simple cellular automata pioneered by von Neumann teach that "dead" hardware somehow can exhibit lifelike properties by finding a way to maintain structure despite constant change [see Toffoli, SUR-12, p. 205]. This notion is always intriguing. To put things in perspective, it is perhaps useful to distinguish between small- and large-scale self-organization.

Small-scale self-organization is exhibited, for example, by the two common supervised learning approaches – backpropagation and the Nestor Learning System – in the sense that they can, to a limited extent, organize their neural nodes internally as receptive fields [see in particular Figure D-2, p. 63, of the *Sonar Classifier*].

BENEFIT	VISION		SPEECH		SONAR		RADAR		ROBOTICS		SIGNAL PROCESSING		DYNAMICAL SYSTEMS/BIOLOGY		DECISION SYSTEMS		TOTAL
SPEED	6	3	2	4	1	5	1	5	1	5	1	5	1	5	1	5	27
PARALLELISM	5	3	2	4	1	5	1	5	1	5	1	5	1	5	1	5	26
EASE OF IMPLEMENTATION	4	1	3	5	2	5	2	5	2	5	2	5	2	5	2	5	24
SUPERVISED LEARNING	5	3	3	5	1	4	1	4	1	4	1	4	1	4	1	4	20
HIERARCHICAL CONTROL	3					4	1	4	1	4	1	4	1	4	1	4	9
SELF-ORGANIZATION	1					4	4	4	4	4	4	4	4	4	4	4	9
DATA FUSION	4																6
UNSUPERVISED LEARNING	3					1	2	2	1	1	1	1	1	1	1	1	5
GENERALIZATION	3																5
FAULT-TOLERANCE	2						2	2	2	2	2	2	2	2	2	2	4
SPATIAL-TEMPORAL	2																4
LARGE DATA																	3
NOISE IMMUNITY							2	2	1	1	1	1	1	2	2	2	3

Table 2-7.
Neural Network Advantages as Expressed by Interviewees

APPLICATION BENEFIT		BROOM BALANCER	PROCESS MONITOR	WORD RECOGNIZER	SONAR CLASSIFIER	RISK ANALYSIS	FORK LIFT ROBOT	TARGET RECOGNITION	IMAGE CLASSIFIER	MODEL RETINA	TARGET TRACKING	IMAGE FUSION	CONVENTIONAL APPROACHES
SPEED		○		○	○	○	○	●	○	○	○	○	✓
PARALLELISM			○	●	○	○	○	○	○	○	○	○	✓
EASE OF IMPLEMENTATION		●	○	○	○	○	○	○					
SUPERVISED LEARNING		○	○	○	●	○	○	○					✓
HIERARCHICAL CONTROL						●	○						✓
SELF-ORGANIZATION					○	○			○		●	○	
DATA FUSION			○			○	○					●	✓
UNSUPERVISED LEARNING									●				
GENERALIZATION		○	○		○	○	●		○		○	○	✓
FAULT-TOLERANCE			○		○	○	○	○	○	○	○	○	✓
SPATIAL-TEMPORAL			○							●	○	○	✓
LARGE DATA			●			○	○			○			✓
NOISE IMMUNITY			○		○	○	○	○	○	○			✓

● PRIMARY ○ SECONDARY

Table 2-8. Neural Network Advantages versus Competing Technologies

A more impressive example of small-scale self-organization is provided by the *Image Classifier* [p. 93], an application based on the neocognitron. Here, a multitude of neuronal layers are set up to permit unsupervised learning using self-organizing receptive feature fields. The *Image Fusion* example [p. 115] illustrates the property of emergent segmentation, which offers automatic larger-scale groupings of complicated imagery.

How biological systems encode sensory information with self-organizing, spatio-temporal patterns was addressed by several speakers. Gross [BIO-4, p. 177] has initiated multi-channel recording of the electrical and chemical response of neuronal networks consisting of 100-400 cells. Similarly, Eggers [BIO-3, p. 176] plans to study the collective properties of artificially selected nerve cell preparations grown directly on a microelectronic, multi-channel recording substrate. Properties of a network of coupled neurons modeled as interacting voltage-controlled oscillators have been analyzed by Hoppensteadt [THE-2, p. 186]. Shaw [BIO-8, p. 181] sees groups of 30-100 tightly-coupled neurons as the basic building blocks for biological information processors and has modeled conditions for obtaining dynamically stable firing patterns among such interacting groups. A novel theory for unsupervised learning with a general neural network has been developed by Jourjine [THE-3, p. 187]. His work includes an explicit prescription of how characteristic feature fields spontaneously emerge from the fluctuations in randomness present in sensory signals.

Global self-organization is a much harder problem. Preliminary simulations in the general area of computational maps were presented to the Panel by Pearson [BIO-6, p. 179] and Kuperstein [ROB-7, p. 170]. An ambitious simulation of a self-learning robot arm – DARWIN III, which is based solely on environmental feedback using visual and touch sensor integration – was demonstrated by Edelman and Reeke [BIO-7, p. 180; see also *Part II, Chapter 10*]. Without doubt, the most exciting future applications of neural networks will center on systems with some degree of global self-organization.

A closely related area, perhaps somewhat easier to achieve than global self-organization – at least in a limited context – relates to the need for studying *complex system control structures*. Hierarchical control is the center of attention of some neural network research and much artificial intelligence work. Here, *complex system control structures* are more far-reaching and less restricted than hierarchical control.

Again, as so often before, living systems provide ample evidence of fine-grained and yet global control structures. As an example – and as an inspiration – the Panel offers, in Figure 2-4, a modern schematic view of the functional organization of the brain. The closely coupled, distributed nature of the system is apparent.

In a sense, the circle is completed: the matter ends where it began – noting the human brain as the special attractor for the field of neural networks. But perhaps some insight has accrued to the effort: above all, it is clear that real-world problems are rarely easy if they are worth pursuing. It is also clear that anthropomorphic associations need not be taken too literally; nobody wants to build an artificial human brain – nobody *needs* to build an artificial brain. On the other hand, in full awareness of the enormous literature on control theory, it should be appreciated that the structure outlined in Figure 2-4 still is largely an enigma. How much of this enigma needs to be understood in order to satisfy the immediate needs for smart weapons and autonomous sensor systems remains an open question.

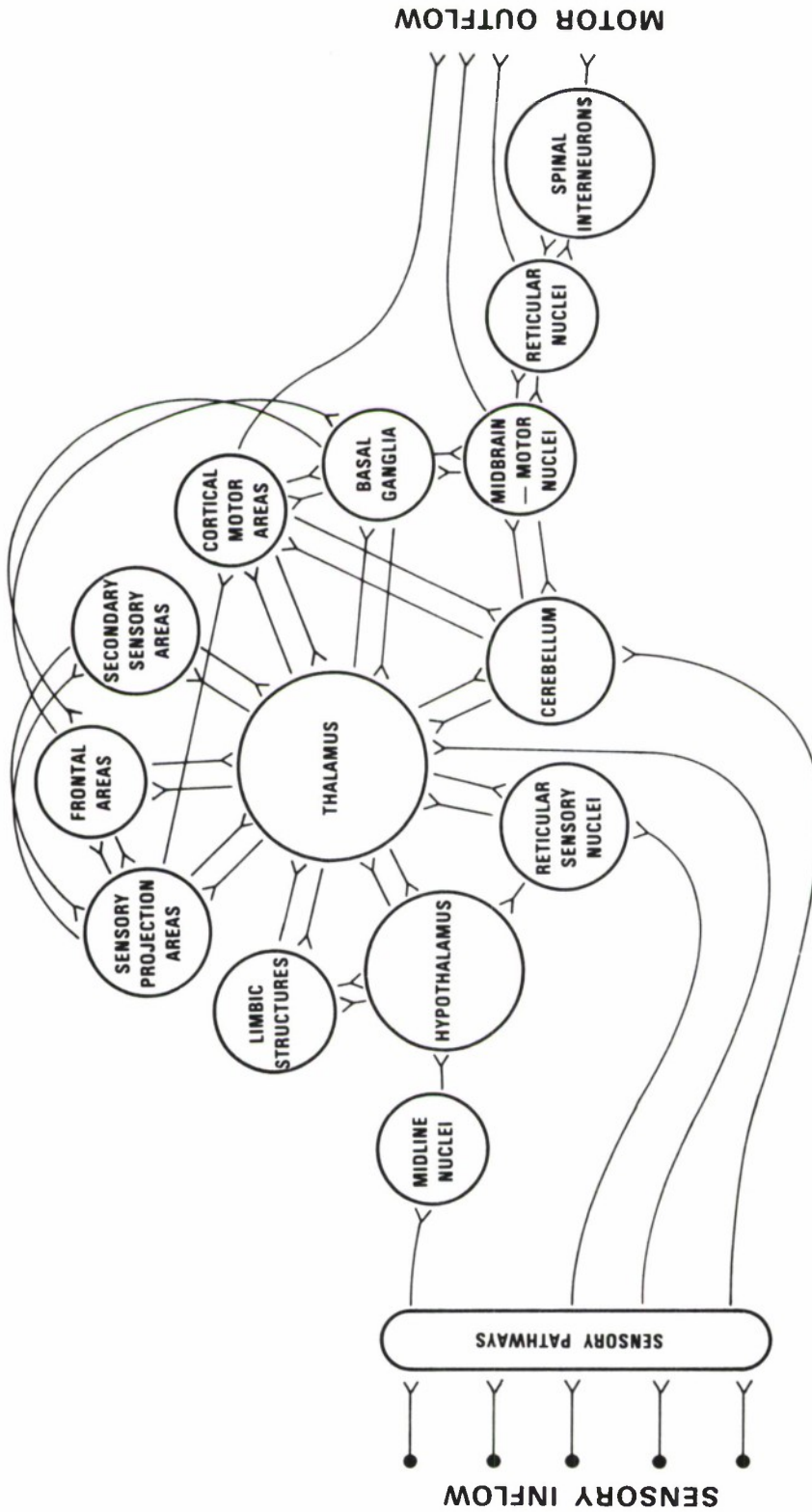


Figure 2-4. Functional Organization of the Brain.

2.11 NEURAL NETWORK ATTRIBUTES

The System Applications Panel concludes this section with a characterization and classification of neural networks which is bold, uplifting, and highly unorthodox. This view is based on the work of Rosen, who is directly connected to the earliest beginnings of neural network research and has followed the field ever since. (Rashevsky launched the field of neural networks in the mid-1930s – McCulloch, Pitts, and Rosen were all Rashevsky’s doctoral students.) To quote Rosen (from *Theoretical Biology and Complexity*, R. Rosen, ed., Academic Press, 1985, p. 165]:

- “Our current systems theory, including all that presently constitutes physics or physical science, deals exclusively with a very special class of systems that I shall call ‘simple systems’ or ‘mechanisms.’
- “Organisms, and many other kinds of material systems, are not mechanisms in this sense. Rather, they belong to a different (and much larger) class of systems, which we shall call ‘complex.’
- “Thus, the relation between contemporary physics and biology is not, as everyone routinely supposes, that of general to particular.
- “To describe complex systems in general, and organisms *a fortiori*, an entirely novel kind of mathematical language is necessary.
- “A simple system can only approximate to a complex one, locally and temporarily, just as, e.g., a tangent plane can only approximate to a nonplanar surface locally and temporarily. Thus in a certain sense, a complex system can be regarded as a kind of global limit of its approximating simple subsystems.
- “Complex systems, unlike simple ones, admit a category of final causation, or anticipation, in a perfectly rigorous and nonmystical way.

“We thus argue, in effect, that any attempt to deal effectively with the material basis of biological organization forces a revamping of our entire traditional scientific epistemology.”

What is the relevance of this for neural networks? As Rosen elaborated in his presentation to the Study [THE-5, p. 189], neural networks can represent “simple” systems, such as mapping transformers, feature extractors, pattern generators, and dynamical systems.

But in addition, neural network systems also represent “complex” systems in the sense defined above. Neural networks, therefore, are the proper physical realization of physical phenomena which are normally attributed to living systems. Complex systems, in particular, do not allow a complete syntactic description. They can be realized only as complete, autonomous entities for which an inherent distinction between hardware and software is not possible.

A further consequence of these arguments is that only for “simple” systems are symbolic algorithms sufficient. In this regard, it is not surprising to find that artificial intelligence efforts have persistently failed to emulate true understanding or reasoning.

It is uplifting to think that neural network research may hold the key to a new science of “complex” systems which relates to “simple” systems in a way which is, perhaps, analogous to the way quantum mechanics relates to classical mechanics.

2.12 LONG-TERM GOALS

The long-range promise of neural network research with a view toward practical objectives can be summarized as follows: Based on an inspiration from biology (which implies, perhaps, a belief in the reality of “complex” systems) the field of neural networks shall . . .

- Develop techniques for dealing with a large sensory data flow,
- Learn representational schemata,
- Understand how biological computational tasks are solved,
- Find control structures for complex systems across a wide range of scales,
- Build autonomous systems capable of surviving in a changing environment.

3. ISSUES IN NEURAL NETWORK RESEARCH

The concept of an electronic “neural network” has existed for more than 20 years. Research on such networks appeared to peak in the 1960s, and then waned, possibly because of a lack of practical methods for implementing such networks; a lack of satisfactory applications and a diversion of researchers into the broader area of artificial intelligence may also have contributed.

More recently, the possibility that neural networks may solve many difficult problems in such areas as machine vision, robot control, speech recognition, etc., has served to revive interest in the field. This spurt of interest seems to be inspired by a variety of developments:

- Advances in integrated circuit technology, which may make fabrication of such devices more practical;
- New algorithms for designing complex multi-layer networks, which may extend their applicability; and
- Discoveries in biology laboratories, which offer new insights into possible network architectures.

(A more detailed construction of the history and evolution of neural network research may be found in *Part II, Chapter 3* of this Report.)

In spite of such progress, there still appear to be very few practical applications for neural networks. Much of the research appears to be motivated by two assumptions:

- If practical implementations can be found for neural networks, then researchers will be motivated to find practical applications.
- If practical applications can be found for neural networks, then researchers will be motivated to find practical realizations.

This “chicken or egg dilemma” appears to occur throughout much of the current activity in neural networks.

3.1 WHAT IS A NEURAL NETWORK?

To some extent, *electronic neural networks* are attempts to model biological neural networks. (The formal definition of ‘neural network’ used by the Neural Network Study and its panels is articulated in *Part I, Appendix B: Neural Network Glossary of Terms* and in even greater detail in *Part II, Chapters 1 and 2* of this Report.) It is known that the biological neural networks that comprise the human brain and nervous system are capable of remarkable kinds of pattern recognition, decision making, and logical inference tasks. While attempts to duplicate capabilities of the human brain using computers (often described by the

term 'artificial intelligence') have generally been failures, could it not be possible that electronic circuits whose architecture more closely resembles that of the human brain may do better at these tasks?

At the same time, it may not be necessary to fully model the detailed organization of the brain. (While studying birds may have helped understand the principles of flight, it has not been necessary to coat airplanes with feathers.) This area is one of some controversy, with researchers differing considerably in the amount of effort they would like to see invested in "reverse engineering the human brain."

Most of the circuits currently considered to be neural networks do have certain similarities. Like biological systems, they are expected to consist of a large number of small processing elements – i.e., "neurons." These neurons are highly interconnected, with the interconnections usually having some variable quality, which permits modifying the performance of the network by adjusting the variable quality. To some degree, this variable quality is expected to simulate the behavior of the synapses which connect biological neurons. The variable nature of the network is expected to provide a variety of benefits, such as adaptation to changes in the problem specifications, compensation for faults in the network, and improved performance with time.

The properties attributed to a single neuron may vary, but, generally, the single neuron is considered to be capable of certain elementary processing of its inputs to generate its outputs. The processing is usually conceived as being much less complex than that provided by one of today's microprocessors, but is performed in parallel and at high speed.

At the state of today's art, electronic neural networks usually exist only in the form of computer simulations. Because neural networks are typically conceived as being very large, the ability to simulate them is generally limited by the speed and storage capacity of available digital computers. Some researchers have developed special hardware to speed up the simulations of certain structures.

3.2 FORMS OF NEURAL NETWORK RESEARCH

There are a variety of topic areas which fall within the purview of neural network research. As a result, the field is one in which several disciplines – biology, psychology, physics, mathematics, engineering – come together. Some of these topic areas include:

- **The study of biological networks of neurons.** These studies include both component level (e.g., how an individual neural cell operates) and system level (e.g., how networks of neurons are organized to perform a given task) research. This work may include the development of models for the behavior of biological neurons.
- **The simulation of neural networks.** Using either models derived from biological studies or hypothetical structures which may or may not possess properties believed to be characteristic of biological systems. These simulations may be used to determine stability of behavior, adaptation properties, learning or memory capacity, characteristics in the presence of faults, etc.

- **The application of hypothetical structures to known problems.** These studies usually consist of using real or artificially created data together with a neural network simulation in an attempt to determine how well a proposed neural network architecture would perform. The neural network performance may be compared with other approaches to the problem solution.
- **The study of ways to implement proposed neural networks.** These studies may lead to new components and/or processes for making them. Also included may be studies of reliability, limitations on performance, expected cost, etc.

3.3 MAGIC OR MATHEMATICS?

Reading some of the literature or listening to some of the researchers in the field of neural networks could lead the naive observer to believe that neural networks are truly magic. The range of problems which neural networks are expected to solve seems awe-inspiring. In some cases, the logic used appears to be based on the notion that “since we haven’t found any other solution, neural networks will have to solve the problem.” Unfortunately, actual performance capabilities obtained from neural network simulations are far from awesome. To some extent, the lack of performance could be attributed to the current inability to simulate very large networks of neurons. After all, the argument goes, “don’t we have our own brains as an existence proof?”

What is not clear from the current state of neural network research is *whether the kinds of circuits now being studied have capability anything like those of the human (or even a frog’s) brain*. It is not presently known which parts of the brain, if any, are “hard-wired” and which are fully self-organized. There does *not* exist an accurate model of the behavior of biological neurons in isolation or in a network.

There do exist, however, a variety of models which can be described in mathematical terms, and which can then be simulated on a computer. In one sense, then, *there do exist neural network algorithms*. Algorithms can be studied and, within the capabilities of present mathematics, can be understood. Therefore, *neural networks might be viewed as a subset of all possible algorithms*. Their applicability to any problem may then be compared with competing approaches based on accuracy and cost.

Thus, the chicken-or-egg dilemma once again becomes important. If neural networks are just a subset of possible algorithms for a solution, is it not unreasonable to believe, *a priori*, that they correspond to the most accurate solution? If they are not the most accurate solution, the basis for choosing them as the preferred solution to a given problem would depend on a *cost/performance trade-off*. But before neural networks can be the most cost-effective solution, a realization for them which is uniquely lower in cost must be found – i.e., the realization must lower the cost of the neural network approach more than it lowers the cost of other competing algorithms.

When figuring the *cost of solution*, more than just the cost of fabrication of the components must be included. For example, the cost of design engineering, field maintenance, etc., should also be included. It may be in these areas – concerning the “ease of implementation” – that the neural network approach will outperform other algorithmic approaches, at least to the extent that self-training, self-healing networks can be built.

To fully achieve such benefits may take new component developments. Experience has indicated that today's computers are not adequate to simulate envisioned neural networks. Improved simulation hardware may make realizations of large networks less expensive, but could negatively impact some of the fault-tolerance expected from neural networks. (For example, if a simulation is achieved by extensively multiplexing a computer processing element, failure of that element might render the entire network non-functional. The predicted graceful degradation of performance under failure would be lost.) However, improved simulation capabilities might still not be adequate to make neural networks the preferred solution to a problem, if such simulation hardware also made competing algorithms more economically realizable.

One school of thought holds that neural networks must necessarily be realized with *analog components* rather than via digital simulations. The use of analog components would, presumably, eliminate the use of shared or multiplexed hardware that is so common in digital systems. Not only might that sharing of hardware be expected to degrade fault tolerance, but the reliability of the digital system may suffer because it requires more components than the equivalent analog system. (Even though a portion of the digital system is shared, several non-shared memory bits are needed to store each "analog" weight value.)

A second perceived advantage of an analog realization is associated with the fact that so much of the operation of currently conceived electronic neural networks consists of adding a collection of weighted input values. Because analog summing circuits are much simpler than the digital equivalent, the analog approach may be the optimal choice.

A counter-argument to analog realizations is that they require the development of new circuit components. A far greater effort is being made throughout the semiconductor industry toward developing new and improved digital components than is being made for analog components. Not only must a new component be developed, but it must undergo extensive scrutiny to establish its reliability and manufacturability. Digital circuits have already passed these hurdles, and digital logic families include components which provide very fast memory and arithmetic capability. Recently introduced digital signal processing components, which offer very high speed multiply-and-add operations, may likely improve simulation of neural networks.

While the availability of neural network components at very low cost should inspire their usage, there are questions one may ask about how "elastic" the market for the product might be. Generally *markets are not infinitely elastic*. Lowering the cost of a component may expand its usage but seldom creates a market for it. For example, a market for integrated circuits was established when ICs cost ten dollars a gate – although the market became far larger at a few cents per gate. On the other hand, billion dollar markets have been predicted for speech recognition products, yet the actual market results have remained in the \$10 to \$20 million range in spite of price cuts. Apparently, the performance of these systems does not justify their usage at any price except in a few special cases.

This view of market elasticity may lead to the question: why is there not more usage of neural network techniques via microprocessor-based simulations, application-specific integrated circuits, etc.? At what cost and performance will the market for such products develop? If these thresholds could be better predicted, it might be possible to better predict the time and cost needed to develop *practical* neural network systems.

One scenario which may occur when using neural networks in algorithmic form is the *discovery of a non-neural network solution* once the behavior of the neural network is understood. The study of those

situations where a neural network does not perform well may give clues leading to a more accurate conventional solution. It is possible that neural networks may have utility in helping researchers learn to understand difficult problems even if they are not used to implement the ultimate solution.

3.4 QUALIFYING PERFORMANCE

One of the most difficult problems faced by current researchers is that of evaluating the performance of proposed neural network architectures. Many architectures appear to show “promise” at solving a given problem, yet may upon further examination be found to be inappropriate, or at least suboptimal.

Whether being tested with simulated data or real data from the field, the *quality of the test* is critical in evaluating the performance of a neural network. The manner of encoding the data can critically affect the performance of the network.

There are methods for allowing complex functions to be realized by neurons. Inputs may be derived from polynomial terms and/or cross products, and multi-layer structures may be used. All have the disadvantage of rapid growth of complexity as the number of input variables is increased.

Once large neural networks are attempted, the performance and limitations are far less obvious, and even experienced researchers can get carried away by their “encouraging first experiments.”

3.5 RATING THE QUALITY OF RESEARCH

The existence of early and encouraging results from many neural network experiments makes judging the work very difficult. It is often quite easy to define a “toy” problem that will show excellent performance in simulations. Real data often have many subtle quirks that make performance much harder to evaluate.

In many cases, the researcher is forced to scale his experiments to much smaller systems than would be needed to process real data, because of limited simulation capacity. It is not always obvious how costly in time, etc., the expansion of a network may be. Researchers, in their zeal, may overlook the impact of *dealing with large numbers*. Consider an example:

A fairly common problem proposed for neural network studies is the development of a pattern recognition system which will be invariant to position of a pattern on the retina. Rather than wire the system so such invariance is designed in, the researcher proposes to *train the system to generalize* – i.e., to learn that its response is supposed to be invariant to position. Assume further that the researcher plans to use a *fully-connected network*, in that all neurons connected to the retina have inputs from all possible retina spots. If the retina is a rectangular array of $m \times n$ spots, there will be $m \cdot n$ inputs to each neuron connected to the retina.

If the retina is so connected, there is no bias to show the neuron which spots are adjacent to which other spots. One could scramble the connections from retina to neural network without affecting the trainability of the network. Using this model, the lateral displacement of a pattern on the retina corresponds to a particular permutation of the inputs.

If the retina is imagined to be toroidally connected in each of the x and y dimensions, there are exactly $m \cdot n$ permutations of the input spots which correspond to lateral displacement of the pattern. Unfortunately, there are $(m \cdot n)!$ total permutations, of which all but the aforementioned $m \cdot n$ do not correspond to a lateral displacement. How many patterns will be necessary to teach this system that of $(m \cdot n)!$ permutations $m \cdot n$ – and only $m \cdot n$ – are acceptable? Since this arrangement has no bias toward any particular permutation, it is reasonable to expect that not only $m \cdot n$ acceptable permutations must be shown to the system, but also a representative sample of all permutations considered unacceptable, such as, for example, those that would transform a “T” into an “L.”

Even if the network is capable of realizing the desired function, $(m \cdot n)!$ is such a large number, that it could easily take longer than the life of the universe to do the training. (Note, for a modest 100×100 retina, the number would be $10,000!$. Written, such a number is roughly equivalent to 3 followed by 35,659 more digits.) While the patience of the researcher who sets the solution of this problem as his/her goal might be admirable, it is questionable whether it is in the best interests of the nation to have the next missile design dependent on his/her success.

In spite of the large numbers cited above, the design of a neural network which will have the desired invariance is not necessarily impractical. Consider Fukushima’s *neocognitron*, a multi-layer network in which each layer contains subcollections of neurons. Each subcollection is a set of neurons such that each neuron covers a small region of the retina – yet the subcollection covers the entire retina. Training is done such that all members of a subcollection will have identical weights. In this manner, each subcollection becomes a “feature detector,” but has an output which is invariant to the position of the feature on the retina. While it may be too early to determine whether this approach will solve the problem, it should be obvious that it has a much better chance for trainability to invariance than the fully-connected array.

Although different approaches may have greatly differing chances for success, as yet there does not appear to be any method, other than the experience of the researcher or his reviewers, for determining in advance whether there are “hidden” problems inherent in the approach. This lack of method makes judging proposals very difficult.

3.6 REAL VERSUS “TOY” PROBLEMS

Evaluating the performance of a neural network approach may not only be costly in terms of simulation time, but also requires source data which adequately represents the type of problem the network is expected to solve. In selecting data for use in simulations, the researcher must be careful not to bias the results. In general, the best data will come from real-world situations.

Too often, sufficient real-world data are not available to the researcher to allow testing the neural network in a lifelike situation. The creation of artificial data may allow simulation of the network, but can seriously bias the results, if the artificial data lack the important variants contained in real data.

The amount of data needed to adequately exercise a neural network may also be difficult to predict. To some extent the number of weights in a network should give some clue to its “learning” capacity. The researcher should be wary of giving too much weight to results obtained from experiments in which

the number of training or testing samples is orders of magnitude less than the numbers of weights in the system.

One approach to improving the ability to judge the quality and effectiveness of neural network research may be the development of standardized problem databases. This technique has proved useful in the area of speech recognition. While neural network research should not be limited by such databases, their availability could allow comparison of various network designs on the basis of error performance, training requirements, etc.

3.7 SOME PROMISING AREAS

In spite of the rather negative viewpoint expressed above, there are some areas where neural networks, or simulations thereof, would appear to have real advantages.

In the control of *robotic modules*, it is very difficult to predict in advance all of the non-ideal behavior of the elements. The effects of friction in joints, warpage and bending of segments under load, inertial mass distribution, etc., might be measured and computed, but at considerable cost. Once the basic distortions and their impact on performance are understood, it would appear possible to design a self-optimizing system which can compensate for the imperfections. Even if the self-optimizing system worked less perfectly than the fully analyzed one, the cost should be so much less that it would still be the preferable choice.

Improved *understanding of the capabilities of different neural network structures* is providing much better insight into neural network behavior. In some cases, certain neural network structures can be shown to be equivalent to known classical pattern classifiers. Here, the availability of truly economical neural network hardware would provide a lower cost method for implementing the classical procedure.

Already mentioned above are efforts toward developing *visual-data processing systems*, such as position-invariant pattern recognizers. While these efforts are still in an early stage of exploration, there would appear to be much promise in this area. While the payoff from a successful solution could be very large, there are still many unsolved problems in this field. Further studies of biological neural networks, emerging from the biological laboratories, may give clues as to how systems for visual data processing should be organized, and may help us design better pattern classifiers. In some cases, the structures appear to be more hard-wired than adaptive, but still may lend themselves to neural network realizations.

Visual data processing not only includes pattern recognition, but also a variety of *related functions* such as estimating the location and velocity of visually presented objects, choosing which objects to track, and using visual data to aim an imaging device. All of these problem areas may lend themselves to neural network solutions.

4. CONCLUSIONS CONCERNING NEURAL NETWORK SYSTEM APPLICATIONS

The opportunity provided System Applications Panel to review many neural network applications provided a valuable perspective for drawing general conclusions and making specific recommendations for the pursuit of future efforts in neural networks. The results of the Panel's efforts allow one to identify near-term and future system applications that could benefit most from neural networks. In this regard, Carver Mead [SUR-9, p. 199], the inventor of retina and cochlea chips, made the following observation during a Panel meeting:

“In a field like neural networks, one is usually too optimistic in the short run, but one is never optimistic enough over the long run.”

With this in mind, the following conclusions and recommendations are aimed at fostering realistic short-term applications while at the same time enabling the broadest possible opportunities for future applications.

4.1 CONCLUSIONS

Neural networks provide significant advantages in solving processing problems that require realtime encoding and interpretation of relationships among the variables of high-dimensional spaces.

Vision, speech, and robotics are prototypical of such processing problems. Other processing problems that can have the same characteristic are the fusing of many simultaneous data streams, the correlation of targets from scan-to-scan in a scanning sensor system, and certain problems of optimization.

All applications and models reviewed by the System Applications Panel addressed some form of the problem of learning and/or implementing mappings from input spaces, usually of a high dimension, into some interpretive output space. Often, the mappings transformed points from some N dimensional Euclidean space R^N , into some M dimensional Euclidean space, R^M .

The *Sonar Classifier* [Appendix D, p. 59; SON-1, p. 155], for instance, uses sonar returns to distinguish between rocks and metal cylinders submerged in sea water. The input space is all possible sonar returns from these objects, and the output space is one of two possible states corresponding to classifying the source of the return: a rock or a cylinder. The classifier partitions the power spectrum of each input signal into 60 bins, and feeds the numerical value of the power in the bins into 60 input neurons. Thus the sonar signal classifier is set up to learn and implement the map on R^{60} that correctly classifies sonar signal returns.

Likewise, the shape-from-shading [VIS-16, p. 148] network encodes the boundary information in a visual scene that arises from the shading data (or, in general, the texture-gradient data) contained in the scene. The input space is the set of all possible visual images. The output for a given image is the identification of the boundaries around all the objects in the image. Thus the output space is the set of

boundary segmentations associated with the images in the input space. Representing the image as pixels, and the boundaries as sets of active neurons corresponding to the pixels that form object boundaries, the input space is R^N , where N = number of pixels (multiplied by any color or grey levels associated with each pixel) and the output space is R^M where M = number of neurons required to form all possible object boundaries. The network is designed to implement the mapping $R^N \rightarrow R^M$ that maps images into their object segmentations. Because M and N are large, possibly millions or more, the input space, the output space, and the quantity of computations required of the network are all very large.

Speed, parallelism, ease of implementation, and ability to learn were most often cited by the researchers heard by the panel as reasons for using neural networks. It is noted that *all* of the reasons cited are related in some fundamental way to the suitability of neural networks to adequately learn and/or implement $R^N \rightarrow R^M$ mappings.

Neural network applications for commercial products are expected.

While most applications are still in early stages, this situation is likely to change as the many workers that are currently in the field bring their applications to maturity. Optimism for applications and for the field of neural networks ran high among the researchers interviewed.

The Panel's interviews demonstrated that the field of neural networks has progressed at different rates in different areas. In some small measure, the field has moved past research and has become commercial. Nestor's applications [Appendix E, p. 65; VIS-(11- -15), p. 145; SPE-3, p. 154] and Intel's *Word Recognizer* [Appendix C, p. 55; SPE-2, p. 153], are examples of work that has progressed to commercial use.

The field is developmental, although for the most part, some workers are no longer asking research questions, but are instead developing products to be put to practical use. The GTE factory *Process Monitor* [Appendix B, p. 51; DYN-1, p. 172], is another example of such an application.

It is recognized from this review that the field is still in the research phase, and workers are asking questions whose answers will determine what the future applications will be. The System Applications Panel saw a wide variation in the quality of results across the neural network arena. This is no surprise; it demonstrates the newness of the field, the arrival of new workers and new ideas, and the different levels of difficulty in the problems addressed.

But it became clear as a result of the Panel's work that few rigorous efforts have been made to compare the performance of neural network methods against other methods or against other neural network approaches. This is largely due to the fact that the field is still in the early stages of experimentation and invention rather than the later, more mature stage of an engineering science. The provision of procedures and institutions that will promote the development of neural networks into an of engineering science with a complete conceptual framework would go far towards removing its deficiencies and fostering real progress in the growth of neural network applications. (*Part III, Chapter 3* of this Report contains the Neural Network Study's *Technology Assessment Panel's* comparison of neural networks and other technologies; comments on the present state of a neural network theory, or conceptual framework, can be found in *Part II, Chapter 12* of this Report.)

Neural networks may offer significant benefits to future defense systems.

Advanced *vision/image processing* and *robotics capabilities* are understood to be of critical importance to future defense systems. Because of the nature of the high-dimensional spaces involved, neural networks can play an important role in addressing the processing requirements for both vision and robotics.

In robotics, neural networks enable the learning of functions that map the space of sensor data concerning the environment and the robotic internal system into the space of robotic manipulations. These mappings enable the implementation of dynamic robotic manipulations that have not yielded to conventional techniques and have prevented critical progress in the field. Results in robotics were among the most encouraging heard by the panel.

In vision, neural networks offer both the implementation of complex mapping functions, such as the boundary segmentation of visual scenes, and the learning of classification functions, such as the recognition of objects from their shape. More vision problems were addressed by the researchers heard by the Panel than any other problem area.

Other areas – such as *optimization*, *sensor scan-to-scan correlation*, *speech*, and *data fusion* are also benefitted by neural networks, and potentially offer high payoff to defense systems.

4.2 RECOMMENDATIONS

In recognition that the importance of neural network technology lies in its promise for the future – not in its accomplishments to date – and in order to foster and to hasten the development of a strong and useful field, the following recommendations are offered by the System Applications Panel:

Pursue the exploitation of existing neural network learning algorithms and develop new ones.

Opportunities for this are widespread, and can provide a valuable impetus to the neural network field. Examples of such opportunities among the applications reviewed by the Panel are Nestor's *Mortgage Loan Evaluator* and Sejnowski's *Sonar Classifier*. Problems pursued should be those in which:

- An $R^N \rightarrow R^M$ map is required to solve the problem.
- Examples of the map are readily obtained.
- The map is not known explicitly and is expensive to derive by conventional means.
- A means for learning the map with a neural network learning algorithm can be readily identified.

Provide the infrastructure required to enable neural network technology to evolve from a realm of ad hoc experimentation to a field of engineering science.

Such infrastructure should include:

- Databases for generic applications areas, e.g.:
 1. Image libraries,
 2. Speech and sound libraries,
 3. Robotic task sets.
- Benchmark problems in the generic areas that enable comparison of key metrics such as network size, accuracy, computational load, and training time.
- Methods for evaluating research and demonstrations, including:
 1. Standards for reporting results,
 2. Techniques for comparing approaches.

Focus research and development activities to promote high-payoff, generic applications in the areas of vision, speech, and robotics.

Undertake immediate activities towards the development of key neural network subsystems in each generic area. Certain subsystems are critical to applications in their generic areas because a broad set of potential applications depends on prior development of these subsystems. Among these key subsystems are:

- A realtime, robust visual scene segmentation system.
- A robotic manipulator that learns its system dynamics through practice, and learns its required behavior by example and practice.
- A connected speech recognizer.

Other key subsystems may also need to be defined and pursued.

4.3 CLOSING OBSERVATIONS

From the perspective of the System Applications Panel, the applications reviewed varied in their intrinsic value as well as in the risks associated with their further development. Figure 4-1 shows how the Panel rated the risks of further development as a function of intrinsic value for the 11 featured applications. These applications tend to cluster in the high-risk, high-value corner of the figure, with the vision-related applications occupying the extremum. None of the 11 applications occupy the most desirable corner of the figure – the high-value, low-risk corner.

However, further progress on the high-valued applications would tend to move them toward the desirable corner, and in fact their current position of higher risk is due at least in part to the fact that these

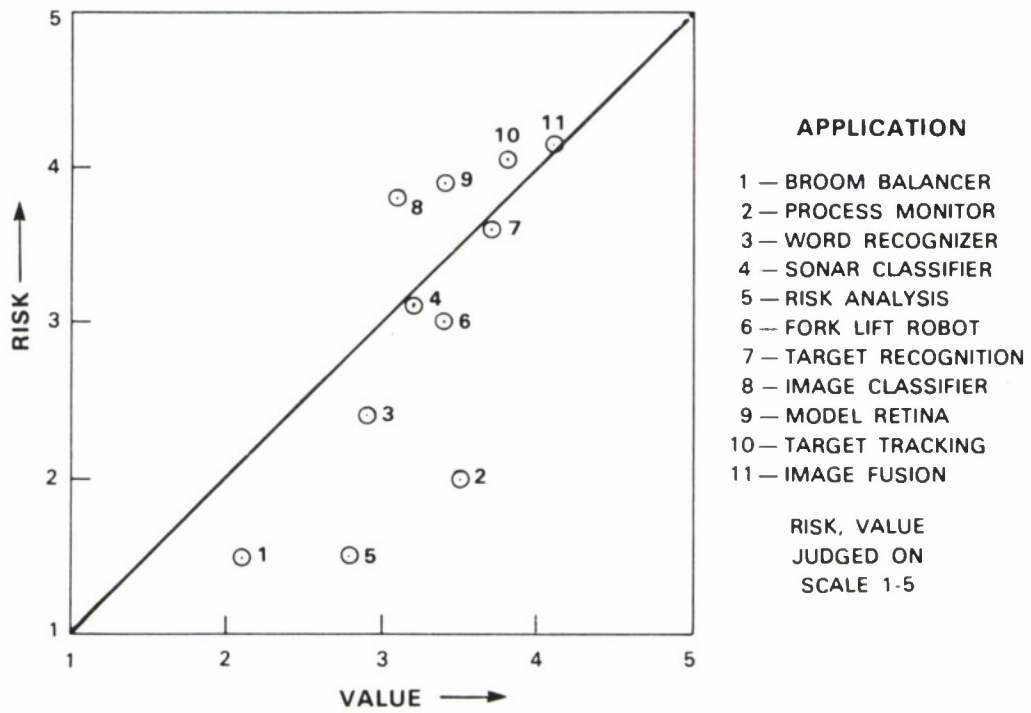


Figure 4-1. Panel Assessment of Risk and Value for the Featured Applications.

applications require more time and effort in their development than some of the others. Thus, for example, the vision applications – which were judged by the Panel to be the most valuable – are also the ones that take the longest to develop.

It appears unlikely that most neural network applications will be “pure plays” employing a stand-alone neural network. Rather, it is expected that neural network applications will actually be integrated systems, with neural networks playing a major role. For example, general automatic recognition devices would almost certainly require a neural network to perform the function of scene segmentation. And, likewise, recognizing portions of the segmented scene might also require a neural network. But other crucial parts of such a device (focus-of-attention, tracking, and evaluation) could probably be handled by non-neural methods.

Important applications requiring neural networks tend to be substantial engineering projects, involving many technologies other than neural network- based ones, and they bring with them technical problems that are not due to neural networks *per se*. For this reason, it is important to keep early neural network developmental activities focused on problems for which the major engineering challenges are primarily neural network-oriented. The key neural network subsystems suggested above are intended to be of this kind. Thus, not only are they important to a wide area of applications, but they also tend to promote rapid technical progress by concentrating effort primarily on neural network issues.

It has been said that neural networks are an answer looking for a problem. The consensus of researchers heard by the System Applications Panel would appear to conclude otherwise. Indeed, it would appear that problems of significant national importance in the realms of vision, speech and robotics are problems looking for an answer, and this Panel has determined that neural networks can provide an important part of that answer.

APPENDIX A

TEACHING AN ADAPTIVE NETWORK WITH VISUAL INPUTS TO BALANCE AN INVERTED PENDULUM

Viral V. Tolat, Dr. Bernard Widrow
Stanford University
Stanford, CA 94305

This work was supported by the NASA Center for Aeronautics and Space Information Systems (CASIS) under grant NAGW 419

A.1 INTRODUCTION

Balancing a broom stick in the palm of one's hand is a fairly complex task from a control point of view. However, for those of us who have a modicum of coordination it is a seemingly simple task. If asked how we do it, most of us would be hard-pressed to come up with an accurate account. An analogous problem is that of balancing an inverted pendulum fixed to a non-stationary platform. If the task of balancing such a system were presented to a person in the form of a video game, that person would probably master the game in a short period of time. Of course, to master this game, we have available to us the most sophisticated adaptive network around, the human brain.

Adaptive networks are of interest because they represent an attempt to simulate or duplicate some of the basic functions of biological brains. They have many interesting properties – for example, they are inherently parallel, fault tolerant, they can learn from examples, and they can generalize. By being parallel, functions that might take a conventional computer seconds or minutes to compute can be computed in real time by adaptive networks. Because of the distributed way in which information is stored in the network, in the interconnections, local damage to the network will cause only minor degradation in the operation of the entire network – as opposed to totally disabling some arbitrary function of the system. Unlike current expert systems, which have to be explicitly programmed and for which algorithms that implement the function must be developed, the training process for an adaptive network is relatively simple. By utilizing learning algorithms, a network can be taught to perform a function by being presented with example inputs and the correct associated responses. Adaptive networks have an innate ability to produce a correct response to input combinations for which the network has not been trained – i.e., they can “generalize” or “infer.” This is a very useful trait, since, in many applications, it is unreasonable to assume that a system can be trained on all possible inputs.

The research objectives are fourfold:

1. To demonstrate that a person with the skill to stabilize an inverted pendulum can train an adaptive network to perform this control task.

2. To demonstrate that an adaptive network can stabilize an inverted pendulum by performing visual dynamic processing to extract the critical state information (pendulum angle, pendulum angular velocity, cart position, cart velocity) from coarsely-quantized images of the pendulum and cart without being “told” what that state information is.
3. To demonstrate that an adaptive network can generalize – i.e., make correct decisions and produce correct responses to inputs it has not been specifically trained on – after training with a limited number of other sample inputs.
4. To demonstrate that an expert can impart learned skills to an adaptive network in complex problem environments for which it may not be possible to develop explicit decision rules.

The first three of the above objectives have been realized and the results are discussed below. The fourth objective is long term. It leads toward a new form of man-machine interaction in which a person trains a machine to perform a task by having the machine “look over the person’s shoulder” and observe the environment and the person’s responses to the environment. After limited training, the machine will be able to perform the task independently of the person.

The next section describes the inverted pendulum problem and discusses why it was chosen as a basis for research. The section after presents the previous work done on the inverted pendulum problem. This is followed by a presentation of the most recent work and results. The last section summarizes the effort and discusses future research.

A.2 INVERTED PENDULUM

In the inverted pendulum system illustrated in Figure A-1, four variables describe the state of the system: the position of the cart (x), the velocity of the cart (v), the angle of the pendulum (θ), and the angular velocity of the pendulum (ω). The force required to stabilize the system is

$$F = U \times \text{sgn}[k_1 x + k_2 v + k_3 \theta + k_4 \omega], \quad (\text{A.1})$$

where U is a constant representing the magnitude of the force to be applied to the system and the coefficients k_1, k_2, k_3, k_4 , are derived from the physical characteristics of the pendulum system and optimal bang-bang control theory. A realization of Equation A.1 is shown in Figure A-2. In an automatic system, sensors on the cart and pendulum feed the state variables into the circuit of Figure A-2, whose output mechanically drives the cart, thus closing the control loop.

There are several reasons why the inverted pendulum was chosen as a basis for adaptive network research. First, the inverted pendulum problem is a classical control problem which has been extensively studied and is well understood. The inverted pendulum problem is representative of many other control problems and therefore an understanding of how to use adaptive networks to balance the pendulum will allow us to solve other control problems. By using a simple visual image of the cart and pendulum, we

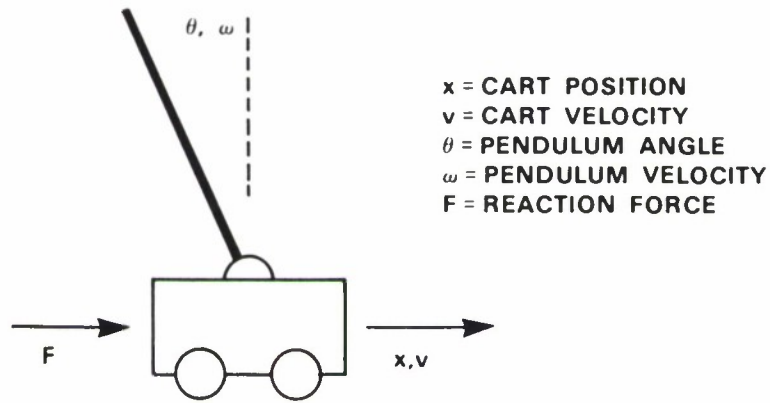


Figure A-1. Inverted Pendulum.

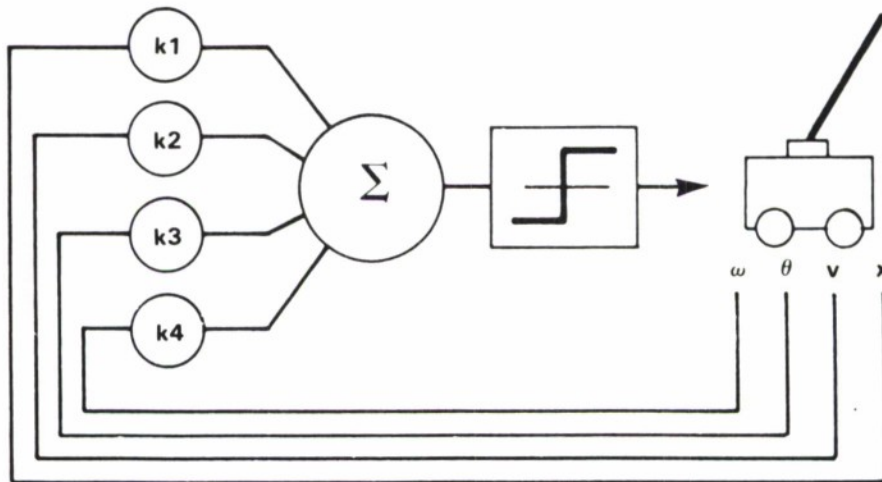


Figure A-2. Realization of a Control System for the Inverted Pendulum based on Equation A.1.

can examine pattern recognition and vision problems as well. Finally, the complexity of the inverted pendulum is significant enough to make the problem interesting while still being simple enough to make it computationally tractable.

A.3 PREVIOUS WORK

The use of an adaptive network to stabilize a mechanical inverted pendulum was first studied by Widrow and Smith [2,4,3]. They demonstrated that a network of one computing element, an Adaline (adaptive linear element), was capable of balancing an inverted pendulum if the Adaline input consisted of the four state variables, each encoded with an N -bit linearly-independent code. The force produced by the Adaline approximated that called for by Equation A.1. The network was trained using the LMS, or Widrow-Hoff, algorithm with the output of an optimal controller of the form in Equation A.1 as the teacher. The first objective, to train an adaptive network to balance an inverted pendulum, was first realized by this work.

Barto [1] has also studied the problem of balancing an inverted pendulum using an adaptive network. Barto used an input code and computing unit similar to that of Widrow and Smith, but for training he used a system based on reward/punishment learning.

A.4 CURRENT WORK

The current research began by replacing the mechanical cart and pendulum of Widrow and Smith with a Macintosh computer simulation and display. The adaptive circuits were replaced by software implementation. This work has led to the realization of the second objective, to train an adaptive network to balance an inverted pendulum with the input consisting of a quantized visual image of the pendulum and cart. Examples of the quantized images are shown in Figure A-3. The image is a 5-by-11 binary pixel representation of the cart and pendulum; 65 different images are possible with five different cart positions and 13 different pendulum angles. Pictures larger than the 5-by-11 image were not used because they provided no additional information and added computational overhead. Smaller pictures did not provide the resolution needed.

The input to the network actually consisted of two 5-by-11 images, one representing the present visual image and the other representing the visual image at a slightly earlier time. Both images were necessary because the network had to derive the cart and pendulum velocities as well as their positions.¹ With these two images, there are 4,225 different possible input combinations (65×65) and a total of 110 input bits ($5 \times 11 \times 2$).

Simulations were performed on a network consisting of one Adaline unit. The learning curve is shown in Figure A-4 and a “generalization curve” is shown in Figure A-5. Because the input set is not linearly-separable, the Adaline was unable to completely learn all the correct responses. On average, about 3.5% of its responses were incorrect. In spite of this, the Adaline performed remarkably well. When tested, it was able to balance the inverted pendulum indefinitely, without crashing.

¹ The system could be designed to work with only one image by internally re-using the current image with the next image with the aid of delay circuits. The end result would be the same.

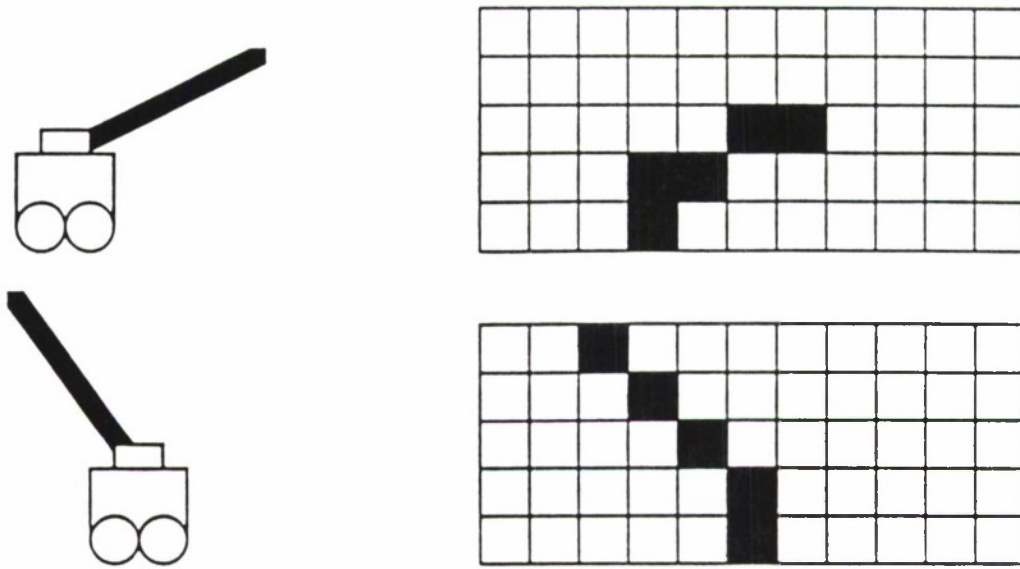


Figure A-3. Two Examples of Macintosh-generated Pendulum and Cart Images. The pictures on the right are 5×11 quantized images of the pendulum and cart figures on the left.

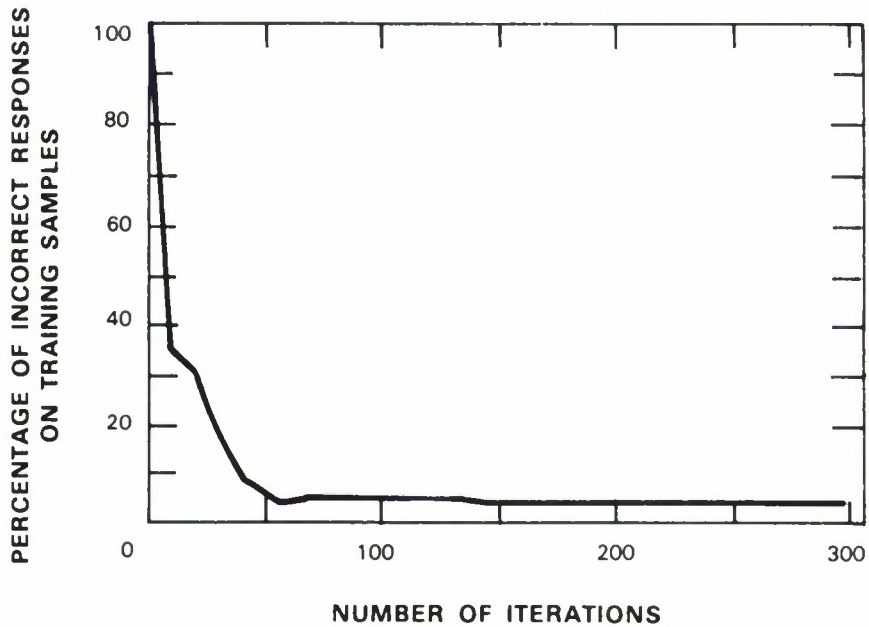
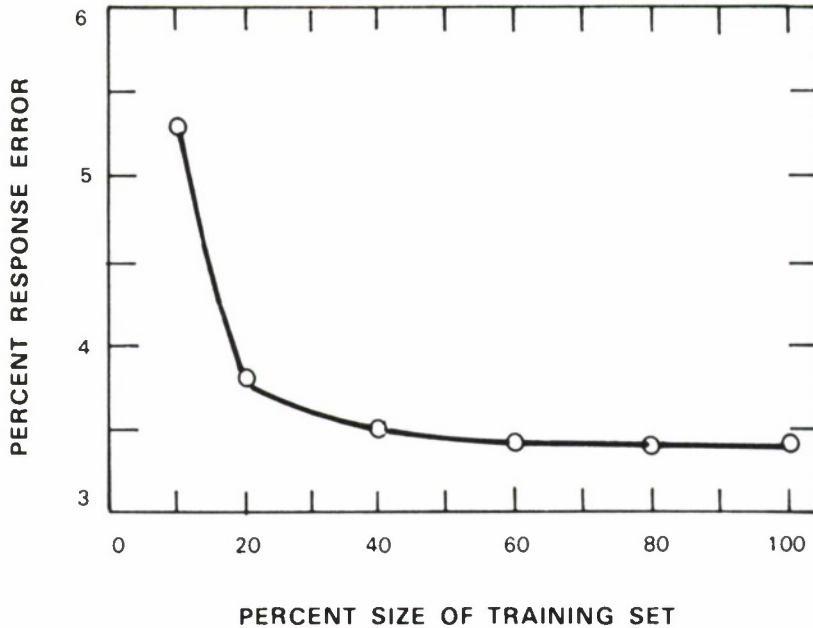


Figure A-4. Learning Curve for a One-unit Network Using Two 5×11 Images as the Input.



101882-57

Figure A-5. Generalization Curve for a One-unit Network Using Two 5×11 Images as the Input.

A.5 SUMMARY AND FUTURE

Although control of the inverted pendulum was able to be achieved with a single Adaline unit with visual inputs, more complicated tasks will require larger adaptive networks with appropriate sensory inputs.

Expert systems are a fertile ground for adaptive network research. The current method of having to extract knowledge from an expert and then explicitly program this into a computer can be replaced by having an adaptive network learn by simply observing an expert. Many of the problems that need expert systems can be reduced to fundamental problems of pattern classification and association. Some of these problems will be ideally suited for adaptive networks.

REFERENCES

1. A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike Adaptive Elements that Can Solve Difficult Learning Control Problems," *IEEE Transactions on Systems, Man, and Cybernetics*, **SMC-13**, Nr. 5, 1983.
2. F. W. Smith, "Contractor Control by Adaptive Pattern-recognition Techniques," *Technical Report 6762-1*, Stanford Electronics Lab, Stanford University, April 1964.
3. B. Widrow, "The Original Adaptive Neural Net Broom-balancer," in *International Symposium on Circuits and Systems, IEEE*, pp. 351-357, May 1987.
4. B. Widrow and F. W. Smith, "Pattern Recognizing Control Systems," in *Computer Information Sciences (COINS) Symposium*, 1963.

APPENDIX B

GTE PROCESS MONITOR

R. S. Sutton
GTE Laboratories
Waltham, MA 02254

This application involves using neural network learning techniques to predict from 100-200 sensor measurements the yield and other performance measures of a fluorescent-bulb manufacturing plant (see Figure B-1). By permitting a better identification of the variables that influence the performance of the plant, substantial improvements in cost and quality can be made. The function performed by the neural network is essentially the same as that performed by conventional statistical analyses, but with significant computational advantages.

For example, a one-layer Adaline network computes essentially the same quantities as a linear regression; both compute numbers relating independent variables (the sensor measurements) to dependent variables (factory yield and other performance measures). In the limit, both techniques in fact produce exactly the same numbers. The difference is that the neural network processes the data incrementally, whereas linear regression is a batch process. The factory generates a new set of independent and dependent sensor measurements every five minutes. The neural network processes each five-minutes'-worth of data as it is generated, whereas conventional regression techniques require all the data to be collected for weeks, months, or years, and then processed all at once as a batch.

The network's incremental approach has the obvious advantages: its statistics are always up to date and it requires less memory. In addition, the processing required is only $O(n)$, where n is the number of sensors – whereas the processing required by linear regression is $O(n^3)$. For hundreds of sensors, this is a tremendous difference; the network could be implemented on a much smaller computer, or it could be used with many more sensors, with more frequently sampled sensors, or with more nonlinear terms considered. In general, the network solution appears better at efficiently tracking the changing relationships in a large data set of this nature. By applying both approaches in the same environment to the same process, we should be able to make a valid and accurate cost performance comparison.

In addition to the single-layer Adaline network, we will also use multi-layer backpropagation networks to capture nonlinear relationships, and temporal-difference learning to capture temporal relationships. Temporal-difference learning methods are also expected to yield faster learning and additional computational savings [1]. Future plans are to move from process monitoring to process control using reinforcement learning techniques. This application is currently under development and will be installed in 1988. Algorithm development is currently proceeding using a simulation of the manufacturing line.

The special features that make this application significant are:

1. It will permit a direct comparison of neural network and conventional statistical methods;
2. If successful, the techniques will permit substantial cost savings in the full range of manufacturing applications;

3. A first level of success relies only on existing, well-understood neural network technology, that is, the ADALINE;
4. A neural network learning algorithm is used to track a moving process, extrapolating from a large database; and
5. The application is precursor to using neural networks for process control.

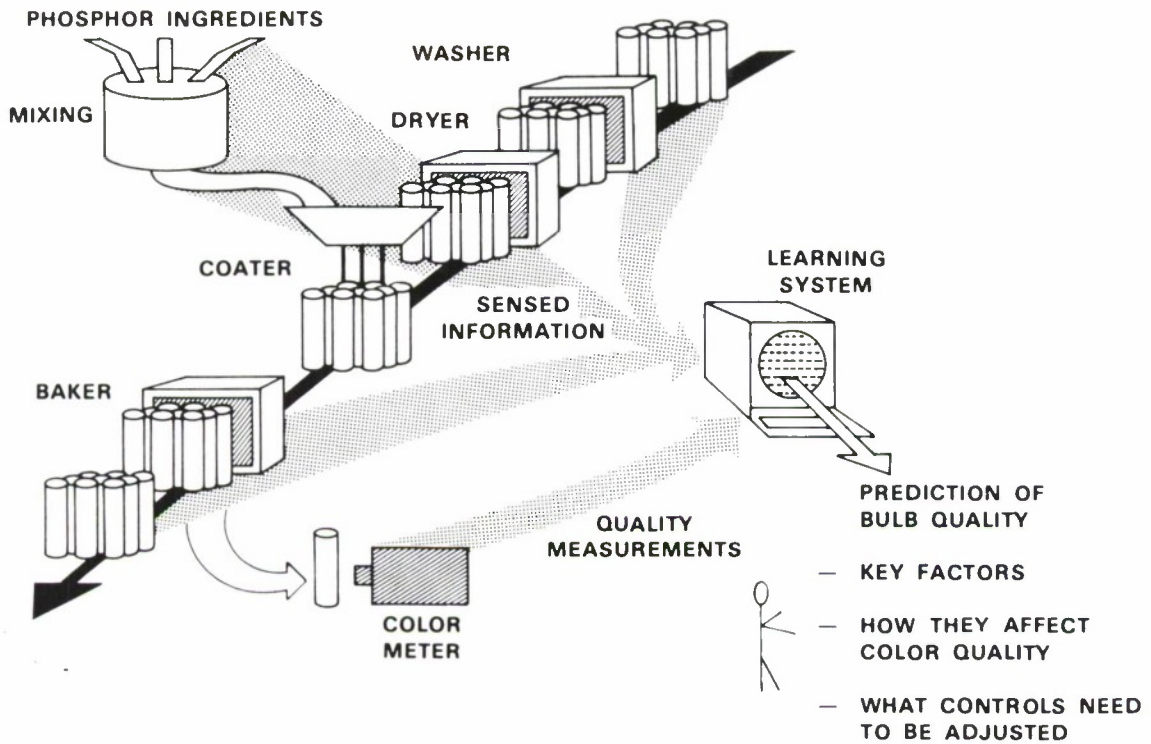


Figure B-1. Connectionist Learning for a Fluorescent Bulb Manufacturing Process Line.

REFERENCES

1. Sutton, R. S., "Learning to Predict by the Methods of Temporal Differences," to appear in *Machine Learning*, 1988; also Technical Report TR87-509.1, GTE Laboratories, Waltham, MA, 1987.

APPENDIX C

SPEAKER-DEPENDENT ISOLATED WORD SPEECH RECOGNITION (E.G., INTEL'S iSBC 576)

M. E. Hoff
Los Altos Hills, CA 94022

Speech is one of the most natural forms of communication between humans. It would therefore be highly desirable for humans to be able to use speech to communicate with machines such as computers. The design of a full natural language speech recognition system appears to be beyond today's technology because of the enormous complexity of human speech. One approach to making speech recognition feasible is to constrain the form of speech to be accepted. In this way, it becomes possible to achieve reasonable performance with a relatively simple system.

The most common constraints applied are:

- Limiting the use of the system to a single speaker at a time. Each such user must train the system by providing examples of his speech. Generally, there is much less variability in a single speaker's utterances than there is over a population of speakers.
- Limiting utterances to isolated words or phrases rather than allowing continuous speech. In this way, words can be isolated and time-normalized easily. While continuous speech recognition is possible using dynamic time-warping algorithms, these algorithms require much more computation than those used with isolated words.
- Limiting the vocabulary. While most humans have vocabularies of many tens of thousands of words, typical speech recognition systems limit the vocabulary to a few hundred words to save storage and reduce processing time.

With the restrictions listed above, speech input utterances can be sampled and reduced to patterns containing up to a few hundred parameters. These patterns are stored during a training phase, then used as templates to be compared with new (unknown) patterns when the system is used in recognition mode. The word recognition process consists of locating the stored template which most closely resembles the pattern generated from the unknown utterance.

By setting a threshold for acceptance, utterances not in the training set can be rejected as unknown. With this technique, if the best match found is worse than the threshold value, the system treats it as "no match." This template matching algorithm is equivalent to a type of neural network known as a matched filter. Each of the stored templates corresponds to one neuron. These neurons may be considered to have dynamically variable thresholds such that no more than one neuron can "fire" for any given input pattern.

Figure C-1 shows a diagram of such a matched filter speech recognition system. Speech input from the microphone is converted to a pattern represented by the set of input values $x_i, i = 1 \dots n$. Each of the templates is stored as a set of weights associated with one neuron which generates an output

$S_j, j = 1 \dots m$. The peak picker selects the largest output, and, if it is acceptable (meets threshold requirements), provides the code j as an output. If the stored template patterns are binary, the neuron

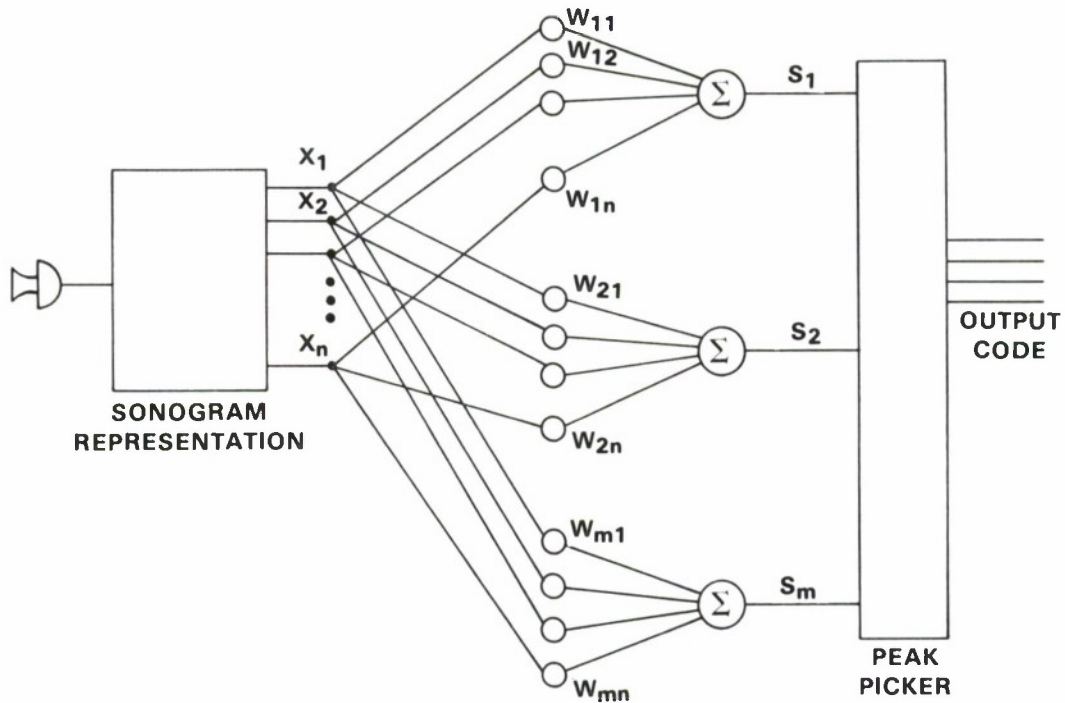


Figure C-1. Matched Filter Word Recognizer.

weights are binary values. For this case, the summation process is equivalent to counting the number of positions in which the input (unknown) pattern matches the stored template weight.

When non-binary templates are used, computing the dot product may require some time, as multiplications must be performed. An alternate procedure consists of summing the absolute magnitude of the difference between input value and corresponding weight value. The best match then corresponds to the lowest sum.

As an example of such a system, consider the Intel iSBC 576 speech transaction module. The module is implemented as a single printed circuit board, 6.75×12.0 inches in size, which contains an 8086 microprocessor, EPROM program memory, 128k bytes of data memory, a speech front end processor, and communication circuits.

The front end processor is equivalent to a spectrum analyzer which provides measurements of energy in 16 frequency bands. This analysis is performed by two 2920 "analog" microprocessors. The outputs of this front end are read by the 8086, and placed in a circular buffer. The 8086 processes the contents of the circular buffer to establish word onset, word offset, and to perform time normalization. The information is then reduced to a binary pattern – i.e., one pattern for each utterance. These patterns are stored as templates during the training mode, or are used as the (unknown) input to the matched filter neural network when in recognition mode.

The matched filter is realized as a program routine by the 8086 microprocessor. This program serially scans the stored templates and computes a match score for each neuron. The best score is saved, and compared against the acceptance threshold. If the score is above this threshold, the system treats the input as matched to the corresponding template. If not, the system responds that the input is unknown.

In addition to performing the input data formatting and the neural network simulation, the 8086 executes several other functions. One of these functions is implementing a "speech transaction processor" which is a form of state sequence machine.

This state sequence machine allows the system to carry on a form of conversation with the user. Different nodes of the state sequence may have different prompts to the user, may limit the acceptable utterances to different subsets of the vocabulary, and may have different outcomes for a given utterance. This technique allows the system to request clarifying information if it is needed but was not provided by the user, and allows one user to transfer the system to another by downloading the new user's templates.

This system was announced in 1982 and has been in use since that time. A typical application is data collection for automobile manufacturing. The inspector wears a headset microphone with a two-way radio link to the speech recognition system. Prompts are converted to speech by a speech synthesis module. The inspector's hands are free, allowing him to maneuver himself around the vehicle. Should he miss some portion of the inspection, the system may prompt him. As soon as the inspection is complete, the speech transaction system communicates the information to the factory host computer, which routes the vehicle to appropriate repair locations as necessary. Prior systems required the inspectors to carry clip-boards, and required manual entry of the inspection data before the vehicle could be routed for any needed repair.

Typical specifications for the speech recognition system are a 200-word vocabulary, of which 50 words may be active (allowed) at any point in the "conversation." The response time is then less than one-half of a second, and the accuracy is better than 99% correct for the speaker who trained the system.

Possible modifications or upgrades to the system include self-adaptation to allow tracking variations of the speaker due to colds, fatigue, etc. Inclusion of this modification would require the coefficients of the patterns to be variables which can accept small changes. When each unknown pattern is successfully recognized, the corresponding template would be changed slightly to make it more like the input pattern.

A second modification of the system would be to allow continuous speech input. This modification would typically require significantly more computation, because the stored templates must be compared to the input many more times. Some continuous speech recognition systems have been implemented with 8086 microprocessors, although it is more likely that much better specifications would result from using one of the newer, very-high-performance general-purpose microprocessors, or even a special-purpose processor.

APPENDIX D

LEARNED CLASSIFICATION OF SONAR TARGETS USING A NEURAL NETWORK

R. Paul Gorman
Allied-Signal Aerospace Technology Center
Columbia, MD 21045
and
Terrence J. Sejnowski
Johns Hopkins University
Baltimore, MD 21218

D.1 INTRODUCTION

Neural network models were trained to classify sonar returns from an undersea mine and rock. The network achieved a performance level of as high as 100% correct classification on a training set of sonar returns, and as high as 90.4% correct classification on a test set. The network classifier performed better than a nearest neighbor classifier, and was also better than trained human listeners on the same set of signals. An analysis of the hidden units in trained neural networks revealed that spectral shift-invariant signal features were discovered by the network and used to achieve accurate classification. Finally, the features discovered by the neural network were shown to be similar to perceptual cues utilized by trained human listeners.

D.2 PROBLEM AND APPROACH

Neural networks were applied to a sonar target classification problem of particular interest to the Navy. The remote detection of undersea mines in shallow waters using active sonar is a crucial capability required to maintain the security of important harbors and coastline areas. It is often very difficult to distinguish active sonar returns from mines and returns from clutter on the sea floor. There is currently no reliable signal classification scheme for automatically interpreting such sonar returns. Instead, highly trained sonar operators must be relied upon to identify the presence of a mine. The present study was conducted to explore the use of neural networks as a means of automating mine-hunting operations.

The database used for the network experiments were active sonar returns collected from a mine and a "mine-shaped" rock positioned on a sandy ocean floor. Both targets were approximately five feet in length and cylindrical in shape. The returns were obtained by a diver equipped with a sonar gun at a range of 10 meters and at various aspect angles. The impinging pulse was a wide-band FM chirp. About 100 returns were obtained from each target. The returns were preprocessed to obtain a power spectral envelope which provided an appropriate signal representation for input to the networks. The spectral envelope consisted of 60 samples whose values ranged between 0.0 and 1.0.

The networks used for the sonar classification experiments were composed of three layers of simple processing units, an input layer consisting of 60 units, a hidden layer consisting of from 0 to 24 units, and an output layer consisting (Figure D-1). Each unit performed a weight sum of all its inputs to determine its activation level and then performed a nonlinear transformation on the activation to obtain a continuous state value between 0.0 and 1.0. Each unit received input from all the units in the layer below it through a weighted connection. The weights on connections could be positive or negative real values and were incrementally adjusted to improve network performance.

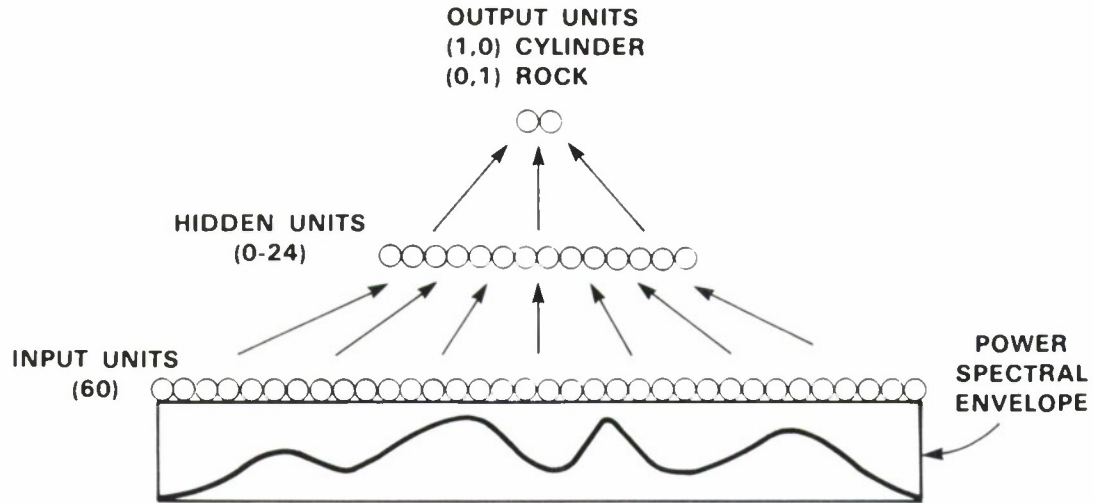


Figure D-1. Schematic Diagram of the Network. The bottom layer has 60 processing units with their states "clamped" to the amplitude of the preprocessed of the sonar signal, shown in analog form below the units. The two output units at the top represent the two sonar targets to be identified. The layer of hidden units between the input and output layers allows the network to extract high-order signal features. The connections between the layers of units are represented by arrows.

The classification task required of the network was to transform the signal representation at the input layer to an output pattern that would code for the appropriate signal class. The states of 60 input units were "clamped" to the 60 spectral envelope values, and the states of the two output units determined the class of the signal: (1 0) for a return from the mine, and (0 1) for a return from the rock. A single learning cycle consisted of:

1. Clamping a sonar return at the input to the network,
2. Computing the resulting state of each unit,
3. Comparing the states of the output units with desired states determined by the class of the input return,
4. Computing the total error as the difference between the actual and desired output states, and

5. Adjusting the weights on each network connection to decrease the error.

To achieve the final step, the backpropagation learning algorithm was used [1].

D.3 NETWORK EXPERIMENTS AND RESULTS

For each network experiment, a given network was presented with a sequence of training returns during which weight values were changed to improve performance. The set of training returns was presented a total of 300 times. The trained network was then presented with a set of test returns excluded from the training set to determine its ability to generalize. Each experiment with a given network was repeated 10 times with different initial weight values to average over variations in performance due to initial conditions.

The results of one series of experiments which demonstrate the network's ability to learn to classify the sonar returns in the training set and to classify new returns in the test set is shown in Table D-1. For this series, the training and testing sets consisted of about 50 returns from each target chosen so that each set contained returns from all available aspect angles. Performance is given in terms of percent correct classification averaged over 10 trials. Experiments were conducted on networks with 0, 2, 3, 6, 12, and 24 hidden units. A network with no hidden units has only two layers of processing units – an input and an output layer. The network with 12 hidden units learned to classify returns in the training set almost perfectly (99.8%), and could use the learned classification strategy to correctly classify 90% of the returns in the test set.

NUMBER OF HIDDEN UNITS	AVERAGE PERFORMANCE ON TRAINING SETS (%)	STANDARD DEVIATION ON TRAINING SETS (%)	AVERAGE PERFORMANCE ON TESTING SETS (%)	STANDARD DEVIATION ON TESTING SETS (%)
0	79.3	3.4	73.1	4.8
2	96.2	2.2	85.7	6.3
3	98.1	1.5	87.6	3.0
6	99.4	0.9	89.3	2.4
12	99.8	0.6	90.4	1.8
24	100.0	0.0	89.2	1.4

Table D-1.

Results for the Aspect-angle Dependent Series

In order to compare the performance of the neural network to a traditional signal classification technique, a nearest-neighbor classifier was designed for the same set of returns preprocessed in the same way. A nearest-neighbor classification rule specifies that the class of a new pattern should be assigned the class of

the closest known pattern. This technique is generally quite powerful but computationally very expensive, since all known patterns must be stored in memory and each new pattern must be compared to all the stored patterns to determine its class. The performance of the nearest-neighbor classifier was 82% on the entire set of 208 returns. The three-layered network classifiers were consistently better.

D.4 NETWORK INTERPRETATION

In addition to demonstrating the ability of neural networks to classify complex signals, we were interested in understanding the classification strategy discovered by the networks. One way to accomplish this is to interpret the patterns of weights on connections between processing units in trained networks that are matched to structure in the signal. We chose a trained network with only three hidden units, but with good performance, to simplify the analysis. We analyzed each hidden unit independently and then determined the strategy of the network as a whole. Since different signal patterns interact with the weight patterns in different ways, we developed a technique which characterized the signals that produced the highest activation for each hidden unit. This is analogous to the concept of best feature for sensory neurons in the nervous system.

Through the use of this technique, we identified three signal features that the trained network learned to extract from the signal in order to classify the sonar returns accurately:

1. The rate of signal onset,
2. The rate of signal decay, and
3. The bandwidth of the signal.

One notable characteristic of this strategy is that all three features were extracted independent of the return's central frequency. This demonstrated an important cooperation among hidden layer units in learning to solve the classification problem.

One potential limitation of neural networks is that they are sensitive to the input pattern's registration. That is, an input pattern that is shifted with respect to the input units represents an entirely different pattern to the network. Since the input signal representation for the present problem was a power spectrum, shifting a signal pattern with respect to central frequency is the same as shifting the pattern with respect to the input units. The strategy learned by the network to overcome the shift-invariance problem is shown in Figure D-2. Each of the three hidden units specialized for patterns in frequency bands with different central frequencies.

Psychophysical studies were also conducted to determine the signal correlates of the perceptual cues used by trained human listeners. The performance of trained human subjects averaged about 92% on training sets. One subject was tested on the same test set used to test the network classifiers and classified 84% of the test returns correctly. The performance of the three-layered networks were better than the performance of trained human listeners. In addition, the cues used by the trained human listeners correlated highly with the features extracted by the trained networks. This suggests that the classification strategy internalized by the network was in some sense similar to the representation utilized the human classifier.

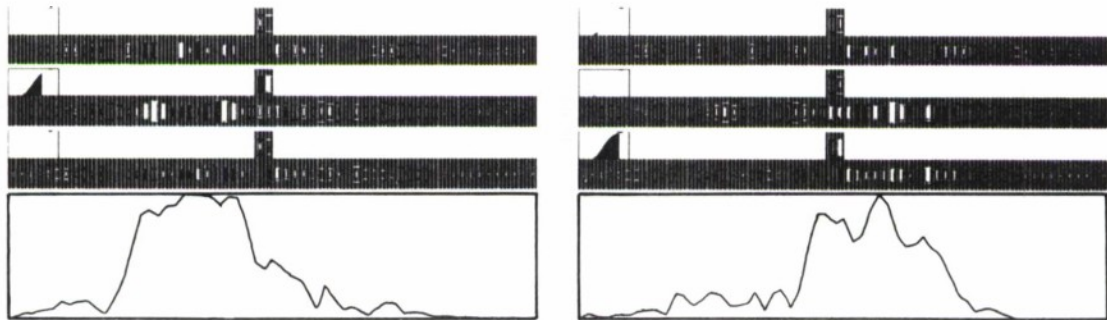
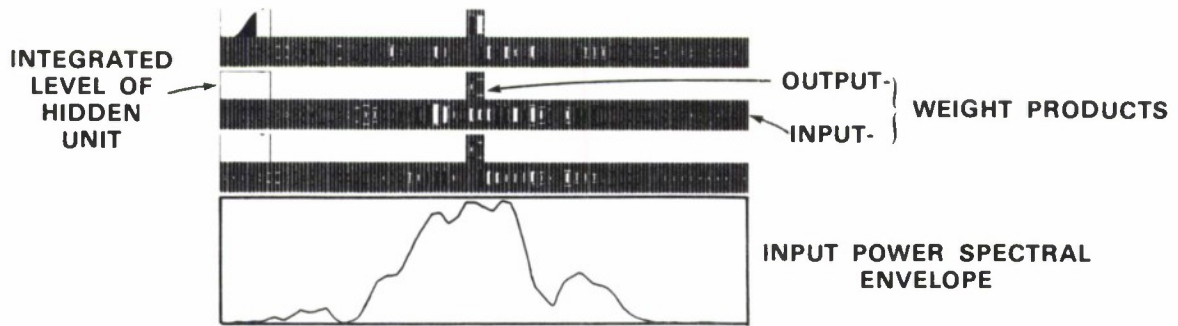


Figure D-2. The response of three hidden units to narrow-band input patterns. Each gray bar represents a hidden unit and the area of the rectangles within are proportional to the weight-state product of each unit. White rectangles are excitatory and black rectangles are inhibitory. The sigmoids at the left of each hidden unit show the output state of the hidden unit as a function of the activation level. The three hidden units shown respond preferentially to narrow-band patterns with different central frequencies.

Finally, the difference in performance between human and network classifiers appeared to be related to the network's ability to memorize exceptional returns that did not conform to the general classification strategy.

REFERENCES

1. Rumelhart, D., Hinton, G., and Williams, R., "Learning Internal Representations by Error Propagation," in: D. Rumelhart and J. McClelland, (Eds.) *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*. pp. 318-362, MIT Press, Cambridge, 1986.

APPENDIX E
A NEURAL NETWORK DECISION LEARNING SYSTEM APPLIED TO RISK
ANALYSIS: MORTGAGE UNDERWRITING AND DELINQUENCY RISK
ASSESSMENT

Edward Collins, Sushmito Ghosh, Christopher Scofield
Nestor Inc.
Providence, RI 02906

Many real-world problems require decisions to be made based on large numbers of inputs that may originate from such diverse sources as insurance applications or the outputs of many sensors. The Nestor Multiple Neural Network Decision System has been designed to handle such problems. We present here, as an example, our application of this system to the problem of mortgage underwriting. This is a specific application, using real-world data, of our risk-analysis decision systems.

Mortgage underwriting, at first glance, would seem to be an unlikely candidate for the application of neural networks. This is a problem in risk assessment that might seem to be best solved with a rule-based system, as the methodology applied by underwriters is thought to be well understood. Further, underwriters must be able to justify the decisions that they make, and it is sometimes said that neural networks are not suitable for applications where this is a requirement. In addition, problems in this domain often involve the use of large numbers of features ranging from personal income to street addresses. A very large volume of data is required to develop a model that would be broad enough to handle the variability of such problems and it is known that the training time of many of the popular algorithms grows rapidly with increases in the scale of problems, making them unsuitable for applications of this type.

However, the difficulty of getting underwriting “experts” to specify the rules they use in making their more subtle decisions, and the high economic value of reducing the number of bad risks presently accepted by underwriters, makes the application of neural network decision systems potentially very valuable.

Three levels of problems had to be addressed by the system developed at Nestor:

- **Mortgage Origination Underwriter:** Mortgage origination filters the general population of potential property owners according to a set of guidelines, some of which are well-defined and some of which are loose. Many potential borrowers can be assessed for risk through examination of a few economic measures, such as the proposed loan-to-value ratio, the proposed housing expense-to-income ratio, and the proposed total obligations to income ratio. In addition, several binary-valued questions readily determine the outcome of an application: these include the existence of a bankruptcy or foreclosure, whether divorce proceedings are in progress, and employment status. However, once these simple measures have been performed, the vast majority of borrowers will require a more careful analysis and weighting of the potential risk factors.

- **Mortgage Insurance Underwriter:** Mortgage insurance applicants are, by nature, a higher risk group than the general population of mortgage applicants. These applicants have already been underwritten by the mortgage originator and assessed as less secure cases, which require insurance. Thus, this second-order underwriting performed by the mortgage insurer is bound to be more difficult, and prone to greater ambiguity.
- **Delinquency Risk Processor:** The portfolio held by a mortgage insurance company contains some insured risks that will become delinquent over time. Although this population is small, it is a high priority to further reduce it since insurance claims are the major operating expense for a mortgage insurer. Delinquency risk assessment is a third-order filtration of the general population, for loan performance. In this case, no expert rules exist at present, since we are dealing with those files already accepted by the underwriter expert. For this reason, this problem is bound to be most difficult.

E.1 THE MORTGAGE ORIGINATION UNDERWRITER

Development of the Mortgage Origination Underwriter began with the study of a pool of mortgage applications obtained from a major mortgage originator. These applications had been classified into two groups: those that were declined, for a variety of reasons, and those that were accepted. The database consisted of 2,032 files, of which 1,179 were files accepted and 853 were declined by human underwriters.

The feature set of the Mortgage Origination Underwriter consisted of approximately 25 fields selected from the full database. These fields included information related to the borrower's "cultural" status: these included the borrower's credit rating, the number of dependents, and the number of years employed. Fields related to borrower financial status included current income, portion of income due to sources other than salary, and amount of obligations other than the principal property. In addition, features were included which related to the mortgage instrument, such as the loan-to-value ratio, the type of mortgage, and the ratio of income-to-mortgage payments. Finally, there was a class of fields related directly to the property: these included the property age, the number of units, and the appraised value of the property.

The Mortgage Origination Underwriter is based on the Multiple Neural Network Learning System (MNNLS) developed at Nestor, Inc. The MNNLS, described in Section E.5 [p. 73], employs an array of coupled subnetworks and a controller. We have chosen an architecture of nine coupled networks in a 3×3 arrangement. Each of the three networks at each of the three levels is focused on a non-exclusive subset of the full feature space. This partitioning of the full space has at least two advantages: each network space is smaller than the full space, making for a more efficient, easier to train system, and, in addition, the cooperative effect of three individual "experts" in a level of the system is used to determine the level of confidence for a system decision. The three levels of the architecture serve as a hierarchical filter. Networks near the "top" of the structure handle obvious discriminations, while the networks near the "bottom," unburdened with the majority of the decisions, focus on fine discriminations.

Figure E-1 depicts the accuracy of the system on the patterns identified as a function of the fraction of files identified unambiguously. This particular kind of plot is possible with the MNLS, since the system can be configured to identify files with varying degrees of boldness. These configurations of the system are user definable at runtime, thus allowing underwriting agency to differentially underwrite a pool of mortgages according to value.

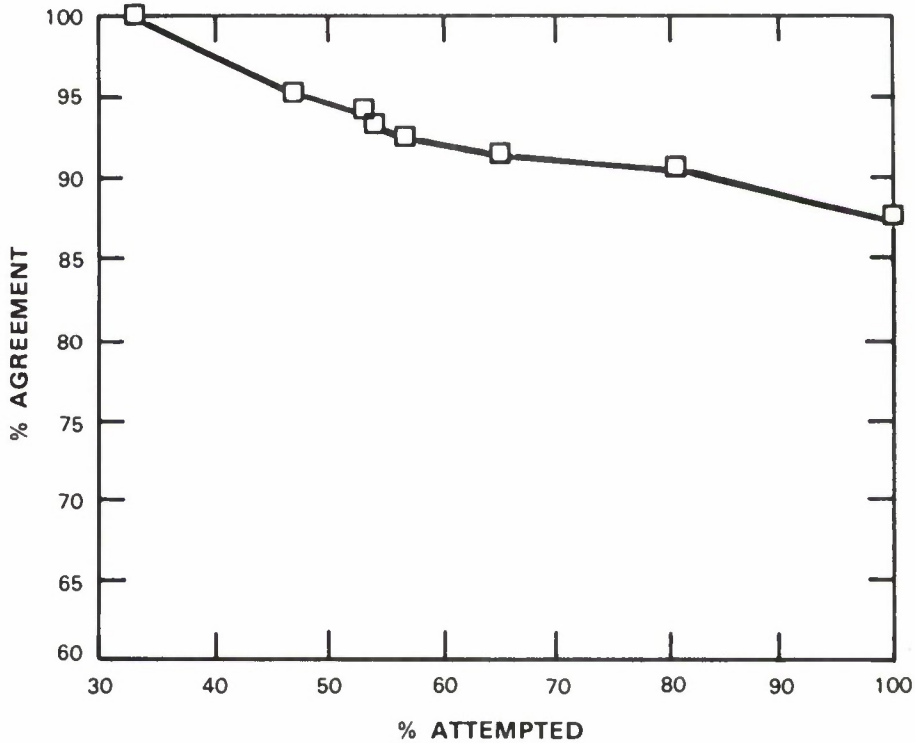


Figure E-1. Mortgage Origination Underwriter: System Agreement with Human Underwriters.

This plot shows a clear trend of increasing confusion in the identifications as the attempt rate increases. This is due to inconsistencies in underwriting judgments in the training database. Analysis has shown that those files identified with high accuracy in the conservative mode are those for which underwriters agree. As the system is asked to attempt the more difficult cases, differences in underwriting practices lead to an ambiguity in the correct classification of files. Those files identified in the liberal mode are in the difficult, "grey" area of underwriter judgment. Overall, the Decision System is found to be much more consistent in its classification of similar applications than are human underwriters.

E.2 THE MORTGAGE INSURANCE UNDERWRITER

The Nestor Research Group has been under contract with one of the nation's largest mortgage insurance agencies to develop an automated mortgage insurance underwriter. The initial goal of mortgage insurance

underwriting is to automate the underwriting process and thus reduce processing costs. The second goal of predicting loan performance is intended to improve on the underwriting practices currently in place.

The structure of data flow in mortgage insurance underwriting creates a filtration of all applications submitted to the insurer. Of all files submitted for underwriting, 80% are accepted, or certified, while 20% are declined as too high of a risk. Of those files which are accepted, approximately 60% are inducted into the portfolio (20% are lost to competing agencies). The delinquency rate of the insured portfolio is quite low: during the first two years, only about 2% to 3% will become delinquent. Over a six year lifetime, as much as 20% of the portfolio will become delinquent. However, this long lifetime makes the acquisition of a training base rather difficult: the type of mortgage instruments and the underwriting practices themselves change substantially over this period. Study of the certification/declination database with the Nestor Learning System technology has produced a prototype mortgage insurance underwriting system, dubbed the Simple Underwriter. The nomenclature is intended to distinguish this system from one which performs the more complex process of improving on human risk assessment. This system is trained on the judgments of the human mortgage insurance underwriters, and learns to mimic their underwriting skill. Further, since the system must be able to provide justification if an application is rejected, the Mortgage Insurance Underwriter must be able to discriminate between types of declinations.

The data for development of the Mortgage Insurance Underwriter consisted of examples of applications submitted to the insurance agency, and the subsequent classifications of the underwriters. The prospective underwriter database on which this study was based consisted of 5,048 applications from all parts of the U.S., collected in the period September 1987 to December 1987. The database consisted of 2,597 files of certifications and 2,451 declinations.

The performance of the Mortgage Insurance Underwriter is illustrated in Figure E-2. As in Figure E-1, we have plotted the accuracy of the system on the files unambiguously classified as a function of the attempt rate of the system.

The Mortgage Insurance Underwriter is able to process over 30% of applications for mortgage insurance with an accuracy of nearly 96%. The system, configured to process all applications, will perform with an accuracy of over 82%. The accuracy of the Mortgage Insurance Underwriter is below the Mortgage Origination Underwriter. This is because the files submitted to the Mortgage Insurance Underwriter have already be underwritten once by humans at the mortgage originator. It is important to note that these results are for a system which performs underwriting on applications from all parts of the country: this system incorporates the geographic knowledge and skills of all underwriters in the agency!

An analysis of the source of disagreement between the neural network underwriter and the human underwriters reveals that much of the disagreement occurs as a result of different human underwriter judgments based on the same file information. In difficult or marginal cases, different underwriters make different judgments based on the same information. As discussed in the next paragraph, it has been possible to assess the quality of the mortgages certified by a separate criterion, and it turns out that the quality of the neural network certified files is consistently higher than those certified by human underwriters. We believe that this is the case because the system makes an informed consensus judgment on each case. This consensus appears to be superior to the typical underwriter's individual judgment. It might be that if the system were trained by the best of human underwriters, it would form an even better consensus judgment.

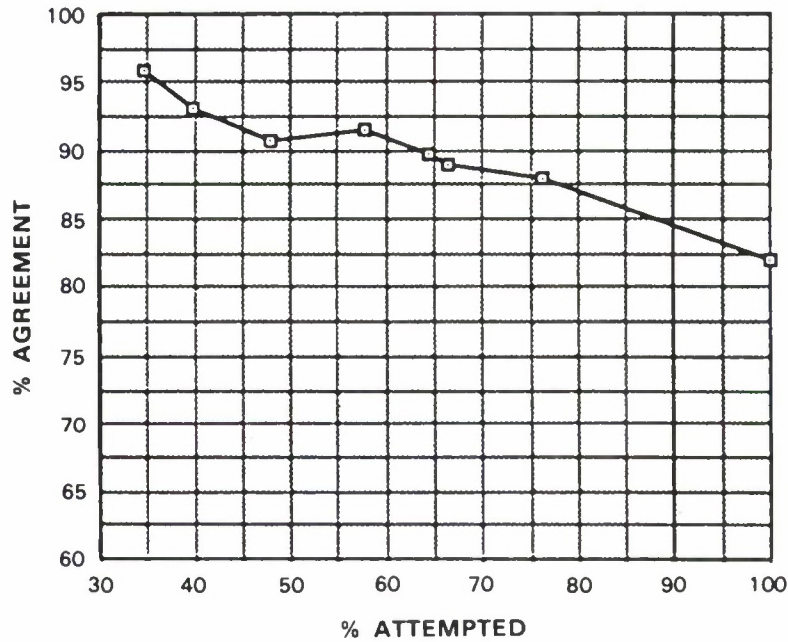


Figure E-2. Mortgage Insurance Underwriter: System Agreement with Human Underwriters.

A comparison of the quality of various sets of files certified and/or declined by the Nestor Mortgage Insurance Underwriter (system) and the human underwriter (underwriter) is listed in Table E-1 for the case of the System being configured for the 100%-throughput mode of operation. Quality, Q , measured by the Nestor Delinquency Risk Processor (DRP [Section E.3]) – a Nestor Learning System trained on the actual performance of insured mortgages to predict delinquencies, is defined as

$$Q = \frac{N_a - N_r}{N_a + N_r}$$

where, for a given set, N_a is the number of files accepted by DRP, and N_r is the number of files rejected by DRP.

In Figure E-3, we plot the quality of various system- and underwriter-certified and declined sets as a function of throughput. We note the generally decreasing quality of certified (and increasing quality of declined) files as the system throughput increases. This indicates that those files for which the system is reluctant to make a decision represent increasing risk. The points on the extreme right of Figure E-3 (100% throughput) are those listed in Table E-1. Note that the quality of system-certified files is greater (and system-declined files is less) than that of the underwriter for all throughput.

The Mortgage Insurance Underwriter is able to offer a measure of confidence in the accuracy of an underwriting judgment. Further, the System provides a list of reasons, ordered according to degree of importance, if a file is declined. This functionality, called the Reason Processor, allows a user to understand the internal mechanisms employed by the networks in the decision-making process. It depends on a

SET	QUALITY
Certified by both System and Underwriter (SC, UC)	0.87
Certified by System (SC)	0.84
Certified by Underwriter (UC)	0.78
Certified by both System and Underwriter (SC, UC)	0.87
Certified by System Declined by Underwriter (SC, UD)	0.50
Declined by Underwriter (UD)	-0.10
Declined by System Certified by Underwriter (SD, UC)	-0.12 0.87
Declined by System (SD)	-0.28
Declined by both System and Underwriter (SD, UD)	-0.36

Table E-1.

Quality Comparison of Data Sets Certified or Declined by the Nestor Mortgage Insurance Underwriter System and by Human Underwriters

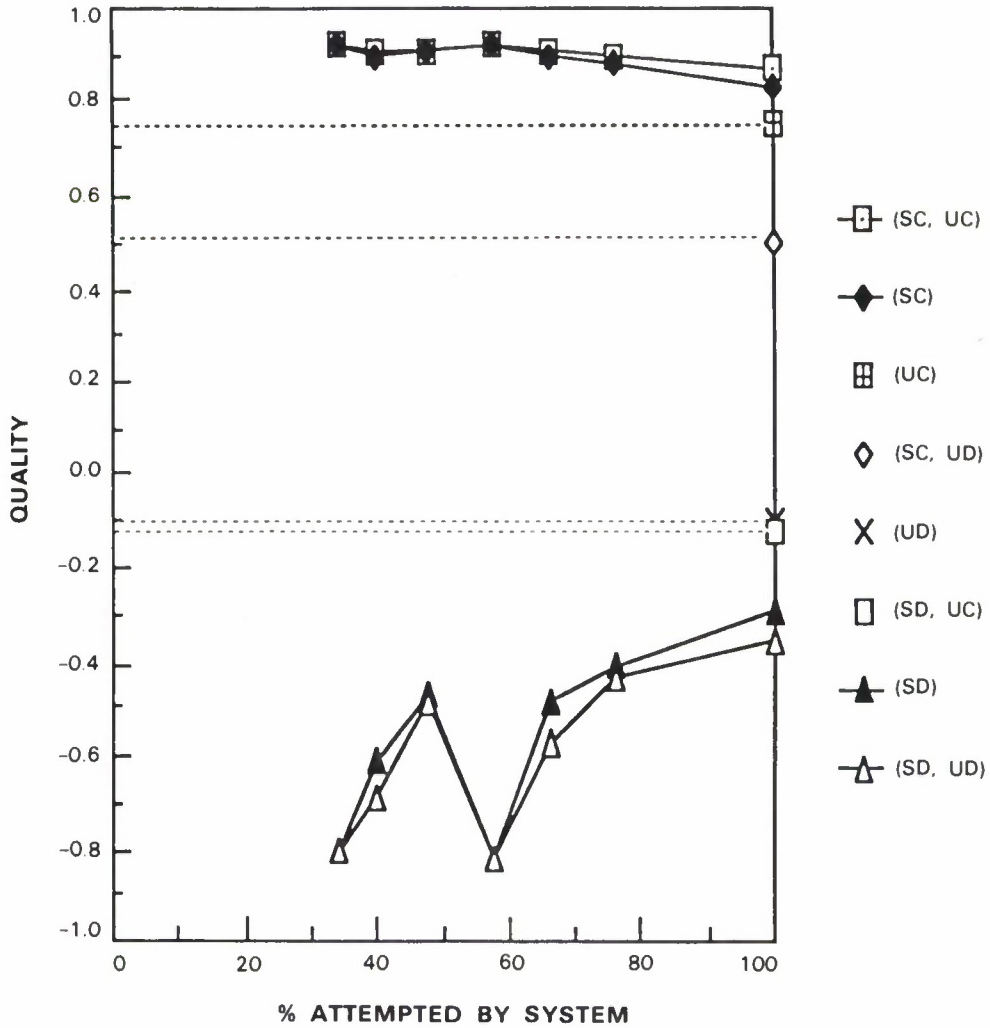


Figure E-3. Mortgage Insurance Underwriter: Comparison of Quality of Various System- and Underwriter-Certified/Declined Sets versus Throughput.

method, developed by the Nestor Research Group, for determining the gradient of the interclass surface within the nonlinear mapping learned by the networks.

E.3 THE DELINQUENCY RISK PROCESSOR

Study of the insured portfolio has produced the Delinquency Risk Processor (DRP). This system is able to determine the risk that a borrower will become delinquent on a mortgage. This system performs a repeated underwriting on files which have already been certified by the insurance underwriters; however, the training database has been reclassified according to delinquency history.

The data which served as the basis for the delinquency database consisted of 111,080 applications taken in the period from July 1984 to December 1986. Files which were not delinquent at the time of delivery of this database (August 1987) were classified as good cases, and files with any degree of delinquency were classified as bad. This classification of the data produced a database of 109,072 good files and 2,008 bad files. From this database, a sampling of good and bad files were selected for training. The final training database consisted of approximately 4,000 files.

Exogenous data provided by Wharton Econometric Forecasting Services was added to the good/bad database. This data contained 14 time-series of macro-economic variables mapped to Metropolitan Statistical Areas in the U.S. In addition, appraisal information, gathered from the appraisal report on the principal property was included. These features covered the appraised value of the principal and three comparable properties, and a variety of ancillary measurements related to appraisal methodology. These features, plus those already described above, constituted the feature set for the DRP.

In attempting to screen good from bad insurance risks in the set of mortgage insurance applications accepted by the underwriter, it must be recognized that for many of the files there is not sufficient data to make this decision. The performance of these insured mortgages will be determined by factors that are at present unpredictable. Thus, the appropriate response of the decision system for these files is undecided. We must base any economic value on that subset of the files for which a DRP decision can be made.

The resulting system is able to correctly identify 10% of files according to loan performance with a greater than 70% accuracy, or nearly one-third of the files with an accuracy of greater than 65%. These modest figures translate into a very real savings in liability expenses: this accuracy has been found to produce a 12% reduction in delinquencies within the insurance agency.

DRP in various configurations makes somewhat different decisions on a given data set. The number of files accepted, rejected, or undecided varies depending on the conservativeness or boldness of the system. To assess the economic consequences of the various configurations, we plot in Figure E-4 the reduction in delinquent loans for a given number of loans insured using the DRP compared with the present delinquent loan ratio against the increase in loans processed to achieve this reduction. One therefore weighs the economic benefits of a reduction in delinquencies against the increased processing costs.

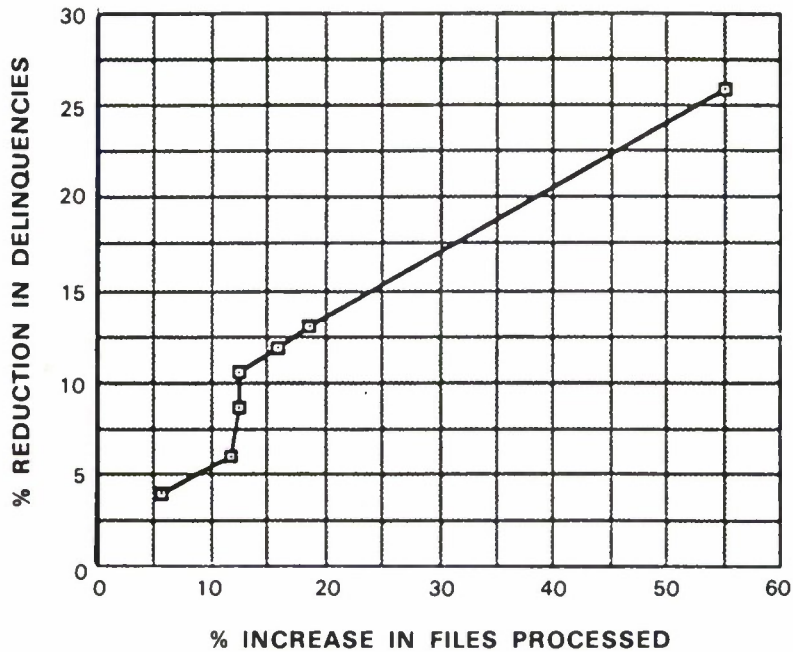


Figure E-4. *Delinquency Risk Processor: Delinquency Reduction.*

E.4 COMPUTATIONAL REQUIREMENTS

The computational requirements of these systems were rather modest: the largest of the three systems described had 6,561 neurons (not including the feature detecting cells). On average, there were over four million binary-valued connections within a system. The MNNLS architecture allows optimization of network traversal times, resulting in an average speed of identification of less than one second on a Apollo DN-3000 workstation. Training required an average of five passes over the full training database. On the Apollo hardware, this usually required on the order of six-to-eight hours. Subsequent test passes in a selected set of system configurations required an additional two hours.

E.5 OVERVIEW OF THE NESTOR LEARNING SYSTEM (NLS)

The NLS is composed of multiple decision-making modules, each a specially-designed artificial neural network. Each network can be defined to process a different feature subspace of inputs. The NLS has a "controller" that synthesizes the outputs of the individual neural networks and directs their training. The entire system is software-based and operational on IBM PC/AT, Sun Microsystems, or Apollo Computer environments.

E.5.1 RCE Network: Neural Network Component of the NLS

Each of the component neural networks in the NLS is an RCE (restricted Coulomb energy) artificial neural network. The architecture of the RCE network specifies three processing layers: an input layer, an internal layer, and an output layer. (See Figure E-5.) Each node in the input layer registers the value of a feature describing an input event. In general, however, there is no requirement to engineer feature values that generate linearly-separable pattern class territories in the feature space for the objects to be identified. The RCE network is able to define a class-separating mapping that can support any required degree of nonlinearity between pattern class territories.

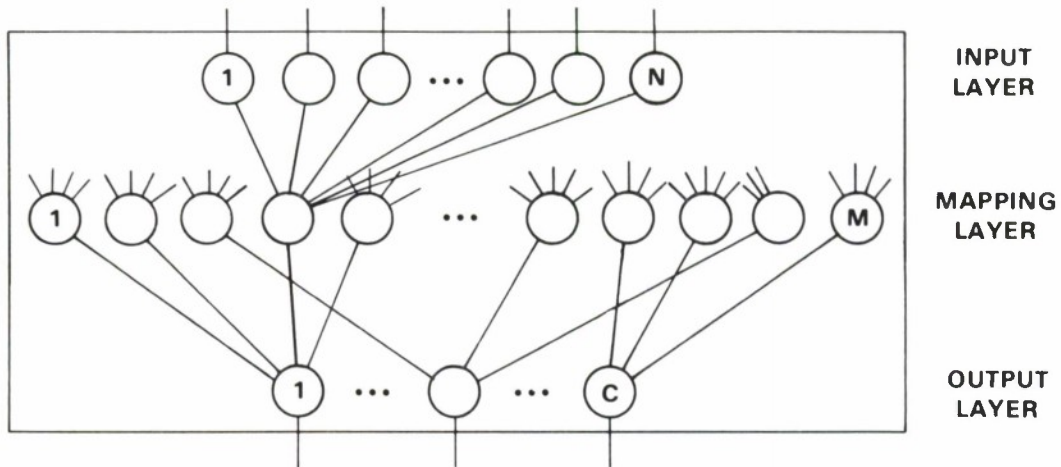


Figure E-5. The RCE Network.

Each cell in the output layer corresponds to a pattern category. The network assigns an input to a category if the output cell for that category “fires” in response to the input. If an input causes only one output cell to become active, the decision of the network is said to be “unambiguous” for that category. If multiple output cells are active, the network response is “ambiguous.” Though confused about the identity of the pattern, the network nonetheless offers a set of likely categorizations. Cells in the middle or internal layer of the network construct the mapping that ensures that the output cell for the correct category fires in response to a given input pattern.

A cell in the internal layer of the RCE network is associated with a set of points in the feature space. The geography of this set of points is defined by the transfer function of the internal cell. For purposes of illustration, consider an RCE network with only two cells on the input layer. (In actual applications, a user defines as many input cells as he needs to represent his input feature vector to the system.) The transfer function can be thought of as defining a disc-shaped region centered at the feature space point w_i (the vector of weights coupling the i^{th} internal cell to the input layer), with a radius λ_i around the w_i . Any point (feature vector) falling within this region will cause this internal cell to become active. Each processing element in the internal layer sends its signal, via a unit strength connection, to one cell in the output layer. The response properties of the output layer are such that, if any of the internal elements to which a given output cell is connected are firing, the output cell will fire.

Two distinct mechanisms support learning in the network. The first is cell commitment. Cells are committed to the internal layer as well as, though less frequently, to the output layer. When cells are committed, they are “wired up” according to the RCE network paradigm. Each cell in the internal layer is connected to the outputs of each of the cells in the input layer. The output of each internal layer cell projects to only one cell in the output layer. That output layer cell is determined by the category of the input event. The second learning mechanism in the RCE network is the modification of the thresholds associated with cells in the internal layer. Thus, the values of the weight vector w_i and threshold λ_i of each internal cell are changed under separate modification procedures.

The commitment of cells to the output and internal layers and the adjustment of internal cell thresholds are controlled by training signals that move from the output layer back into the system. If an output cell (representing a given category of patterns) is off (0) and should be on (1), an error signal of +1 may be generated for that output cell. An error signal of +1 traveling from the k^{th} output layer cell into the system causes a new internal processing element to be committed. Its output is connected to the k^{th} cell in the output layer, and its vector of connections w_i to the input layer assumes the value of the current feature vector of the input layer ($w_{ij} < -f_j, j = 1 \dots N$). The threshold of the cell is set at some positive value λ_0 . In the feature space of the system, this adds a new disc-shaped region that covers some portion of the class territory for this input’s category. (See Figure E-6.)

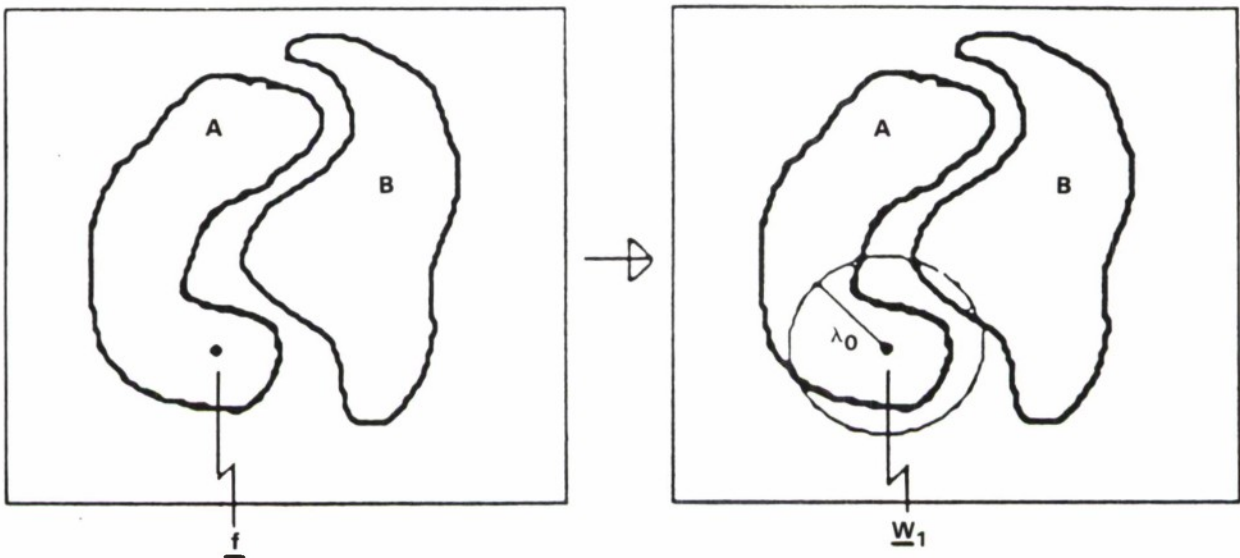


Figure E-6. Feature Space Representation of Cell Commitment in the Internal Layer.

If an output cell is on (1) that should be off (0), an error signal of -1 travels from that cell back into the internal layer. If an error signal of -1 is sent from the k^{th} output unit back into the system, then the system responds by reducing the threshold values (λ_i) of all the active internal units that are connected to the k^{th} output cell. This has the effect of reducing the sizes of their disc-shaped regions so that they no longer cover the input pattern. (See Figure E-7).

Further, this learning strategy is able to map out category regions, even if they are disjoint in the feature space. This functionality is critical to recognizing patterns from real-world data where the pattern signature

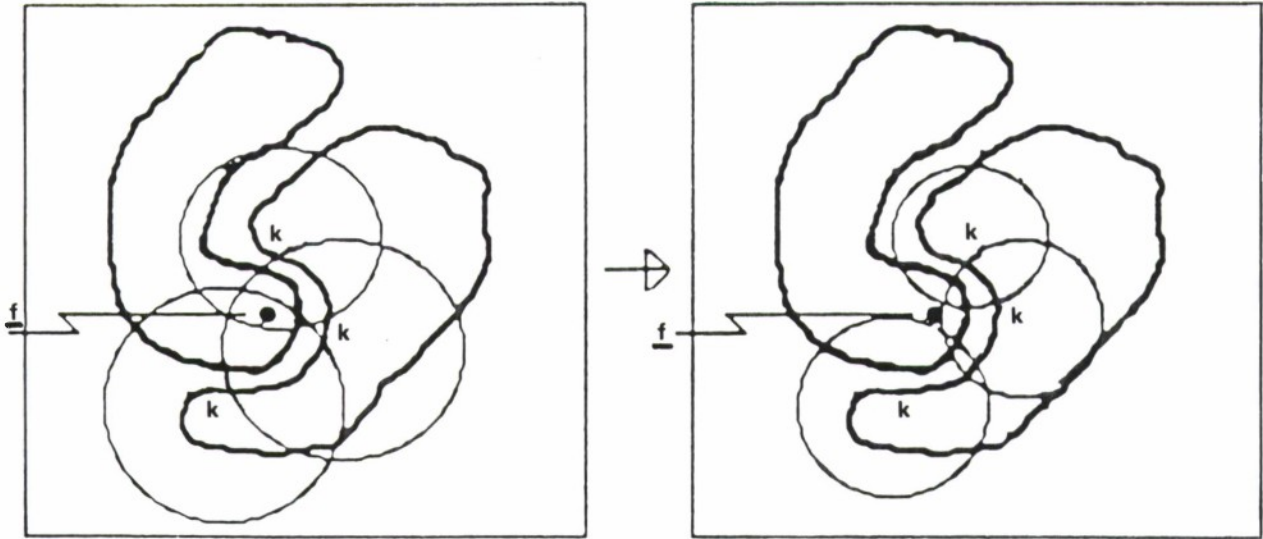


Figure E-7. Threshold Reduction for Active Internal Layer Cells Associated with k^{th} Output Cell. After modification, \underline{f} is no longer covered by an internal cell disc for the k^{th} output cell.

at different times, due to different environmental conditions or modes of operation, may in fact consist of a set of possible pattern signatures defining a number of disjoint regions in the pattern space.

The RCE network can thus be viewed as a network that implements a partially distributed, high storage density memory using a learning process that modifies the number of output and internal processing elements according to a feedforward connection paradigm. It is able to define mappings supporting an arbitrary degree of nonlinear separability. A supervised learning process constructs these mappings rapidly and allows for dynamic category learning through its ability to make local updates to memory at arbitrary points in time.

E.5.2 NLS – A Functional Description

The NLS consists of multiple RCE networks and a controller that both synthesizes a response from the outputs of the networks and directs their training. The controller module consists of a fixed set of rules for determining system output and training.

Different RCE networks are coupled to different feature inputs describing the event. In the case of multi-sensor systems, different networks may be looking at the outputs of different sensors. Some objects may be identifiable on the basis of feature information available to one network and not another. One network may, if properly trained, be able to reliably distinguish α from β , but have insufficient information to unambiguously decide between γ and δ . In other words, a network can be an “expert” at resolving some pattern classes but not others. The controller must function to ensure that a network trains to make whatever discrimination it can. Its strategy for polling networks must take advantage of a network in those

cases where it has clear expertise. Further, it must avoid letting a particular network's uncertainty about a pattern introduce confusion into the system response.

E.5.3 NLS Training to Find the Expert Network

We illustrate how the NLS trains to find and use network expertise with the following hypothetical problem. Imagine two classes of patterns, α and β . Assume that in a feature space representation based on information from sensor 1, the two pattern classes consist of two separable (but not linearly-separable) point regions. In terms of features based on information from sensor 2, however, the two pattern classes are completely indistinguishable (i.e., they share equivalent point sets in the feature space.) (See Figure E-8.) Without *a priori* information as to which sensor can discriminate the two pattern classes, the NLS will learn to base its response to these two input types on the basis of information in sensor 1. If the system response is incorrect, the controller generates error signals for the component RCE networks.

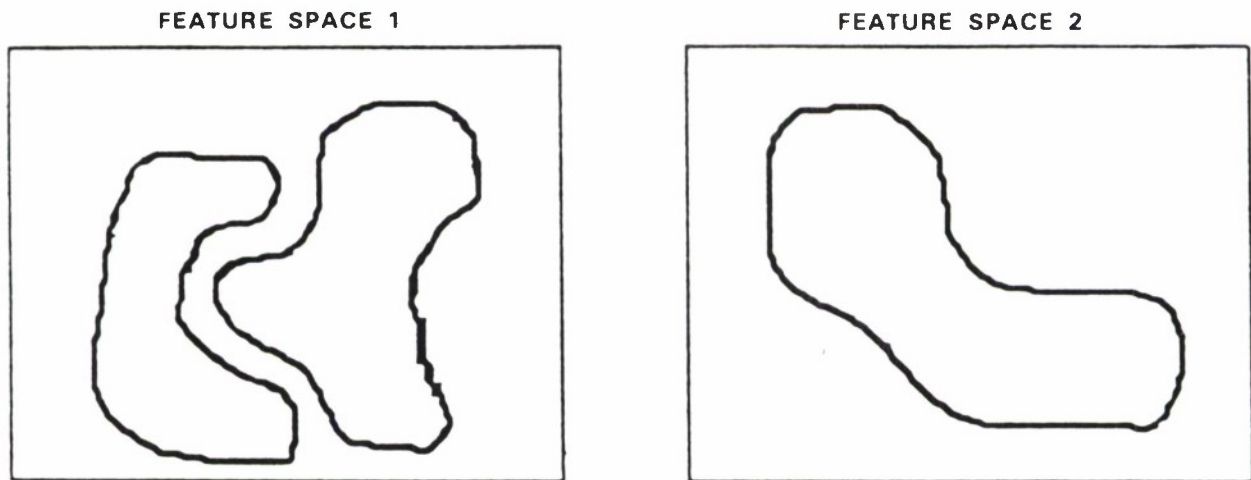


Figure E-8. An Example of Two Pattern Classes with Separable Territories in Feature Space 1, but Nonseparable Territories in Feature Space 2.

E.5.4 NLS Correlating Outputs of Ambiguous Networks

An extension of the RCE training algorithm allows a special class of internal cells to develop which can map out feature space areas in which multiple class distributions overlap. To an input event occurring in such a zone, an RCE network responds with active output layer cells corresponding to the set of classes sharing the overlap zone. The NLS controller can correlate the answers of such confused networks to arrive at an unambiguous response.

As an example, consider the situation depicted in Figure E-9. There, network 1 can separate α from β and γ , but not β and γ from each other. Network 2, however, can separate γ from α and β , but not α and

β from each other. Once the overlap zones are covered on both networks, an example of β will produce a confused response of (β, γ) from network 1 and (α, β) from network 2. The controller can synthesize these answers to arrive at the classification β . Thus, even in the absence of any network uniquely qualified to identify β , the confused responses of multiple RCE networks can be used to produce a correct, unambiguous classification.

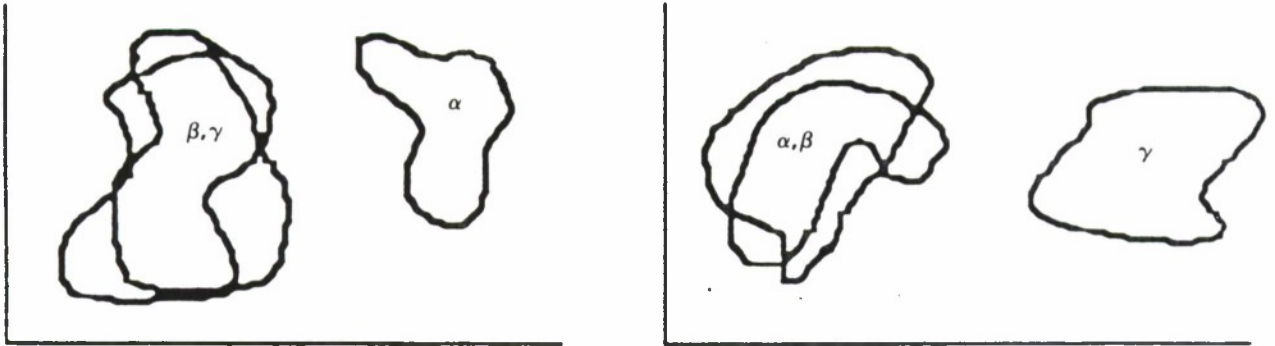
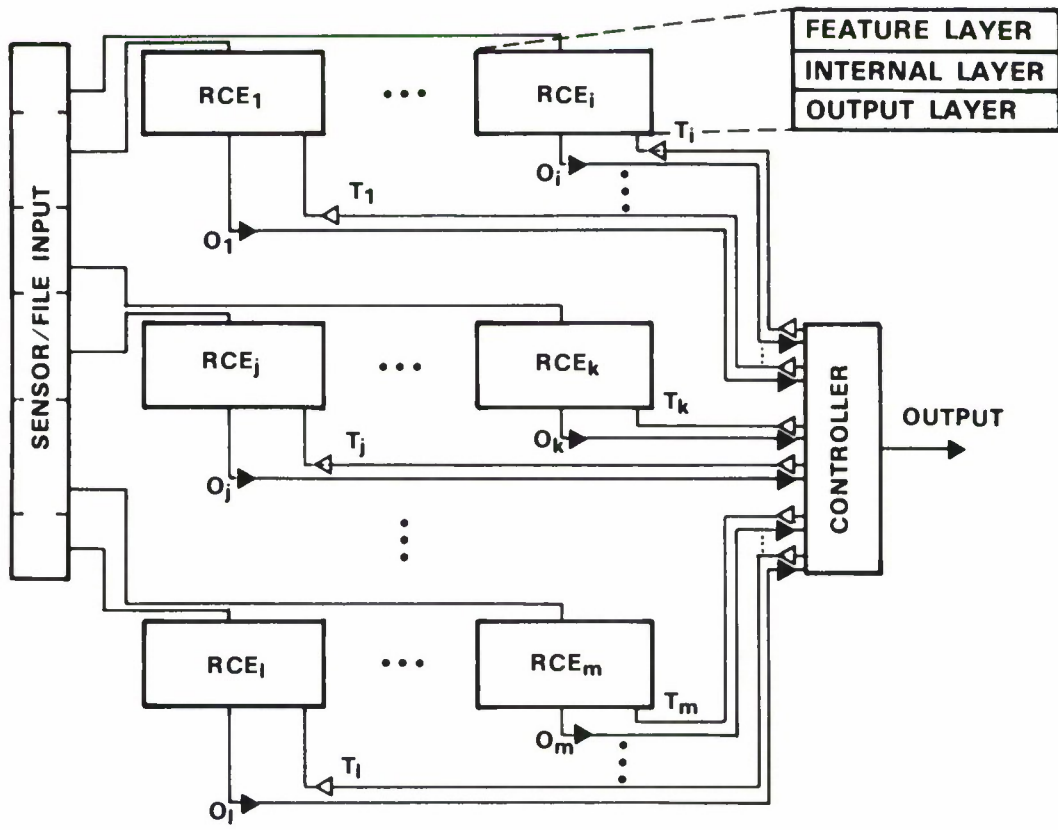


Figure E-9. An Example of Unambiguous Classification of Overlapping, but Mutually Exclusive Response Fields Achieved by Two RCE Networks. Pattern Class β can be uniquely identified.

The obvious modularity (Figure E-10) of decision-making in this system allows new RCE networks to be incorporated at any time in system training and use. Further training will commit internal cells in the new network for those classes of targets that the previously configured system could not identify. For practical applications, an important task is the assessment of the best strategy for placement of new networks in an existing architecture; in particular, the placement of the additional networks at upper versus lower levels in the system must be considered.



O_i OUTPUT SIGNAL FROM RCE_i
 T_i TRAINING SIGNAL TO RCE_i

Figure E-10. The Nestor Learning System.

APPENDIX F

TRAINABLE AND ADAPTABLE NEURAL NETWORKS FOR ROBOTIC CONTROL

Von Ayre Jennings
Martin Marietta
Baltimore, MD 21220

The potential for robotics within the military is great. However, the realization of this potential has been hampered by the much greater than anticipated difficulty in creating computer systems that can provide reliable control in complex environments. Two fundamental problems exist:

- Mathematical models in the form of equations of the mechanics, kinematics, dynamics, and energy states of the controlled devices are very difficult to express and solve. A similar problem exists for the processing and interpreting of sensor data.
- Descriptions of the task to be performed in terms of all of the possible alternate actions needed to cope with all situations are also extremely difficult to generate. These task descriptions are traditionally in the form of conditional branching, state tables, and production rules. The basic problem is that even a moderately complex task places an overwhelming burden on the programmer, who must account for every possible interaction between a large number of possible events.

Neural network models are a radically different approach to computation that offer an alternative to the rules and equations that have plagued traditional control systems. The computational behavior of these models is a collective property that results from having many relatively simple processing elements act on one another in a richly interconnected system. This results in three distinct features of tremendous benefit to robotic control:

- *Control functions can be generated implicitly.* As an inherent property of the stored pattern of weights and connections, inputs to a neural network are transformed implicitly into desired outputs without using equations or rules. This will allow complex nonlinear transformations – such as inverse kinematics, inverse dynamics, coordinate transformations, and trajectory formulations – to be performed non-algorithmically.
- *Control functions can be learned rather than programmed.* By altering weights in response to an internal or external teacher, neural networks can learn complex nonlinear control functions without explicit programming. Thus, low-level servo control functions that incorporate such parameters as backlash, friction, flexibility, and load interactions into a single implicit input-output function can be learned. Furthermore, function generation by learning offers the intriguing possibility of training neural networks for task-level control by observing a human expert perform the task.

- *Multiple control functions can be performed simultaneously.* Computations that are an inherent, collective, and distributed property of a massively parallel system of connections and weights have the potential for enormous power and speed. These qualities will be needed to deal with the multiple and complex sensor and actuator systems of future robotic mechanisms.

While it is clear that neural networks have a very significant potential for robot control, very little is known about how to specify the best network architecture and training procedure for a particular control problem. Furthermore, there have been few studies where neural networks have been applied to actual robot hardware. Such studies are a necessary first step towards the development of neural networks that can deal with real-world problems.

F.1 OBJECTIVE

The overall objective is to use robotic test-beds to investigate the strengths and limitations of basic neural network models with the goal of constructing increasingly powerful network structures that dramatically improve the ability of robots to function in the real world. As a first step towards this goal, the objective of the experiment described in this report is to investigate the ability of a neural network to learn to guide robot movements by observation of human behavior.

F.2 APPROACH

The neural network model used in this experiment is the “Cerebellar Model Articulation Controller” (CMAC), developed by Dr. James Albus at the National Bureau of Standards [ROB-1, p. 164]. CMAC is basically a trainable function generator. It uses a table look-up method to implement complex nonlinear functions implicitly rather than by mathematical solution of equations. Unlike the perceptron, CMAC can implement any smooth and continuous function, including the exclusive OR function and other higher order problems, using simple guaranteed learning procedures such as the Widrow-Hoff algorithm. CMAC has an additional advantage that it can operate in real time on conventional digital computers.

In CMAC, a pattern of input values is mapped into tables of random numbers (Figure F-1). These numbers, in turn, serve as addresses into a look-up table where trainable weights are located. The outputs for the transformed inputs are determined by summing the values of the designated weights. As a result of the structured input mapping, sets of memory locations corresponding to similar input conditions will overlap. This results in the property of generalization between nearby input states. Generalization allows CMAC to predict on the basis of a few representative learning experiences what the appropriate behavioral response should be for similar situations. This is essential in order to cope with the complexities of real-world environments where identical sensory input combinations seldom, if ever, reoccur.

The fact that for N input variables with R distinguishable levels there are R^N possible inputs prevents a simple one-to-one mapping between input and output states using tables of practical size. However, CMAC is based on the assumption that real-world constraints limit the number of possible input-output

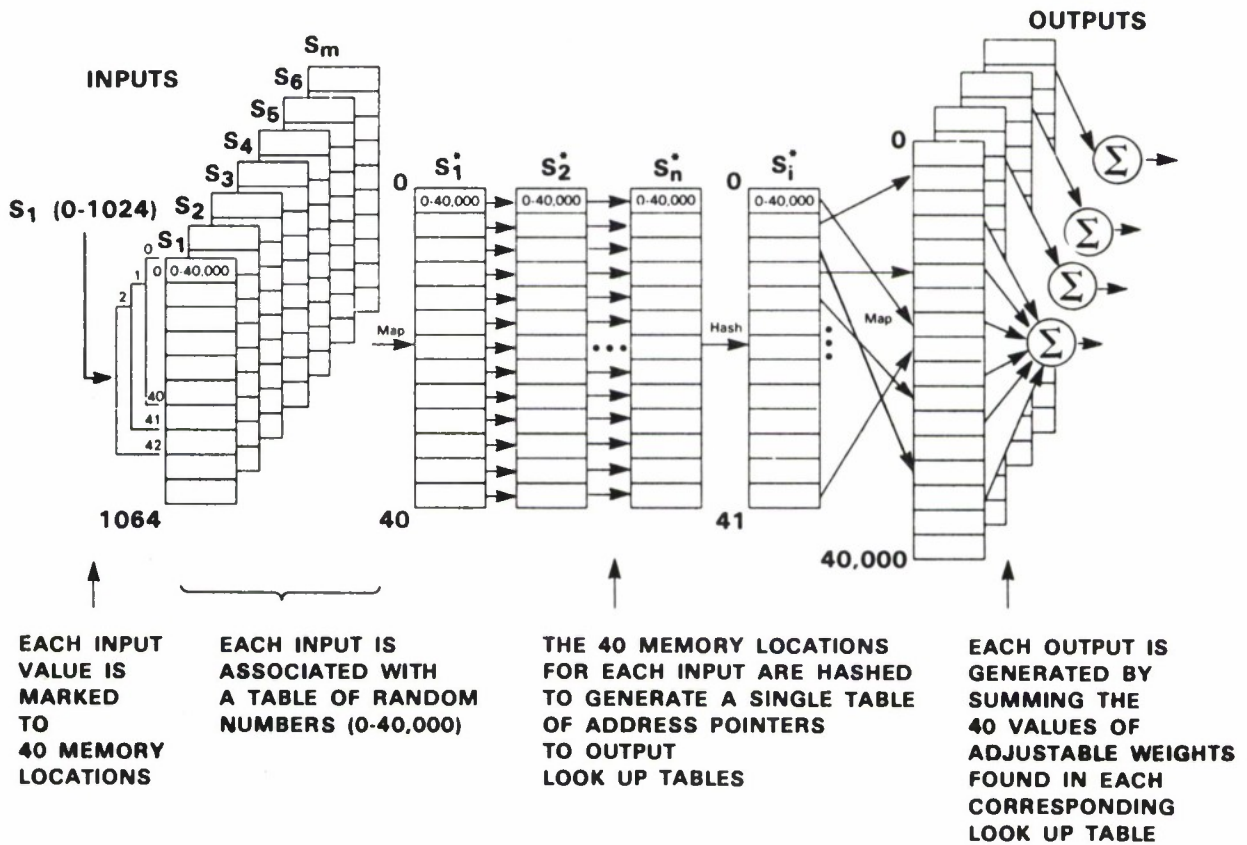


Figure F-1. Cerebellar Model Articulated Controller (CMAC). The CMAC is a table look-up method for training arbitrarily complex input-output functions.

values that are likely to be encountered. This mapping of a very large input space into a smaller, more manageable memory space is achieved, first of all, by limiting the range of random numbers found in the initial look-up table. A second many-to-few mapping technique uses hash coding to take the random numbers generated by all of the inputs and produce one set of addresses to be summed as the output. Together, these two techniques allow very complex nonlinear functions dealing with many input variables to be generated in real time using conventional computers such as the Motorola 68020 used in this investigation.

The test bed for this experiment consists of an industrial robot with a forklift end-effector on which is located a series of infrared proximity sensors (Figure F-2). The task for the neural network is to guide the insertion of the forklift into a pallet that is offset in all six dimensions. This requires the network to learn to generate a Cartesian trajectory command ($x, y, z, \text{roll, pitch, yaw}$) to the conventional robot controller based on the pattern of sensor input values. This task is very difficult for conventional control systems because the relationship between a particular pattern of sensor readings and the appropriate movement direction is extremely complicated and nonlinear.

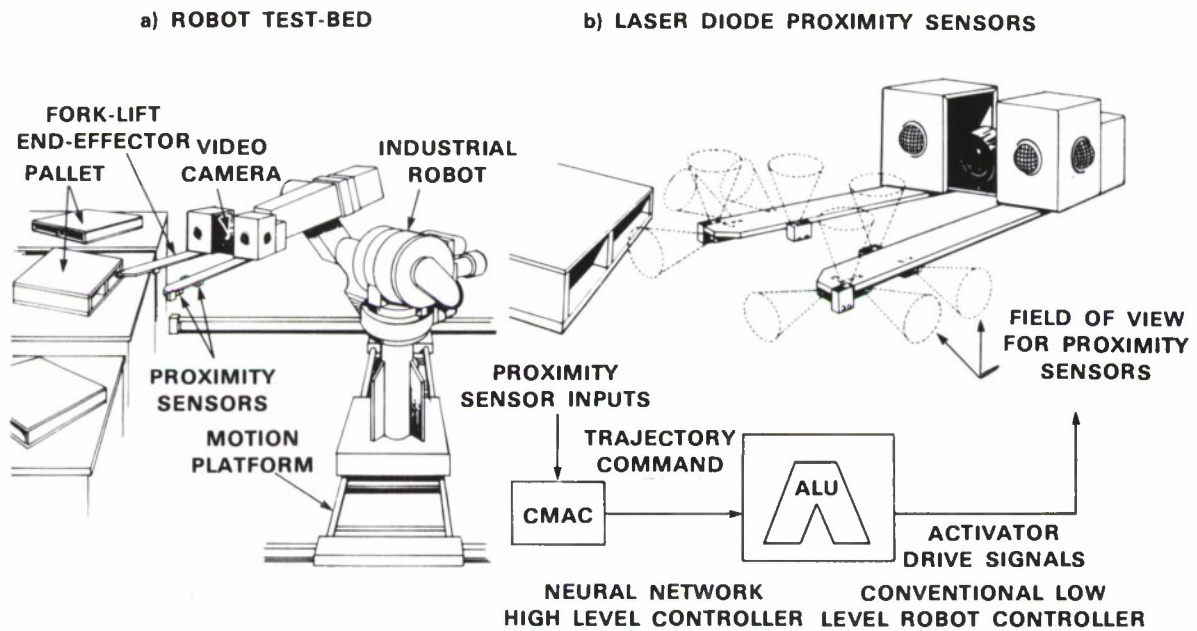


Figure F-2. Robot Test-bed to Study Sensory-guided Object Acquisition. The neural network is required to learn to use inputs from proximity sensors to generate trajectory commands to an industrial robot that enable the robot to insert the fork-lift into a pallet.

The basic approach to teaching the neural network is for a human operator to grasp the end effector and perform the pallet acquisition task (Figure F-3). This is accomplished using a 6-DOF force/torque sensor on the robot wrist as a joystick. The direction of the forces exerted by the operator is used to specify the desired trajectory command to the robot control system and to the learning procedure for the neural network. The goal of the learning procedure is to adjust the network weights so that a particular pattern of

proximity sensor inputs generates a trajectory command that is identical to or very similar to the command generated by the human operator. After some learning criterion has been achieved, the neural network output is connected to the low-level controller to carry out the pallet acquisition task autonomously.

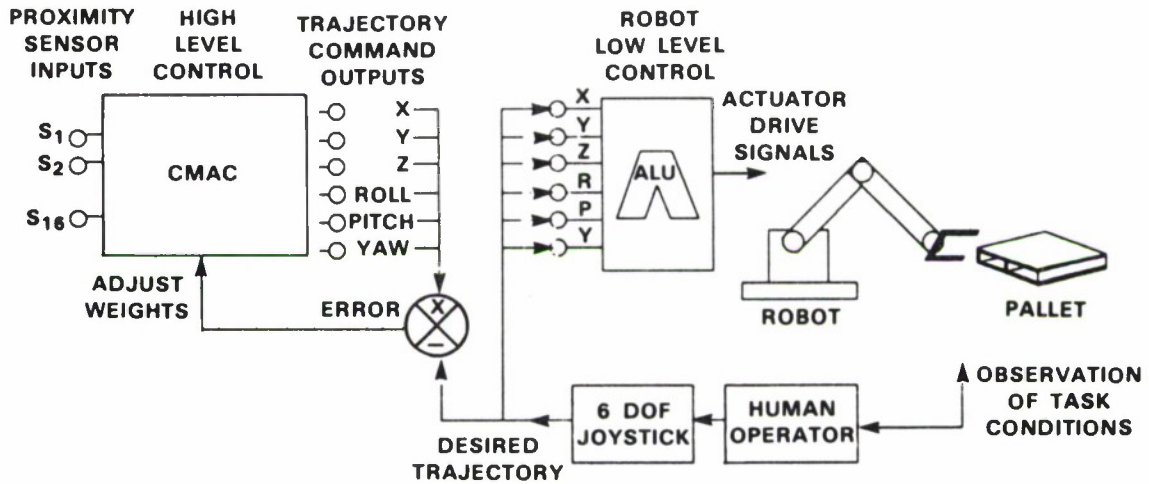


Figure F-3. Method for Teaching by Observation of Human Task Performance. The neural network learns to associate patterns of proximity sensor inputs with the trajectory commands generated by the human operator.

A major consideration in using a human as the teacher is making sure that the information available to the neural network is similar to the information used by the human to make a particular guidance adjustment. For example, fork lift movements made beyond the range of the proximity sensors obviously cannot be used as teaching examples. It is not necessary that the sensors of the human (vision, proprioception) and the neural network (range) be of the same modality. The requirement is for the network inputs to provide sufficient information that is correlated with the outputs generated by the human teacher.

It should be emphasized that in this application the neural network performs the high-level control function of formulating a movement trajectory, while the low-level servo control function is performed by a traditional servo system. Work is in progress to implement both high- and low-level functions in a structurally-specialized CMAC system. In this system, multiple CMAC modules will be arranged in a hierarchical manner to perform high- and low- level functions. In addition, there will be a parallel arrangement of CMAC modules dealing with different sensory modalities. By configuring relatively simple and easily understood modules into increasingly complex structures, it should be possible to generate complex robot behavior while avoiding the scalability and training problems inherent in large homogeneous neural networks.

F.3 RESULTS

As a preliminary experiment to the use of the force/torque sensor as a joystick, a manual teaching method was used. In this method, the desired trajectory command for a particular teaching situation is entered by a keyboard. With this technique, very encouraging preliminary results have been obtained. The neural network has learned to acquire the pallet from any arbitrary starting position with as few as seven teaching points.

To demonstrate the power of the teaching technique, a previously-trained neural network was presented with the problem of dealing with two pallets stacked on top of each other. The normally acquired bottom pallet was rotated away from the robot, making it impossible to acquire. Only one teaching point was required for the network to learn to ignore the bottom pallet and move up to acquire the top pallet. It should be emphasized that the teaching point is a particular point of the path – not an entire trajectory, as is done with a teach pendant. The neural network's ability to generalize allows it to specify the trajectory for points not previously encountered.

The significance of these results becomes clear when the immense number of possible input combinations is considered. Ten sensor inputs that are digitized to 100 values result in 100^{10} possible combinations. The fact that successful sensory-guided task performance was obtained in such a large sensory input space after only seven teaching instances demonstrates the tremendous potential of neural networks for robot control. The ability to learn complex control functions and to generalize in unexpected situations will allow robots with neural network controllers to function in environments that were previously impossible for traditional control systems.

APPENDIX G

HUGHES PARAMETER NETWORK FOR AUTOMATIC TARGET RECOGNITION (ATR) IN TACTICAL IMAGERY

M. Oyster
Hughes Aircraft, Electro-Optical & Data Systems Group
El Segundo, CA 90245

G.1 PROBLEM ADDRESSED

The problem addressed here is the automatic recognition of targets (ATR) in infrared, visible, or range imagery. This militarily important problem has been worked extensively by Hughes Aircraft with important successes, but many problems remain. The Hughes approach is unique in that a class of neural networks called *parameter networks* is employed. Parameter networks are particularly well-suited to the problem of object recognition in natural scenes. Parameter networks provide powerful mechanisms for handling the complex and unpredictable characteristics of three-dimensional targets, camouflage, sensor noise, and scene clutter. Since these are the key issues for solving the ATR problem, parameter networks promise to significantly advance the military effectiveness of ATR systems.

G.2 PREVIOUS APPROACHES

Previous approaches are best described by reviewing the generic current-generation ATR system (Figure G-1). The approach described in Figure G-1 is a statistical pattern recognition technique that involves the reduction of the image into a binary silhouette from which primitive shape features can be extracted. These shape features form the basis for object recognition.

The shape features are global attributes which belong to one complete object. In real imagery, frequently the global shape is too severely perturbed to reliably generate the features. Often, it is lost in the silhouette formation process, which is highly vulnerable to noise and variations in target and scene conditions. In addition, the statistical pattern recognition techniques do not provide a means for recovering from occlusion. These problems have been well known to the computer vision community for more than five years and a very large number of academic publications have addressed them. The methods developed in these publications are too many to mention here, but they have some important common characteristics. In order to cope with the uncertainties in natural scenes, highly redundant information is retained and the object recognition algorithms are able to function and produce reliable results given partial information.

The redundancy of the image data can be achieved by providing spatially localized features and the image can be transformed into a number of independent representations. Examples of image representations include: intensity maps, reflectance maps, edge maps texture maps, color maps, depth maps, Fourier

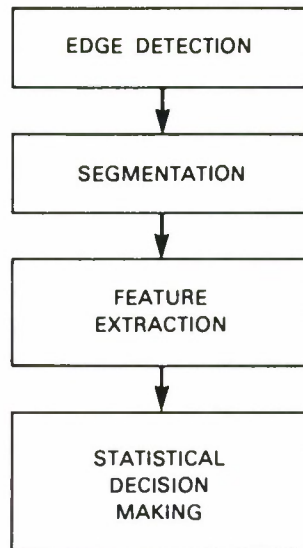


Figure G-1. Generic Current-Generation ATR.

spaces, among others. While redundant representations are recognized as being crucial to successful machine vision and ATR, the formation of many redundant representations in real time poses new challenges to computer technology. Also, the algorithmic problem of combining several sometimes conflicting image representations to recognize an object is still a topic of major research.

Some previous approaches have utilized AI technology. In a significant portion of this work, AI techniques have been employed to achieve robust image understanding by integrating multiple image representations. AI techniques have been limited by requiring the explicit encoding of a large number of rules – a very labor intensive process.

We believe that the problem of integrating a number of image representations is most naturally represented using neural networks. Hughes Aircraft is using neural networks for generating multiple image representations (or scene encodings), and we are also investigating neural networks for object recognition. In this paper we will discuss object recognition only.

G.3 HUGHES NEURAL NETWORK APPROACH

The Hughes ATR neural network is a generalization of the powerful parameter networks of Dana Ballard [1]. Parameter networks provide neural computational structures which transform intrinsic images into feature spaces. Patterns of neural activity in the feature spaces correspond to the recognition of objects from the intrinsic images. A simple example of a parameter network utilizes the edge image as the intrinsic image and points in the parameter space are represented by neurons which encode the grouping of edges into complete objects. While this model is very simple, it is also very powerful because the loss of some of the edge information does not prevent the recognition of the object.

In the Hughes ATR neural network, several intrinsic images are used simultaneously to map to the feature space. The larger and more varied the number of intrinsic images, the more detailed a representation of the object and its shape is possible. The organization of the neural units in the parameter space depends upon a mathematical model. The model defines how intrinsic image properties are encoded into shape, target signature or other object attributes. Proper choice of the mathematical model is crucial to preventing a combinatorial explosion of neural connections.

The Hughes ATR neural network uses a number of techniques for increasing the power of parameter networks, including:

- **Recursive refinement.** Initial parameter estimates are made at low resolution which guide the recognition of objects at increasingly higher levels of resolution. This greatly reduces the number of connections and speeds convergence of the network.
- **Multi-layer networks.** The number of parameters is reduced by transforming the problem into a hierarchical grouping of intrinsic image properties into progressively more detailed object representations.

Figure G-2 is a layout of the Hughes ATR neural network.

G.4 RESULTS

Hughes has developed a prototype parameter network and tested it on tactical imagery. These tests have shown this network to be superior to statistical pattern recognition. Advanced networks are expected to produce even higher performance. The network was tested against both occluded and non-occluded targets.

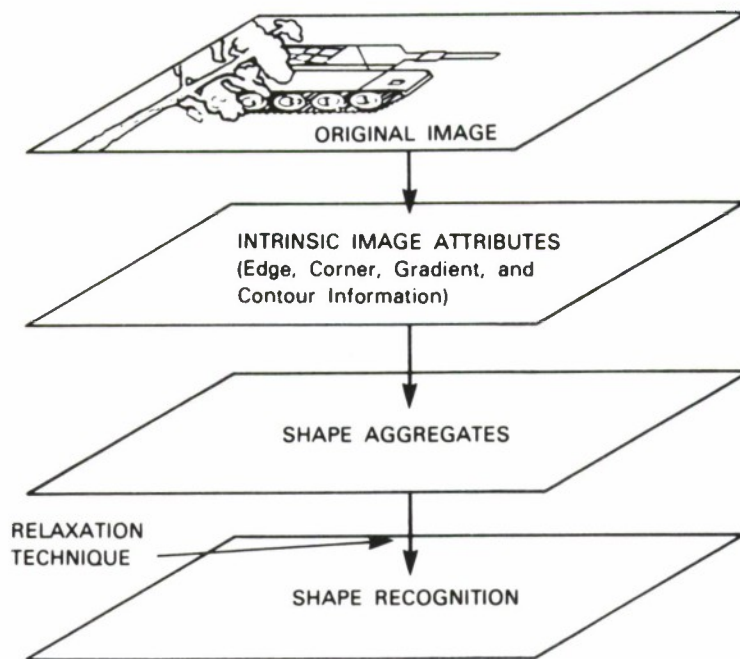
Figure G-3 shows a tank in an infrared image. Portions of the target image have been occluded by superimposing solid bars on the image. The parameter network was able to recognize the tank, whereas the conventional statistical pattern recognition technology was not. This differential performance can be demonstrated over a wide range of cases.

In addition, experiments were run over a large database of target imagery with realistic signal-to-noise ratios, in which none of the targets were occluded. The parameter network had a 20% higher probability of recognition than the conventional ATR algorithm. In battlefield environments with occluded targets, the performance gain would be substantially higher.

G.5 CONCLUSIONS

The Hughes ATR neural network directly addresses the most significant problem in computer vision: how to achieve robust object recognition in real time given unpredictable scene and sensor conditions. The network:

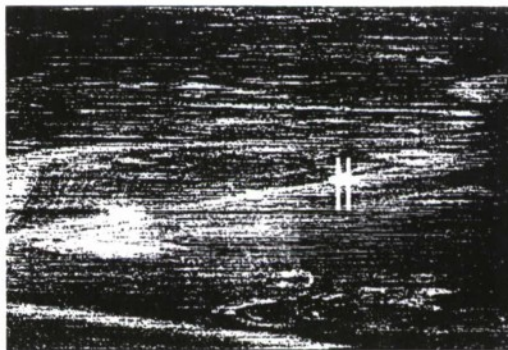
- Is massively parallel,



101882-47

Figure G-2. A Neural Network Process to Recognize Shape. Intrinsic image attributes are combined to form more complex shape aggregates. A relaxation labeling technique integrates the shape aggregates to produce a unique global shape interpretation.

RESULTS



101882-46

Figure G-3. Parameters Networks for Recognizing Occluded Targets.

- Utilizes distributed local process control, and
- Requires each processing element to perform a very simple pattern matching computation defined only by the connections between elements of the network.

The network makes decisions through competitive-cooperative relaxation processes. These relaxation processes constitute a parallel search for an object interpretation. The parallel search capabilities give it greater speed and flexibility than conventional AI methods, which must perform time-consuming backtracking and conflict resolution. For ATR problems this is crucial.

While for many applications, learning is a desirable characteristic of neural networks, it is not the key issue in neural networks for ATR. This point is particularly crucial with respect to ATR and other vision applications. It is simply not practical for a neural network to learn on its own all of the intermediate representations required to produce a solution to the ATR problem because the convergence times would exceed any reasonable time span. Parameter networks do not employ learning in their operations.

Parameter networks are a powerful model for solving ATR and vision problems. They exploit the essential desirable characteristics of neural networks while representing a prudent compromise to the learning convergence problem. Preliminary experimental results have shown that even the simplest parameter networks can outperform conventional statistical pattern recognition techniques. Major improvements in ATR performance can be realized through the implementation of such networks.

REFERENCES

1. Ballard, D. H., "Parameter Nets," *Artificial Intelligence*, **22** pp. 235-267, 1984.

APPENDIX H IMAGE CLASSIFIER

Murali Menon
MIT/Lincoln Laboratory
Lexington, MA 02173

The Neocognitron of Fukushima (1980) is a massively parallel multi-level neural network which performs visual pattern recognition, including classification of two-dimensional patterns. The model is invariant to shifts of the pattern in the plane, and can tolerate noise-corrupted and slightly deformed images. Its architecture models the anatomy of the human retina in a qualitative way. This system also behaves like the adaptive resonance model of Carpenter and Grossberg (1985) in that it is self-organizing and operates without a “teacher.” The Neocognitron has a demonstrated capability to discriminate alphabetical characters stored in a matrix of 16×16 pixels. Performance on handwritten characters in a 19×19 matrix was demonstrated by Fukushima (1983). A more recent study by Stoner and Schilke (1986) has confirmed the model’s ability to classify dot-matrix characters [VIS-18, p. 149]. While many accurate character recognition algorithms already exist, the Neocognitron is noteworthy because it handles positional shifts and moderate deformations in the shapes of input characters. These properties suggest that Fukushima’s model might be very useful in solving more demanding machine vision problems. Work at the Massachusetts Institute of Technology’s Lincoln Laboratory has produced a simulation of the Neocognitron on a serial machine. This program has operated successfully on wire frame images embedded in a matrix of 128×128 pixels. The model was able to classify images by extracting features from the input images and retaining only those whose response was above the average.

H.1 NEOCOGNITRON MODEL DESCRIPTION

The Neocognitron contains analog processing elements that are referred to as cells. There are two cell types: S and C-cells. Both types receive their input through a set of weighted inputs, and produce an output by applying a nonlinear function to the inputs. The particular functions used for S and C-cells are given by:

$$\begin{aligned} S - cell : \phi(x) &= x, \quad x > 0 \\ &= 0, \quad x \leq 0 \end{aligned} \tag{H.1}$$

$$C - cell : \psi(x) = x/(\alpha + x) \tag{H.2}$$

In addition to the functional differences in Equations H.1 and H.2, S-cells receive inputs through variable weights that are adjusted during training, whereas C-cells use fixed weights. When these cells are combined together in a multi-layer architecture, the model as a whole has the ability to classify patterns.

The architecture of the Neocognitron is shown in Figure H-1. The cells are arranged in two-dimensional square arrays called planes, indexed by the pointer (k). Planes of like cells are grouped together into layers called S and C-layers. An S-layer contains cells described by Equation H.1, while C-layers contain cells

described by Equation H.2. An S- and C-layer together form a level indexed by (l) . A cell in a given layer in a particular plane receives its inputs from a cluster of cells in a plane on the layer below. This cluster is referred to as the cell's receptive area. The size of the receptive area is fixed for any given layer. The output of each cell in the receptive area is modulated by an adjustable weight for S-cells receiving input from a receptive field of C-cells and by fixed weights for C-cells obtaining input from an S-layer. The position of a cell in the plane is given by (n) , and the relative offset in the receptive field by (ν) . The top layer contains single C-cells, which represent the classified output. The Neocognitron is strictly a feedforward network, where signals propagate from the input to the top layer. The S- and C-layers have distinct functions in the Neocognitron, and, when combined with a learning algorithm, allow relevant features to be extracted and grouped together to classify different patterns.

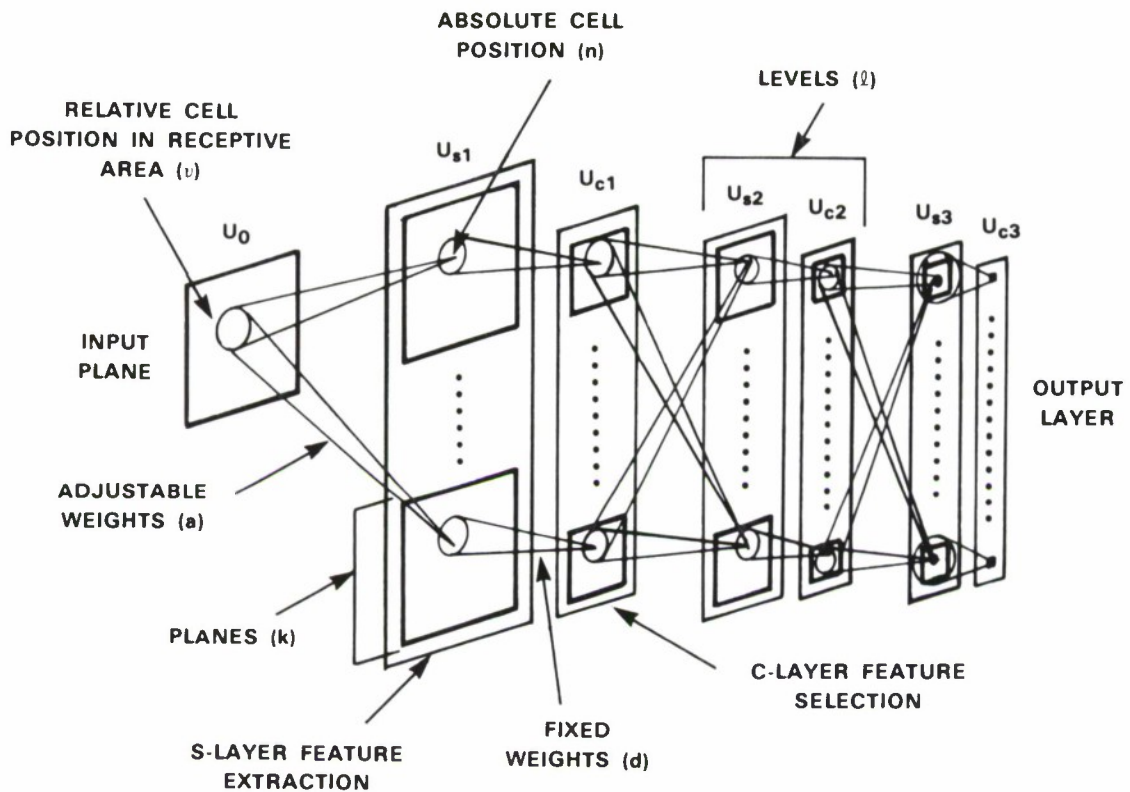


Figure H-1. Architecture of the Neocognitron.

The function of a S-layer is to extract features from the patterns of outputs of cells on the C-layer below. The output of a S-cell is given by:

$$S_{out(l)} = r_l \phi \left[\frac{1 + \sum^k \sum^{S_l} a_l \cdot C_{out(l-1)}}{1 + r_l \cdot b_l \cdot V_{Cl} / (1 + r_l)} - 1 \right] \quad (H.3)$$

The adjustable weights (a_l) in Equation H.3 are initialized to small random values at the start of a training sequence. The adjustable inhibitory weights (b_l) are initialized to zero. The term V_{Cl} is a weighted RMS value of C-cells in the receptive field (S_l) . In Equation H.3, a S-cell compares the receptive field

output in the previous C-layer modulated by a set of learned weights (a_l) to its RMS value; a non-zero response is obtained only when the weighted response is stronger than the RMS response by a certain amount. The parameter (τ_l) controls how selective the S-cell is to learned features (a_l) by scaling the ratio of weighted to RMS responses. A large value of (τ_l) produces a response only when the learned features and the actual output pattern in the receptive field match very closely. The learned weights (a_l) become sensitive to specific features through the action of a simple learning algorithm. The position and planes of the S-cells with a maximum output response are recorded, and the weights associated with these cells are updated proportional to the C-cell response in the receptive area,

$$\delta a_l(\hat{k}_l) = q_l \cdot c_l \cdot C_{out(l-1)}(\hat{n} + \nu) \quad (\text{H.4})$$

$$\delta b_l(\hat{k}_l) = q_l \cdot V_{Cl-1}(\hat{n}) \quad (\text{H.5})$$

In Equations H.4 and H.5, q_l is the rate of learning, and c_{l-1} is a fixed weighting function in the receptive area. Note that the maximum S-cell response for a position (\hat{n}) can occur on different S-planes (\hat{k}_l). Repeated application of Equation H.4 during training allows different planes on an S-layer to be sensitive to different features in the input.

Proper choice of the selectivity parameter (τ_l) will classify input patterns either uniquely, or according to differences in subsections of the inputs. The interesting point about this procedure is that learning occurs automatically.

The function of a C-layer is to select among the features that were extracted by the previous S-layer. The selection is based on those cells in the S-layer that have a response greater than the average response across all S-planes in the layer,

$$C_{out(l)} = \psi \left[\frac{1 + \sum^{D_l} d_l \cdot S_{out(l)}}{1 + V_{Sl}} \right] \quad (\text{H.6})$$

In Equation H.6, d_l is a fixed weighting function, usually taken to be unity, and V_{Sl} is the average S-cell response. A C-cell has a non-zero response if the S-cell output in the previous layer for a particular receptive area (D_l) is greater than the average output in the same receptive field across all planes in the S-layer. The top C-layer is used to encode different patterns by choosing the C-cell with maximum response for each input pattern.

Most implementations of the Neocognitron are multi-level systems with an S-layer and a C-layer at each level. Different patterns of connectivity decompose the input image into distinct spatial features and a learning rule reinforces those patterns which produce the greatest response. As this type of system learns, different S-cells become sensitive to distinct combinations of features in the input plane. The C-layer examines all feature groupings and rejects those that yield weak or mediocre output.

Successive levels act to recognize increasingly complex feature groupings. On the top C-layer, each cell comprises an entire plane. Each C-cell in this final layer produces its maximum response only when a specific input image is presented.

The model can tolerate shifts in the plane because each cell in a plane receives inputs through a weighted receptive field placed at different positions on the layer below. The use of overlapping receptive fields

allows different cells to respond to shifted versions of the same feature. The method is illustrated in Figure H-2. Application of the scheme in Figure H-2 across multiple layers produces a top layer C-cell that has identical responses for shifted versions of the entire input pattern. The same mechanism that gives the model shift invariance also allows patterns to be slightly deformed and still be correctly classified.

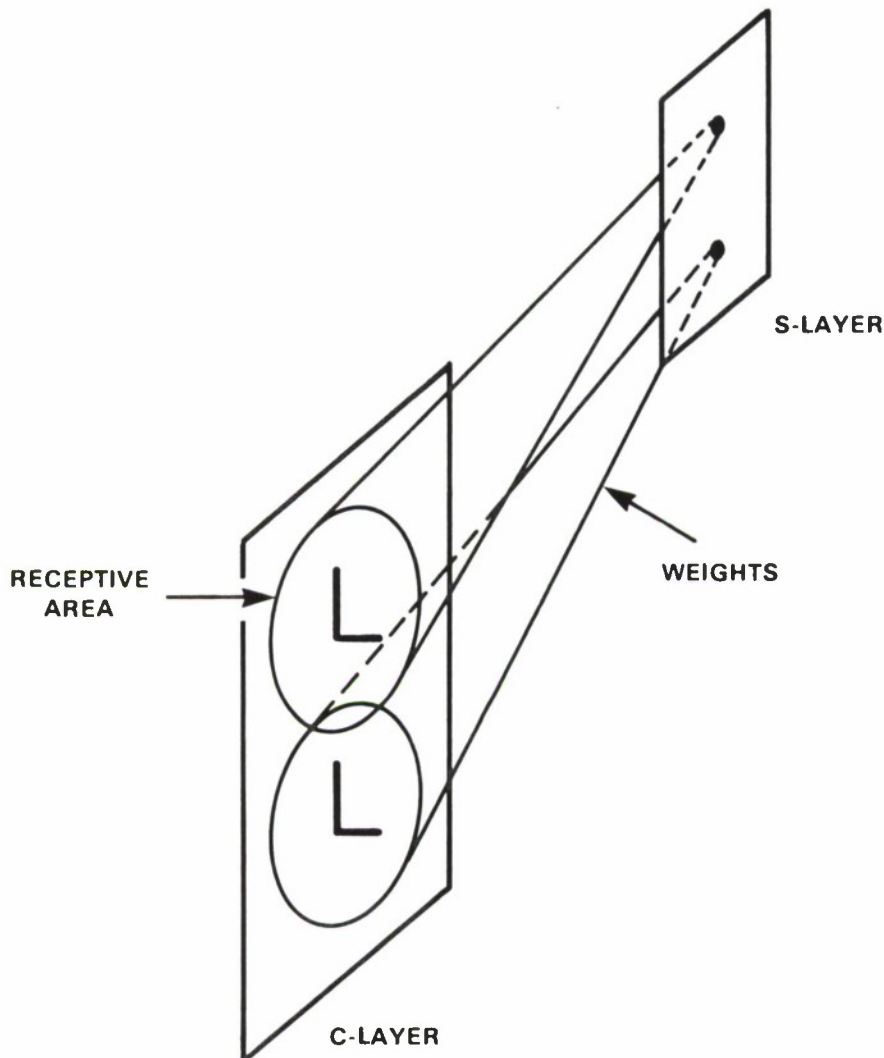


Figure H-2. Mechanism of Shift Invariance in the Neocognitron.

H.2 RESULTS

An example of the capability of the Neocognitron to classify patterns was demonstrated by training the system on a set of binary input patterns shown in Figure H-3. The Neocognitron was implemented as a software simulation on a minicomputer. With the proper choice of the selectivity parameter (r_l) at each

level, the model was able to classify each vehicle in Figure H-3. A four-level system with six planes per layer was used. Each input pattern produced a maximum response in a different position on the fourth (top) C-layer. The training was complete (stable weights) after 20 presentations of each of the input patterns in sequence. In the process of training, the system produced planes that are sensitive to different features in the input patterns. The features extracted by the level 1 S-layer for each of the six S-planes is shown in Figures H-4 and H-5, when input image 1 in Figure H-3 is presented to the system. The intensity scale in Figures H-4 and H-5 is proportional to the S-cell outputs. Each of the planes has a different set of responses: the fourth plane seems to be sensitive to horizontal features, while the sixth plane is sensitive to certain vertical features as well. The significance of this exercise is that features were selected, grouped together, and the patterns were classified without a "teacher." Further experiments need to be performed to determine the sensitivity of the model to noise, its ability to scale to a large number of patterns, and a methodology for choosing system parameters, e.g., r_i , at each level.

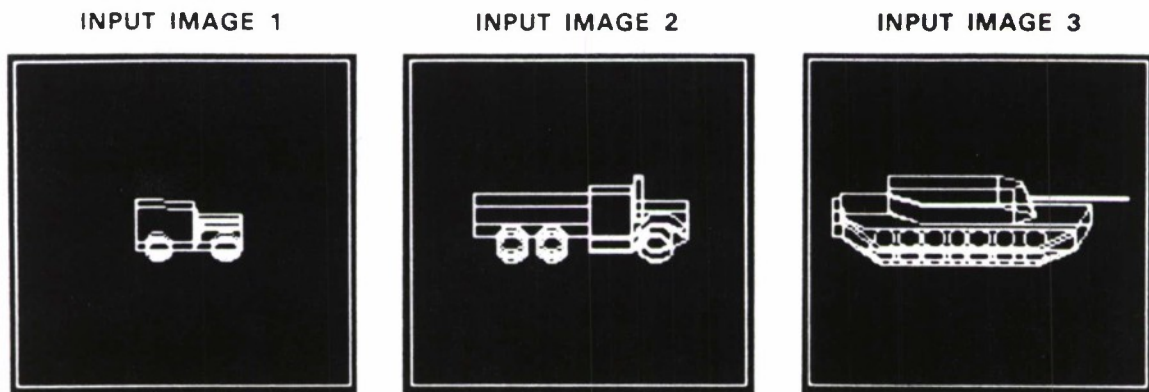


Figure H-3. Input Patterns Used to Train the System.

H.3 SUMMARY

The Neocognitron is a model for shift-invariant unsupervised classification of patterns. The model uses a multi-level architecture in a feedforward mode to extract and select features to produce a unique classification for each input pattern. The top-level C-layer contains planes with single cells whose positions are used to classify inputs. From an information processing point of view, the data rate is reduced by abstraction of the input pattern into sets of features. A software simulation confirmed the classification and shift-invariance abilities of the model. However, for input image resolutions of 128×128 pixels and higher, many hours of computer time are needed to classify a few patterns. The Neocognitron is very promising for pattern recognition problems but is best implemented in parallel analog hardware for problems of practical interest.

The Neocognitron model is very interesting from a neural network design point of view since it incorporates a modular, hierarchical structure where information is coded using a distributed representation across many levels. A rough analogy to biological neural systems can be made if each level containing a C- and S-layer is taken to be a module. This suggests a very difficult control problem in which a stable

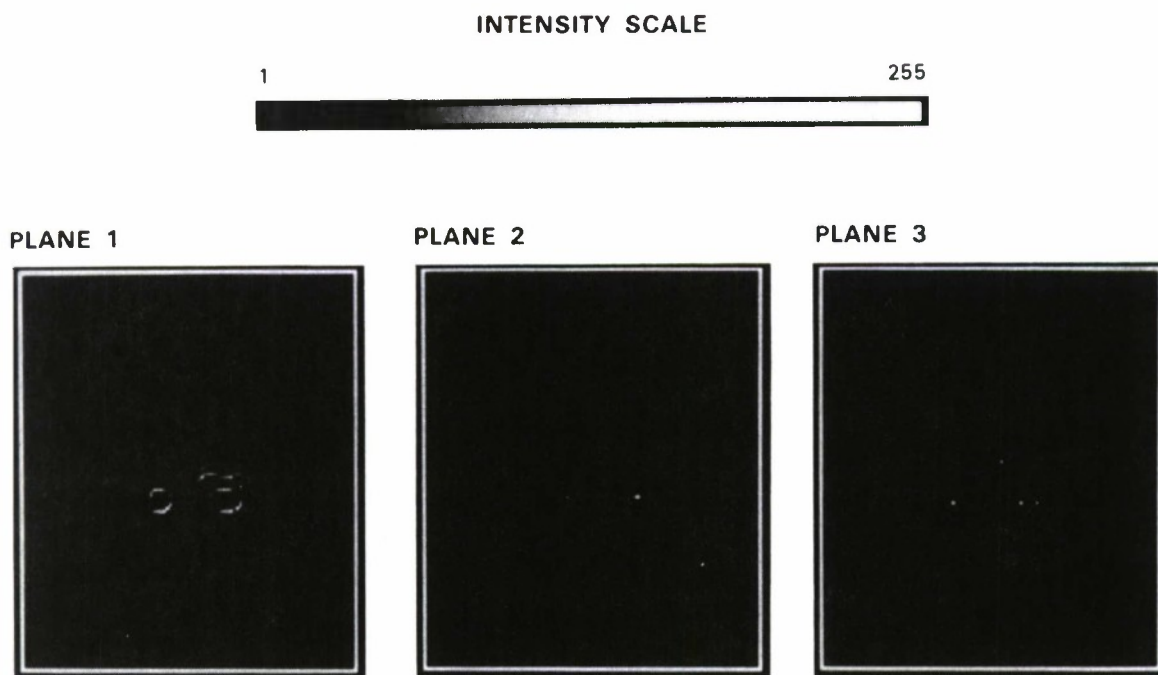


Figure H-4. First S-layer Output Response on Planes 1-3 for Input Image 1.

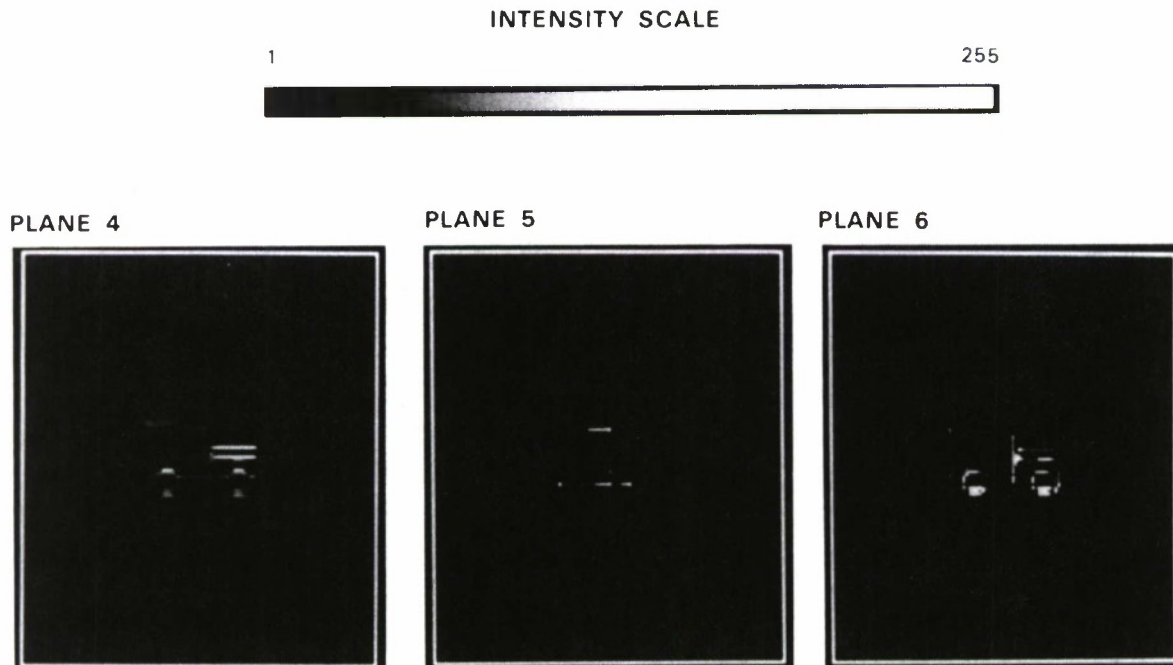


Figure H-5. First S-layer Output Response on Planes 4-6 for Input Image 1.

101882-37

101882-36

system must be designed from several modules with similar time constants. A weakness of the present model is that parameters, such as the selectivity (τ_l), must be specified for each module. The analogy with biological systems suggests that feedback between modules plays an important role in “tuning” parameters according to changes in the environment. An interesting extension to the present model would be to use feedback to adjust the selectivity at lower levels so as to provide the best classification at the top level. This would also elucidate some of the design principles used by evolution in building “real” neural systems.

REFERENCES

1. Carpenter, G. A. and Grossberg, S., "Brain Structure, Learning and Memory," in J. Davis, R. Newburgh and E. Wegman (Eds.), *AAAS Symposium Series*, pp. 1-49, 1985.
2. Fukushima, K., Miyake, S. and Ito, T., "Neocognitron: A Neural Network Model for a Mechanism of Visual Pattern Recognition," *IEEE Trans.Sys. Man & Cybern. SMC-13*, No. 5, pp. 826-834, 1983.
3. Fukushima, K., "Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position," *Biological Cybernetics* **36**, pp. 193-202, 1980.
4. Stoner, W. and Schilke, T. M., "Pattern Recognition with a Neural Net," *SPIE, Real Time Signal Processing* **698**, pp. 171-181, 1986.

APPENDIX I
**A MODEL RETINA WITH PUSH-PULL, COOPERATIVE RECEPTORS PROVIDING
ADAPTIVE CONTRAST SENSITIVITY AND RESOLUTION**

Michael H. Brill, SAIC
Falls Church, VA 22046
and
Doreen W. Bergeron, SAIC
Tucson, AZ 85711
and
William W. Stoner, SAIC
Billerica, MA 01821

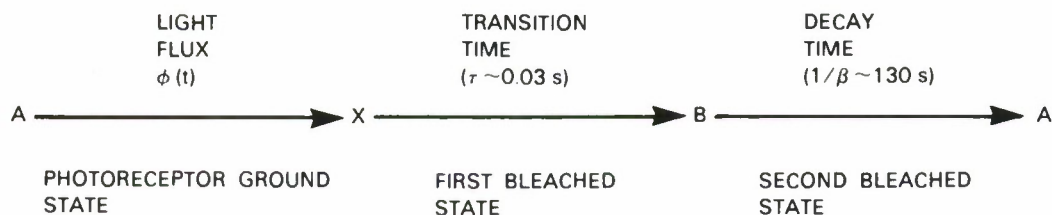
The visual sense is an instrument for survival, and has evolved under some of the same constraints facing designers of computer-vision systems. For example, during the course of a day, the outdoor light intensity varies by a factor of 10 billion, but real photosensors (either natural or man-made) tend to see contrasts over ranges of less than 1,000-to-one. This difficulty shows up, for example, in photographs of objects in deep shadows. Fortunately, most surfaces differ in reflectance by factors less than 100, and the lighting varies gradually in space or time (as when a cloud passes in front of the sun or a shadow boundary persists in time). Therefore, a photosensor needs to be sensitive only to relatively small changes of light intensity about a local average light intensity in space and time. The attainment of such light adaptation is a common experience in vision, and also a goal of computer-vision systems.

Another goal of a functional visual system is to enhance its light-gathering ability in dim light in order to combat photon noise. The most familiar way the human eye does this is to enlarge its pupil, thereby compromising spatial resolution (more aberrations are present than when the pupil is smaller). A second familiar mechanism is the shift to scotopic vision – switching in a receptor system (the rods) which is dense in photopigment and wired to perform spatial integration over large retinal areas. A third strategy – the subject of the present discussion – is to wire the photopic (cone) system so it adapts automatically to dim light by performing more spatiotemporal integration and less differentiation.

The shift in the set-point of dynamic range of vision is adaptive contrast sensitivity, and the variation of spatiotemporal properties with light intensity is adaptive resolution. We describe here a model retina [1], called Iris, that has both these properties. The model was simulated on a Sun Microsystems workstation in FORTRAN. Besides having attributes that are desirable for a robotic visual system, the model retina bears some instructive similarities to human vision. The photosensor design is based on a photoreceptor model, and the conducting grid may be realized by tight-junction coupling of receptors and by horizontal-cell interconnects.

In the model retina, transduction of light into receptor response is the result of ionic photoconductors embedded in the receptor cell membrane. Depending on its instantaneous state, each photopigment molecule in the membrane can open a conducting channel to either of two monovalent cationic species. When a photon is incident, the receptor quickly opens a channel to the first of these species, then slowly closes this channel and opens a channel to the other species, and finally returns to a non-conductive state

(by the addition of metabolic energy). The ionic species are driven by membrane voltages in opposite directions, so the receptor acts in a “push- pull” way. As in human vision, the voltage response is zero to a sustained (steady-state) light; the response versus log-intensity function shifts to the time-averaged intensity, thereby giving the adaptive contrast sensitivity desired for a visual system with limited dynamic range. Also, the receptor’s response is governed by photopigment kinetics whose rate increases with light level, so its temporal resolution increases with light level. Hence the model retina has adaptive temporal resolution – as desired to defeat photon noise. The photopigment kinetics are described by a set of coupled first-order differential equations provided in Figure I-1.



- **FIRST-ORDER KINETIC MODEL WITH LIGHT FLUX ϕ AS A COEFFICIENT**

- **DIFFERENTIAL EQUATIONS**

$$\dot{A}(t) = -\phi(t) A(t) + \beta B(t)$$

$$\dot{B}(t) = \phi(t - \tau) A(t - \tau) - \beta B(t)$$

$$\dot{X}(t) = \phi(t) A(t) - \phi(t - \tau) A(t - \tau)$$

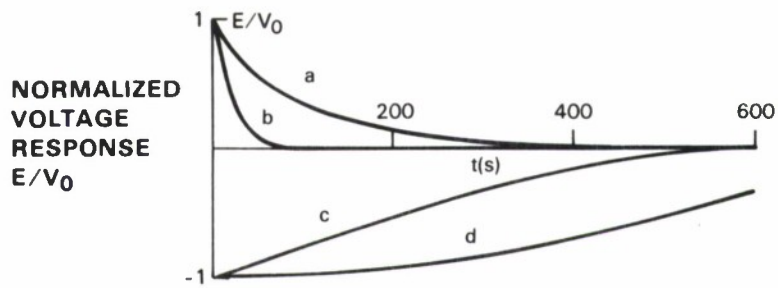
$$A + B + X = C = \text{CONST.} \approx 10^8$$

- **A, B, X ARE POPULATIONS OF PHOTOPIGMENT MOLECULES IN A CONE RECEPTOR**

Figure I-1. Photopigment Population-transition Model for the Iris Retina.

The adaptive time behavior of the model receptor has an additional helpful property borne out in our computer simulations. A pulse of light causes a response that decays slowly to steady state; however, a pulse of darkness produces only a short-duration change of response. This is fortunate behavior for an eye that has to blink. See Figure I-2 for simulations of the model receptor response to changes in light level.

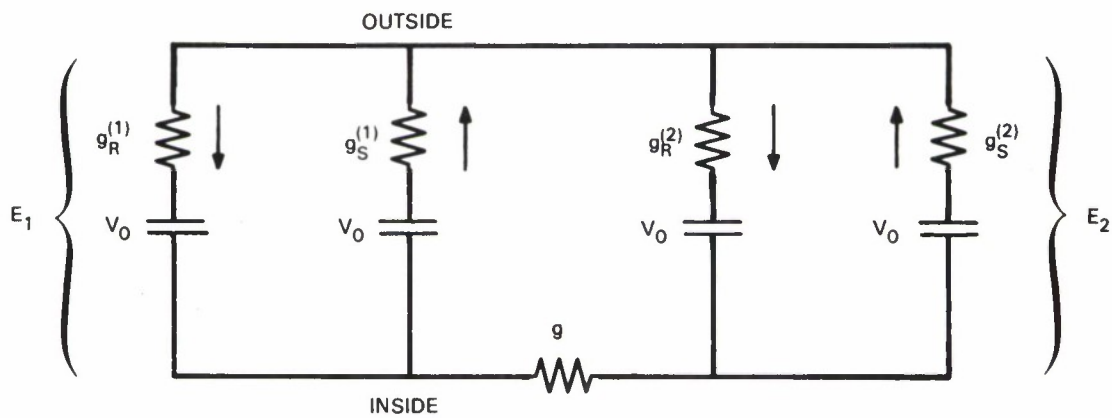
The adaptive temporal resolution of the model retina has a simple counterpart in the spatial domain. The model retina is tiled with a lattice of photoreceptors, wired together with a passively conducting grid of constant conductivity. The visual signal for each receptor – a transmembrane voltage – is now modified by lateral interaction (see Figure I-3). When the retinal illumination is (and has been) very low, most of the current passes between receptors when light hits one of them, and the receptors are functionally coupled. However, when the eye is light-adapted, the receptors tend to keep to themselves; very little current flows between them. This behavior shows up as an “iris” of lateral influence, contracting in the presence of light and dilating when light is removed. It provides a simple mechanism whereby visual acuity can be traded off against light sensitivity as the prevailing light gets dimmer.



	INITIAL FLUX	FINAL FLUX	
a	10^{-5}	10^{-3}	POSITIVE STEP INPUT
b	10^{-3}	10^{-1}	
c	10^{-1}	10^{-3}	NEGATIVE STEP INPUT
d	10^{-3}	10^{-5}	

FLUX UNITS ARE PHOTONS/MOLECULE/ s (ppms)

Figure I-2. Temporal Receptor Response to Light Steps.



- LATERAL CONDUCTANCE $g = \text{CONSTANT}$
- MEMBRANE CONDUCTANCES g_R, g_S INCREASE WITH INCREASING ϕ
- IF ϕ LARGE $\rightarrow g \ll g_R^{(K)}, g_S^{(K)}$
 \rightarrow UNCOUPLING IN BRIGHT LIGHT

Figure I-3. Intensity-dependent Spatial Coupling of Receptors.

Results of an Iris simulation provided in Figure I-4 demonstrate this intensity-dependent spatial resolution. The test pattern is split into two portions along the diagonal. Above the diagonal, a pattern of black and white squares recedes into the upper left corner. Below the diagonal there is a sinusoidal pattern with linear FM modulation in the radial direction (Fresnel pattern). The limiting resolution is determined by the discrete photoreceptor sampling at high light levels. This is demonstrated in Figure I-4, where the peak light level is 0.12 of the level that would bleach 10% of the photopigment into a state which requires a metabolic input before returning to the ground state. At one-tenth this light level (see Figure I-4), a small decrease in resolution is evident because of spatial averaging among nearby photoreceptors. A further ten-fold decrease in the light level results in the obvious reduction in spatial resolution shown in Figure I-4. Such intensity-dependent resolution is also found in human vision.

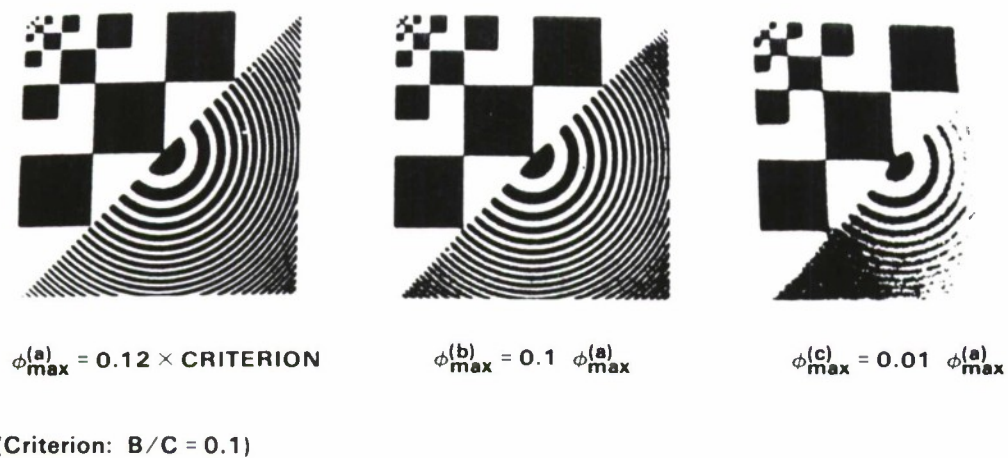


Figure I-4. Intensity-dependent Adaption of Resolution in the Iris Model Retina for a Squares/Zone Plate Input Test Pattern. (Left Image:) Light level on the model retina reduced by a factor 0.12 from saturation. (Middle Image:) Light level reduced by a factor 0.10 compared with (Left Image). (Right Image:) Light level reduced by a factor 0.01 compared with (Left Image).

In common with human vision, the model retina displays Weber's law (proportionality of increment threshold to a pre-existing background intensity over a much greater intensity range than the instantaneous dynamic range of the receptor).

Iris was designed to give repeatable response in dim light to visual scenes that are nominally the same except for photon noise. The repeatability was bought at the expense of spatiotemporal resolution. Repeatability of percepts from the same reflecting objects is a necessary but not sufficient condition for lightness constancy and color constancy – the illuminant-invariant assessment of reflectance information.

When implemented in VLSI hardware, Iris promises to be very helpful in robotic pattern recognition – in which invariances must be computed robustly in the presence of noise. The robotic application will be appropriate even when the dynamic range of photosensors is extended to exceed that of human vision. A large extension of dynamic range at the photoreceptor level must always be in the direction of increasing intensity; increased dynamic range at low intensities requires mechanisms like those in Iris to pool the photoreceptor responses in space and time to provide a reliable signal in the presence of photon noise.

The next logical step in developing Iris is to implement the model in analog electronics by means of a two-dimensional lattice of photosensors attached to a passively conducting grid. The implementation of early visual processing in silicon has some precedent. Carver Mead of Caltech has developed a chip that contains receptors with a very large dynamic range [2]. These receptors have a logarithmic nonlinearity, and are connected by a resistive grid that offers center-surround organization and makes the output of the receptor array independent of change of light intensity by the same factor over the entire array. However, there does not seem to be an adaptation “iris” on the receptive field size of the output channels that pools the photoreceptor responses adaptively to the light level in order to combat photon noise.

Another retinal model which does have an intensity-dependent spatial weighting function is called IDS [3,4]. The Iris model differs from IDS in that Iris incorporates temporal as well as spatial adaptation, at the expense that the model does not seem to admit the elegant closed-form expressions and theorems of IDS. Also, Iris is based on an explicit circuit model that was, in turn, motivated by some facts of retinal neurophysiology and therefore may be a useful simulation tool for visual modeling.

Since the retina is a part of the brain, it is natural to speculate further that the adaptive contrast sensitivity and adaptive resolution mechanisms of the retina are used elsewhere in the brain. The adaptive contrast sensitivity mechanism of the retina would also be useful in “adapting out” certain unwanted signals, for example the presence of several pure tones that mask a more complicated sound signal such as the human voice.

Similarly, the adaptive resolution mechanism found in the retina would be helpful in stimulus generalization and pattern recognition. At high levels of stimulation, more discrimination should be possible than at weaker levels since decisions must be robust against random events such as cluttered, noise inputs and internal processing noise.

REFERENCES

1. Michael H. Brill, Doreen W. Bergeron, and William W. Stoner, "Retinal Model with Adaptive Contrast Sensitivity and Resolution," *Applied Optics*, **26**, pp. 4993-4998, 1987.
2. Carver Mead and M. A. Mahowald, "A Silicon Model of Early Visual Processing," *Neural Networks*, **1**, pp. 91-97, 1988.
3. T. N. Cornsweet and J. I. Yellott, "Intensity-dependent Spatial Summation," *J. Opt. Soc. Am.*, **A2**, pp. 1789-1786, 1985.
4. J. I. Yellott, "Photon Noise and Constant-volume Operators," *J. Opt. Soc. Am.*, **A4**, pp. 2418-2448, 1987.

APPENDIX J

TRW INVESTIGATION OF SCAN-SCAN CORRELATION AND MULTI-TARGET TRACKING

R. Kuczewski
TRW ANS Center
San Diego, CA 92128

J.1 PROBLEM AREA

Multi-target tracking and scan-scan track association is an important function in avionics systems, air traffic control, and SDI. The problem of multi-target tracking involves associating sequences of sensor reports with corresponding object. For optimal solutions, the time and computation required to rigorously solve the problem grows with $n!$, where n is the number of tracks, and leads to computational overload if n increases beyond the expected number of reports and there are no target unique identification means.

In the simplest case of the multi-target tracking problem, the input data is a collection of instantaneous “snapshots” of a group of moving objects (targets) whose paths may intersect or overlap; and the goal is to identify the path (track) each object has traveled. Figure J-1 shows such a sequence of snapshots. The paths are identified by correlating the snapshots using some information about the probable motion of the objects. The desired solution is an assignment of tracks which best satisfies the known constraints on the dynamics of the objects. The problem just described is a two-dimensional spatial tracking problem; in a general problem, objects may be tracked through higher-dimensional spaces which include features other than location, such as electromagnetic emissions.

J.2 PRIOR APPROACHES

The multiple track assignment problem has traditionally been solved using sequential search algorithms to test all track association hypotheses using higher-dimensional information returns where possible to permit pruning the tree search. This task is frequently performed by human operators in the air traffic control mode, where many targets are identified by specific labels, and the operator provides the probabilistic estimate of the track associations for other targets.

One obvious method of solving the multi-target tracking (MTT) problem is to exhaustively search all possible solutions and choose the one which best satisfies the constraints on the motion of the objects. This method is guaranteed to find the best solution; but the computing time increases as $n!$, where n is the number of objects being tracked. The complete search takes too long for even a modest number of objects.

Traditional approaches to this problem attempt to reduce the search space using heuristic tree-pruning techniques or by using gating functions which put hard limits on the possible motion of the objects. A

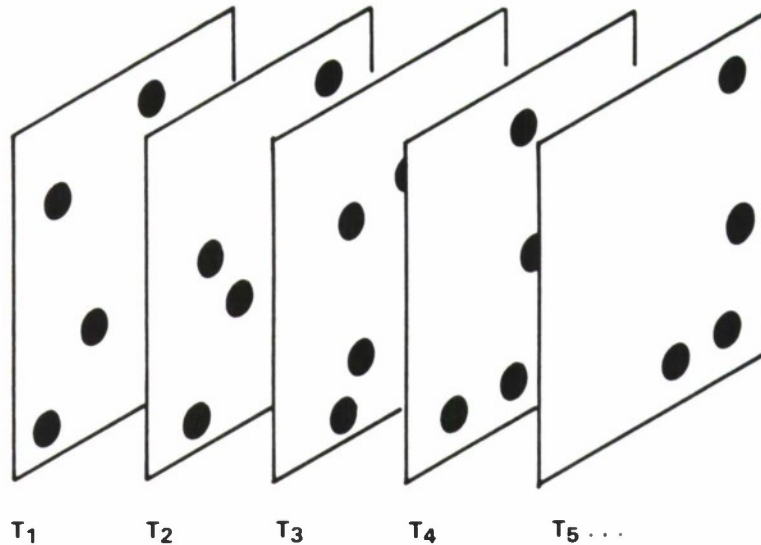


Figure J-1. Input Frames for Multi-target Tracking.

typical approach is a two-step process. First, a gating function is applied which discards many possible solutions, then a heuristic algorithm is used to choose the best one of the remaining solutions. In this approach, the gating function must make very safe assumptions, or the correct solution may be discarded before it can be examined. However, if the gating function leaves too many possibilities, it may take too long to choose among the remaining solutions. Even if the gating function and heuristic are well chosen, the processing time still rises as the number of objects increases.

J.3 NEURAL NETWORK APPROACH USED

We designed and successfully implemented a neural network solution to the multi-target tracking problem which is similar to the conventional approaches exercised by traffic control operators. We implemented our solution for the two-dimensional spatial tracking problem, but our concept is extendable to any number of dimensions. The technique we invented for our MTT system is called *Interpolative Probability Fields (IPF)*. This technique is a combination of the Boundary Contour System of Grossberg and Mingolla [1,2] with interpolation based on the probabilistic dynamics of the targets. The input to our system is a collection of snapshots such as shown in Figure J-1. The IPF network finds the most likely heading of each object at each time instant, which allows us to “connect the dots” and assign tracks to the objects.

The IPF network contains knowledge about the probable motion of the objects. In general, any probabilistic function can be used; we chose a Gaussian distribution in distance, turning radius, and heading deviation. Figure J-2 shows the resulting distribution function for a specific location and heading. The darkness of the shaded areas to the left and right of the object shown indicate the probability that it occupied that location in the previous time frame and the probability that it will occupy a location in the next time frame respectively.

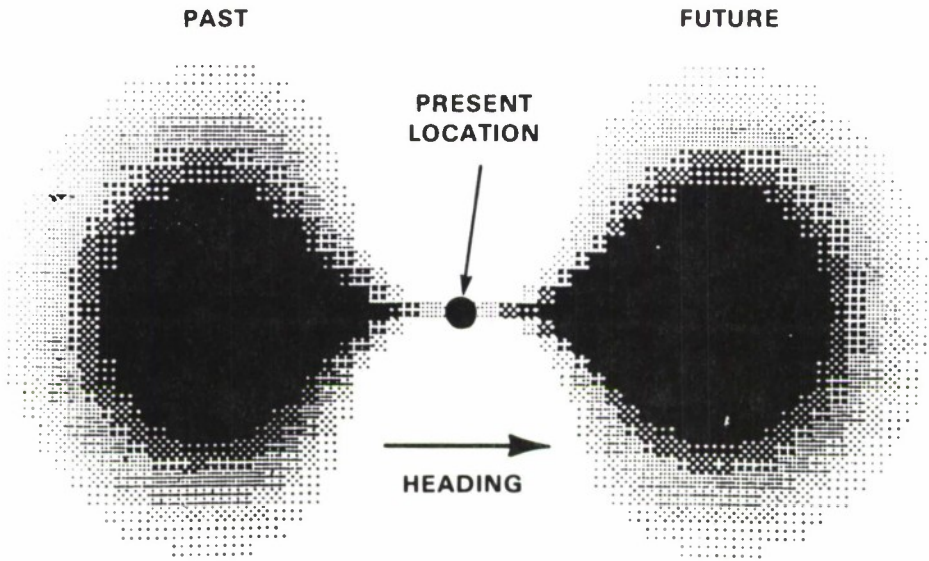


Figure J-2. Probable Object Motion Distribution.

The IPF system (and neural networks in general) consists of a highly- interconnected array of processing elements (PEs). The interconnections can have positive or negative weights of any value. Knowledge is represented by the values of the weights on the interconnections. In the IPF, one PE is assigned to each point in feature space. Here, the feature space has four dimensions: location (x, y), time (t), and heading (θ). The output value of a PE represents the probability of that point being included in the solution to the problem. The PEs are interconnected so as to find the best global solution to the problem by finding the best local solutions that are consistent with neighboring solutions. For example there are excitatory connections from each PE to points in the next time slot which correspond to the shape of the probability distribution of Figure J-2 (indicating an increased probability that the object occupies the dark areas in the next or previous time slots); but there are also inhibitory connections between PEs that represent the same location and time but different headings (forcing the network to select only one heading for a given location and time). As the network iterates, passing the output values of each PE along the interconnections to other PEs, the solution emerges.

J.4 RESULTS

We tested the IPF network with several typical tracking problems. Figure J-3 shows a vector display of the IPF network solving a two- track problem. The network solves the tracking problem by choosing one of eight possible headings for the objects at each time instant. The length of the vectors corresponds to the network's "certainty" that the object is heading in the direction of that vector. Initially, all headings are assumed to be equally likely, and the vector pattern looks like a starburst. As the network iterates, the best heading emerges for each point.

2 TRACKS

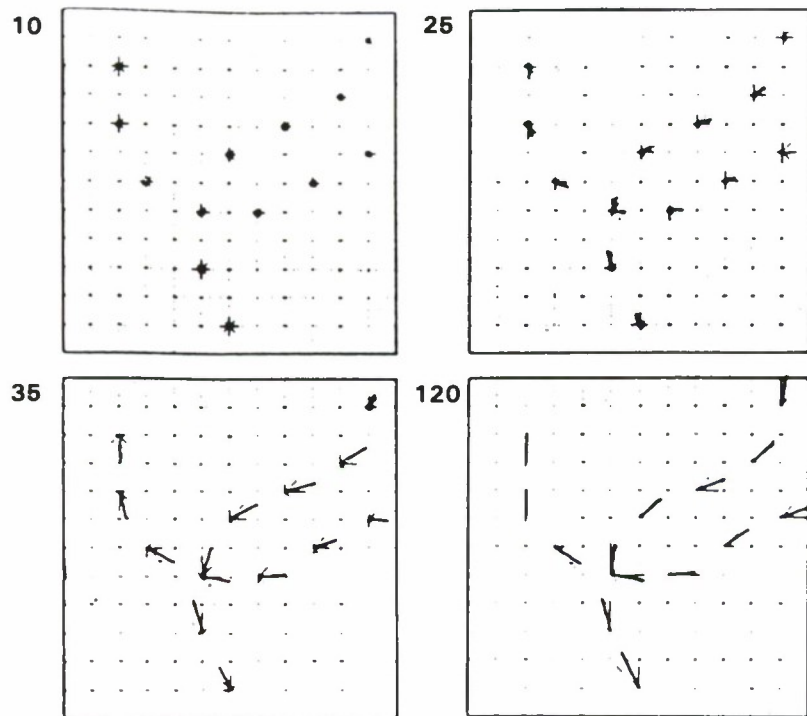


Figure J-3. Multi-target Tracking by Interpolation Probability Field Technique: Simulation of two objects with eight possible headings.

Figure J-4 shows the IPF network solving a 15-track problem. This example demonstrates that the processing time of our system is independent of the number of targets. The network solved the 15-track problem in about the same time as the two-track problem.

15 TRACKS

ITERATION No.

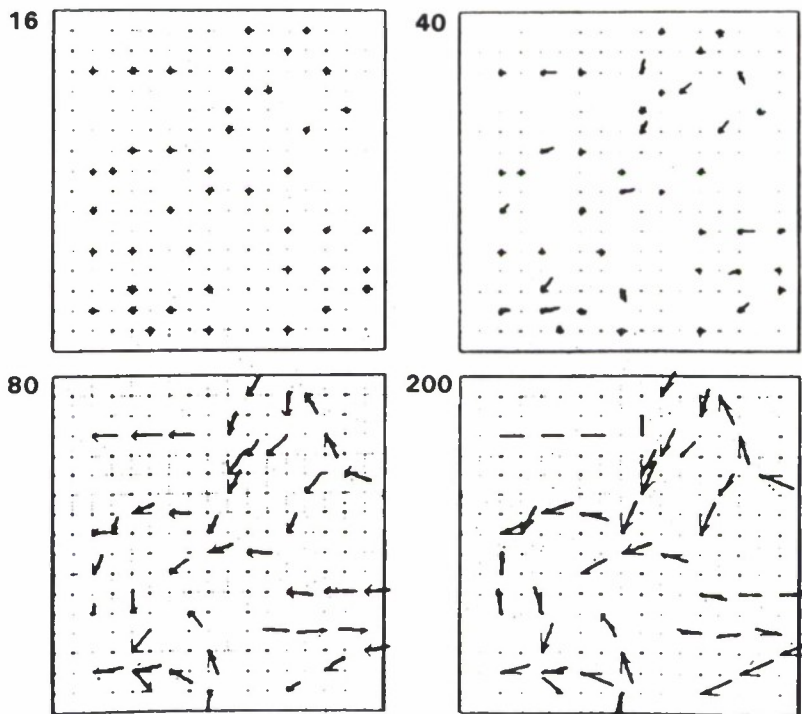


Figure J-4. Multi-target Tracking by Interpolation Probability Field Technique: Simulation of 15 objects with eight possible headings.

A difficult problem in multi-target tracking is designing a system which can handle late-arriving data. If a conventional, algorithmic system has found a good solution to a problem and some back-dated input data arrives, it may have to start all over in assigning tracks. Our IPF system can handle late-arriving data very well because the system naturally retains some uncertainty in its solution. Figure J-5 demonstrates the network's ability to "change its mind" due to late-arriving data. The input data is from two tracks which cross in an "x" pattern. The data from the third time frame, which happens to be the center of the "x," arrives late. Initially, the network selects two curved tracks, but there is still some uncertainty in the headings close to the center. Then the late data arrives, the network continues to iterate, changing its answer to be consistent with the new data, and finds the correct solution. The additional processing time required to incorporate the late data is less than the time it would take to start over with the complete set of data.

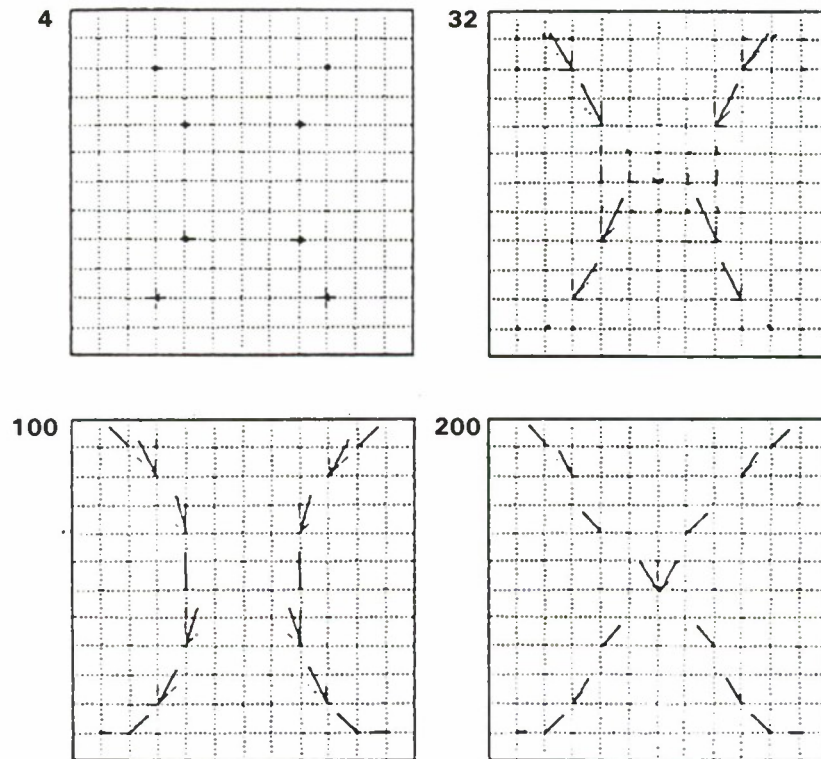


Figure J-5. Ambiguous Two-Track Example. In this problem, there are two tracks which cross in an "x" pattern, but the data for the center of the "x" arrive late. The network first chooses two curved tracks, but retains some uncertainty in the center. When the late data arrive, the network modifies its solution to be consistent with the new data.

The IPF system could be implemented in hardware in the near term using available technology. An IPF is locally-connected in a repetitive fashion and does not require modifiable weights. These characteristics suggest a straightforward parallel digital hardware. A fully parallel digital hardware could solve each of the sample problems we studied in about $50 \mu s$. The same characteristics of the IPF also make it a good candidate for an efficient optical implementation in the longer term as optical technology improves.

J.5 FUTURE PLAN

Future research will focus on several areas. First, we will look for near-term fieldable hardware implementations of the IPF. Second, we will study the application of neural network training techniques to the IPF to get it to learn the probabilistic relationships which we now must specify beforehand. Finally, we will investigate the total neural network solution to MTT, which combines an IPF filter and a higher-level cognitive structure that uses other features. This would be a very high payoff solution to the multi-target tracking problem.

REFERENCES

1. Grossberg, S. and Mingolla, E., "Neural Dynamics of Perceptual Grouping: Textures, Boundaries, and Emergent Segmentations," *Perception and Psychophysics*, **38**, pp. 141-171, 1985.
2. Grossberg, S. and Mingolla, E., "Neural Dynamics of Surface Perception: Boundary Webs, Illuminants, and Shape-from-Shading," *Computer Vision, Graphics, and Image Processing*, **37**, pp. 116-165, 1987.

APPENDIX K

MULTIDIMENSIONAL IMAGE FUSION AND SEGMENTATION

S. Grossberg
Center for Adaptive Systems
Boston University
Boston, MA 02215

K.1 THE CONTEXT-SENSITIVE PROCESSING OF NOISY MULTIDIMENSIONAL IMAGE DATA

Many AI algorithms for machine vision have been too specialized for applications to real-world problems. Such algorithms are typically designed to deal with one type of information – for example, boundary, disparity, curvature, shading, or spatial frequency information. Moreover, such algorithms typically use different mathematical schemes to analyze each distinct type of information, so that their unification into a single general-purpose procedure is difficult at best. For such AI algorithms, other types of signals are often contaminants, or noise elements, rather than cooperative sources of ambiguity-reducing information.

When humans gaze upon a scene, our brains combine several different types of locally-ambiguous visual information to rapidly generate a globally consistent and unambiguous representation of color-and-form-in-depth. What new principles and mechanisms are needed to understand how multiple sources of visual information automatically cooperate to generate a percept of three-dimensional form?

The Center for Adaptive Systems has been developing such a general-purpose automatic vision architecture. This architecture clarifies how scenic data about boundaries, textures, shading, depth, multiple spatial scales, and motion can be cooperatively synthesized in realtime into a coherent representation of form-and-color-in-depth that is more informative than a representation derived from any one type of scenic data taken in isolation. Moreover, it has become clear through cooperative work with colleagues at MIT/Lincoln Laboratory that the same processes which are useful to automatically process visual data from human sensors are equally valuable for processing multi-dimensional image data from laser radar sensors. These processes are called *emergent segmentation* and *featural filling-in*.

These processes are carried out by nonlinear interactions between a pair of parallel systems called the Boundary Contour System (BCS) and the Feature Contour System (FCS) (Figure K-1). Such an architecture promises to provide improved automatic processing of noisy multi-dimensional image data, as well as to provide a new technique for penetrating camouflage. This neural network architecture also possesses a regular geometry which makes it suitable for implementation in a compact hardware realization that will run in realtime in government and industrial applications.

The theory can be motivated through an analysis of the sensory uptake process. Such an analysis shows that there exist fundamental limitations of the visual measurement process at each stage of neural processing. The theory shows how the nervous system *as a whole* can compensate for these uncertainties using

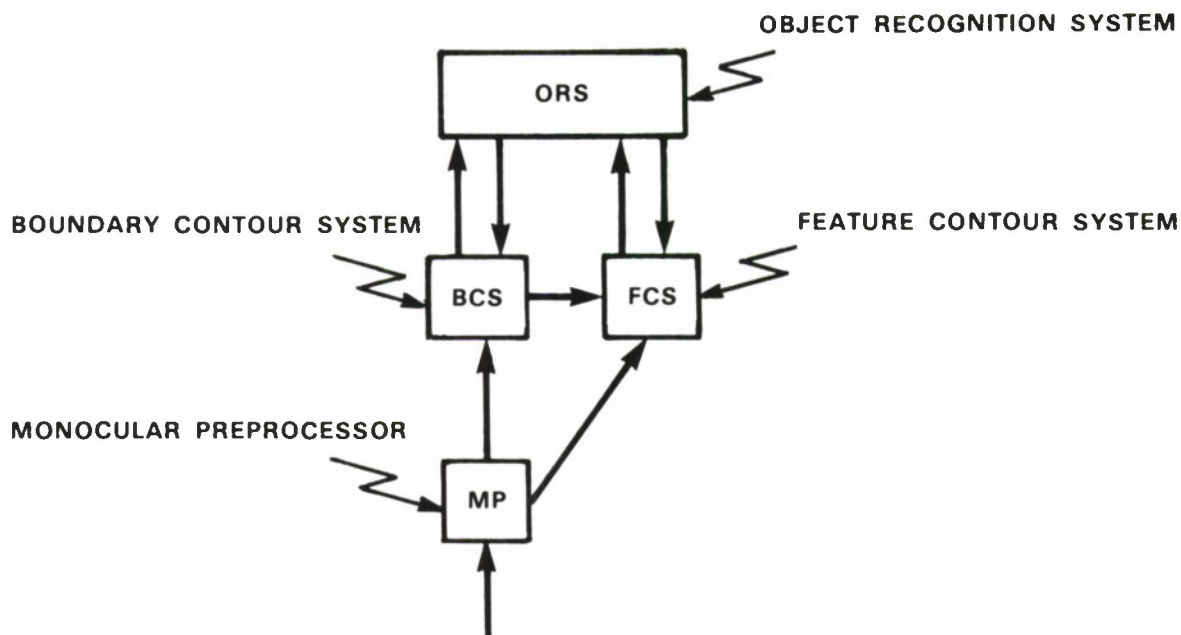


Figure K-1. Automatic Processing of Noisy Multi-dimensional Image Data Using Emergent Segmentation and Featural Filling-In. A macrocircuit of processing stages: Monocular preprocessed signals (MP) are sent independently to both the Boundary Contour System (BCS) and the Feature Contour System (FCS). The BCS pre-attentively generates coherent boundary structures from these MP signals. These structures send outputs to both the FCS and the Object Recognition System (ORS). The ORS, in turn, rapidly sends top-down learned template signals to the BCS. These template signals can modify the pre-attentively completed boundary structures using learned information. The BCS passes these modifications along to the FCS. The signals from the BCS organize the FCS into perceptual regions wherein filling-in of visible brightnesses and colors can occur. This filling-in process is activated by signals from the MP stage. The completed FCS representation, in turn, also interacts with the ORS.

both parallel and hierarchical stages of neural processing. Thus the visual nervous system is designed to achieve *heterarchical compensation for uncertainties of measurement*.

Rules for monocular boundary segmentation and featural filling-in were discovered through an analysis of how each type of process interacts with, and complements deficiencies of, the other [3,7,8,9,10,11]. It was then noticed that these rules for monocular boundary segmentation and filling-in also provide a basis for analysing stereopsis and the suppression of binocular double images [4,5,6]. Such results suggest that the popular hypothesis of independent modules in visual perception is both wrong and misleading. Specialization exists, to be sure, but its functional significance is not captured by the concept of independent modules.

K.2 THE BOUNDARY CONTOUR SYSTEM AND THE FEATURE CONTOUR SYSTEM

Our approach specifies both the functional meaning and the mechanistic interactions of the model microcircuits which comprise the macrocircuit schematized in Figure K-1. This macrocircuit is built up from the two systems mentioned above, the BCS and the FCS.

The BCS controls the emergence of a three-dimensional segmentation of a scene. This segmentation process is capable of detecting, sharpening, and completing boundaries; of grouping textures; of generating a boundary web of form-sensitive compartments in response to smoothly shaded regions; and of carrying out a disparity-sensitive and scale-sensitive binocular matching process. The outcome of this three-dimensional segmentation process is perceptually invisible within the BCS. Visible percepts are a property of the FCS.

A completed segmentation within the BCS elicits topographically-organized output signals to the FCS. These completed BCS segmentations regulate the hierarchical processing of color and brightness signals by the FCS (Figure K-1). Notable among FCS processes are the extraction of color and brightness signals that are relatively uncontaminated by changes in illumination conditions. These feature contour signals interact within the FCS with the output signals from the BCS to control featural filling-in processes. These filling-in processes lead to visible percepts of color-and-form-in-depth at the final stage of the FCS.

K.3 EMERGENT SEGMENTATION: BOUNDARY COMPLETION, NOISE SUPPRESSION, AND REGULARIZATION

Each of several spatial scales within the BCS contains a hierarchy of orientationally-tuned interactions, which can be divided into two successive subsystems called the Oriented Contrast (OC) Filter and the Cooperative-Competitive (CC) Loop (Figure K-2). The OC Filter contains two successive stages of oriented receptive fields that are sensitive to different properties of image contrasts. The OC Filter generates inputs to the CC Loop, which contains successive stages of spatially short-range competitive interactions and spatially long-range cooperative interactions. Feedback between the competitive and cooperative stages synthesizes a coherent, multiple-scale representation of boundaries, textures, and smoothly shaded image regions.

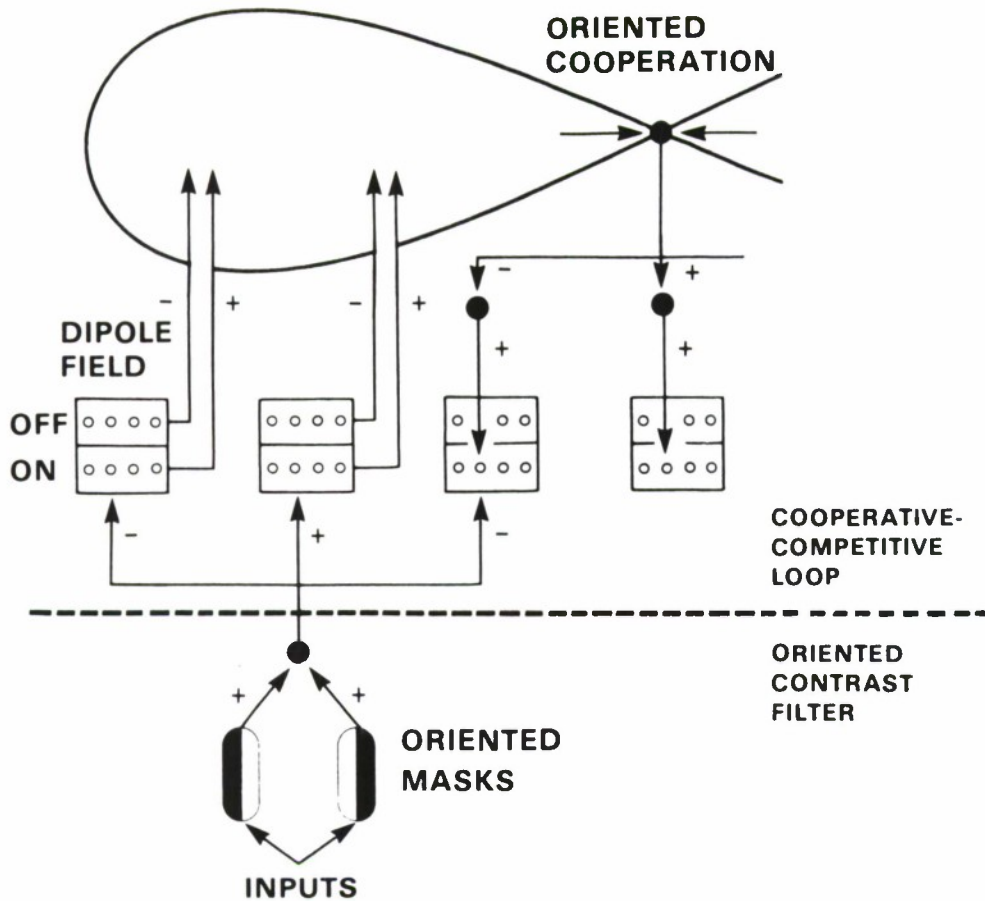


Figure K-2. Architecture of the Boundary Contour System: Inputs activate oriented masks of opposite direction-of-contrast which cooperate at each position and orientation before feeding into an on-center/off-surround interaction. This interaction excites like-orientations at the same position and inhibits like-orientation at nearby positions. The affected cells are on-cells within a dipole field. On-cells at a fixed position compete among orientations. On-cells also inhibit off-cells which represent the same position and orientation. Off-cells at each position, in turn, compete among orientations. Both on-cells and off-cells are tonically active. Net excitation of an on-cell excites a similarly oriented cooperative receptive field at a location corresponding to that of the on-cell. Net excitation of an off-cell inhibits a similarly oriented cooperative receptive field of a bipole cell at a location corresponding to that of the off-cell. Thus, bottom-up excitation of a vertical on-cell, by inhibiting the horizontal on-cell at that position, disinhibits the horizontal off-cell at that position, which in turn inhibits (almost) horizontally-oriented cooperative receptive fields that include its position. Sufficiently strong net positive activation of both receptive fields of a cooperative cell enables it to generate feedback via an on-center/off-surround interaction among like-oriented cells. On-cells which receive the most favorable combination of bottom-up and top-down signals generate the emergent perceptual grouping.

BCS operations occur automatically and without learning or explicit knowledge of input environments. A perceptual process is said to be *pre-attentive* if it occurs rapidly and automatically without recourse to stored templates or expectancies. Thus the emergent segmentations generated by the model are not the result of training on image exemplars. Nor do the equations embody *a priori* assumptions about such variables as direction of illumination or the shapes of objects to be encountered. Instead, the model embodies a number of circuits specialized to perform emergent, context-sensitive segmentations of a wide variety of images.

By *emergent segmentation*, we mean a partition of an image into regions and boundaries that may have no direct corollary in differences in gray level of the image itself. Boundaries perceived in this way are often referred to as “illusory” when seen by humans. (See Figure K-3.) Characteristic of the human

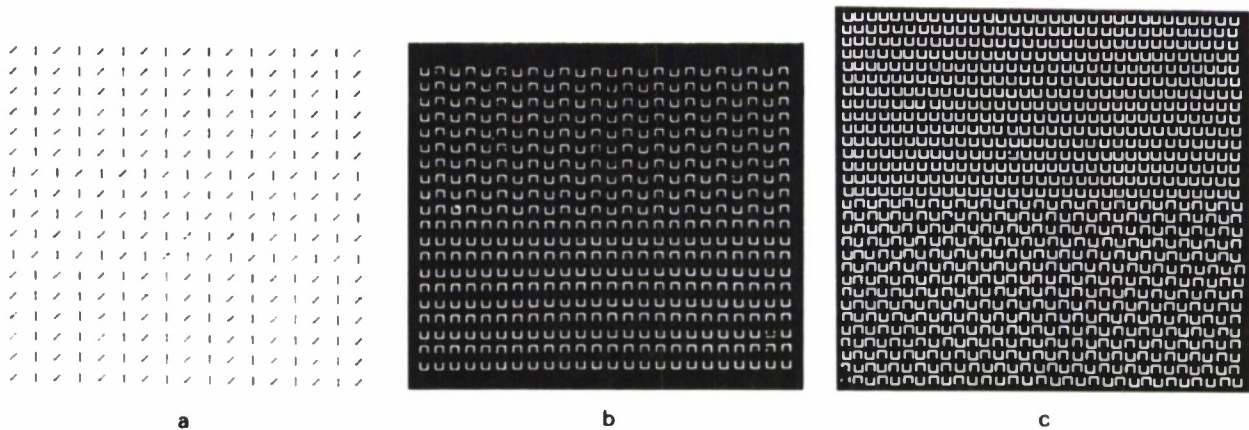


Figure K-3. Emergent segmentation can be (a) co-linear with image contrasts, (b) perpendicular to image contrasts at line ends, or (c) diagonal even though there are no diagonal image elements. Note that the visible featural contrast (light or dark) stays within inducing elements. Reprinted with permission from Beck, Prazdny, and Rosenfield, 1983.

perception of such boundaries is the control of perception of one part of an image by what surrounds that part of the image. The tasks of multiple-scale grouping and regularization are, therefore, approached from the unifying standpoint that every scenic input provides its own context, which the BCS uses to organize local measures. Moreover, the BCS is sufficiently flexible to maintain several potential groupings simultaneously (Figure K-4) and sufficiently rapid, when realized in hardware, to quickly converge on the most favored grouping for a given visual scene.

Regularization refers to the detection of structure at a given scale despite abrupt variations in signal at a small scale. Figure K-5 illustrates the BCS ability to detect and complete sharp boundaries over long distances in the presence of severe noise. The BCS has no external temperature parameter or *a priori* cost function that controls its segmentation process, as in simulated annealing. Instead realtime cooperative-competitive interactions in a feedback loop regulate a rapid convergence to equilibrium. Consequently, the BCS automatically self-calibrates its criteria for grouping and segmentation according to the distribution of signal and noise in a particular image. Thus the relatively gradual ramp transition of noise statistics in

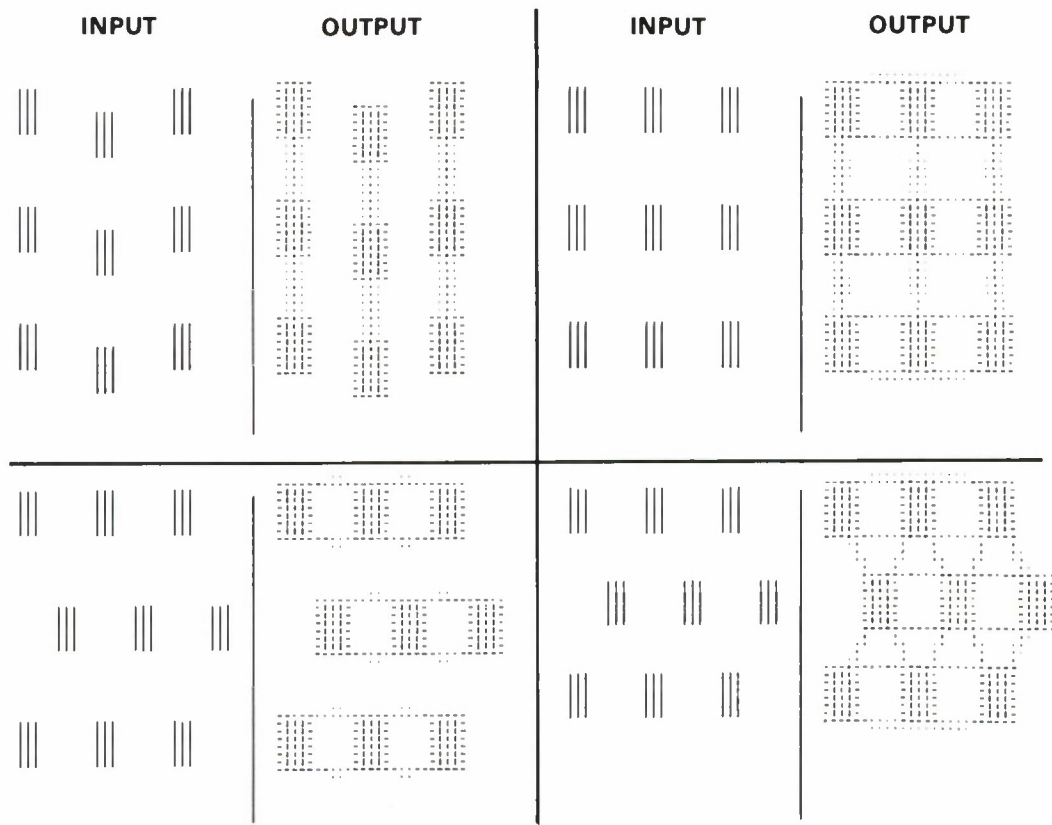


Figure K-4. *Examples of Emergent Groupings Discovered by Using the Boundary Contour System. A computer simulation of emergent groupings that are vertical, vertical-and-horizontal, horizontal, and diagonal in response to input patterns, and which contain corresponding global symmetries within the spatial bandwidths of BCS interactions.*

101882-26

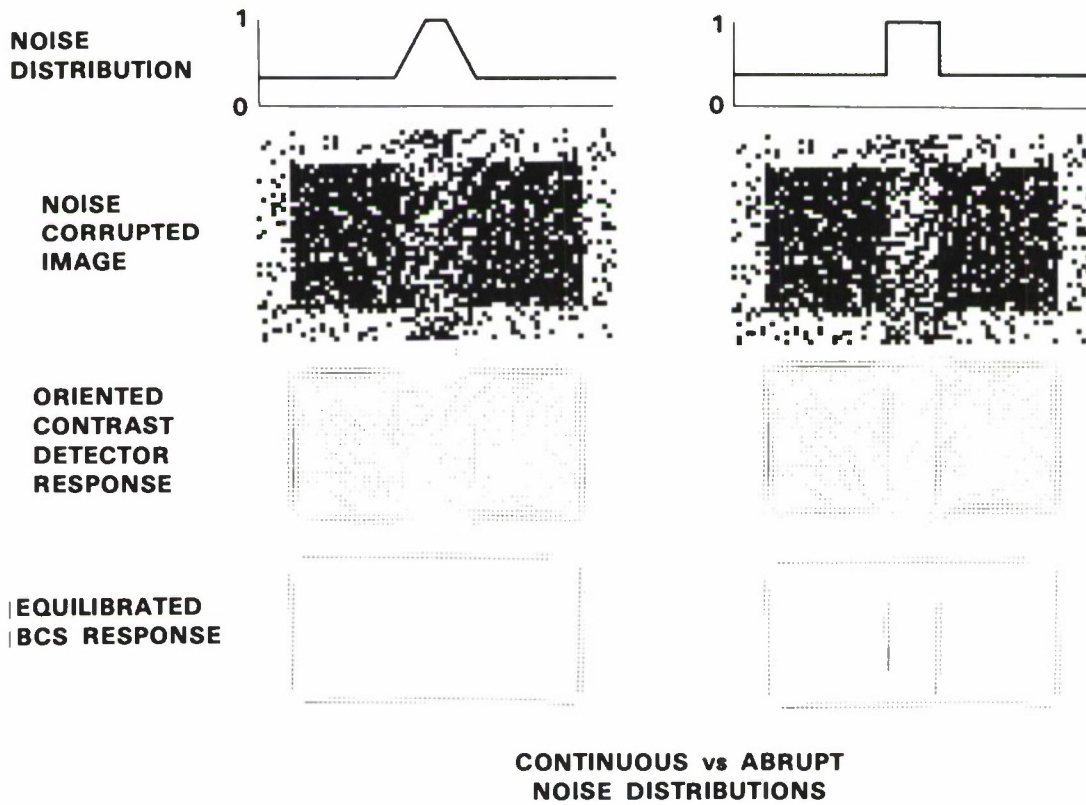


Figure K-5. Image Segmentation in the Presence of Continuously versus Abruptly Changing Noise Distributions. Top Curves – Distribution of noise in horizontal dimension of image. Below – Binary images of a rectangle corrupted by the corresponding noise distribution. Below – Responses of oriented contrast detectors to the noisy image. Bottom – Equilibrated responses of cooperative feedback cells of BCS. For continuous noise (left side), the rectangle is recovered, and the ramped increase of noise in the middle of the image is ignored. For abrupt noise (right side), the rectangle is recovered, and the sharp increase of noise in the middle of the image supports vertical segmentation.

Figure K-5 (left side) is ignored, while a transition that is identical in magnitude but more spatially abrupt invokes a segmentation response in Figure K-5 (right side).

Figure K-6 illustrates the BCS capability for multiple-scale segmentation. Figure K-6 (top) shows a curved textured surface, and Figure K-6 (left) illustrates the responses of oriented contrast detectors to the image (Figure K-6–top). The equilibrated CC Loop outputs (Figure K-6–right) are not simply filterings of contrast at different spatial frequencies, but a detection of the *coherence* of *oriented* contrasts at a given scale. Note that although the inputs are a pattern of discrete elongated texture elements, the

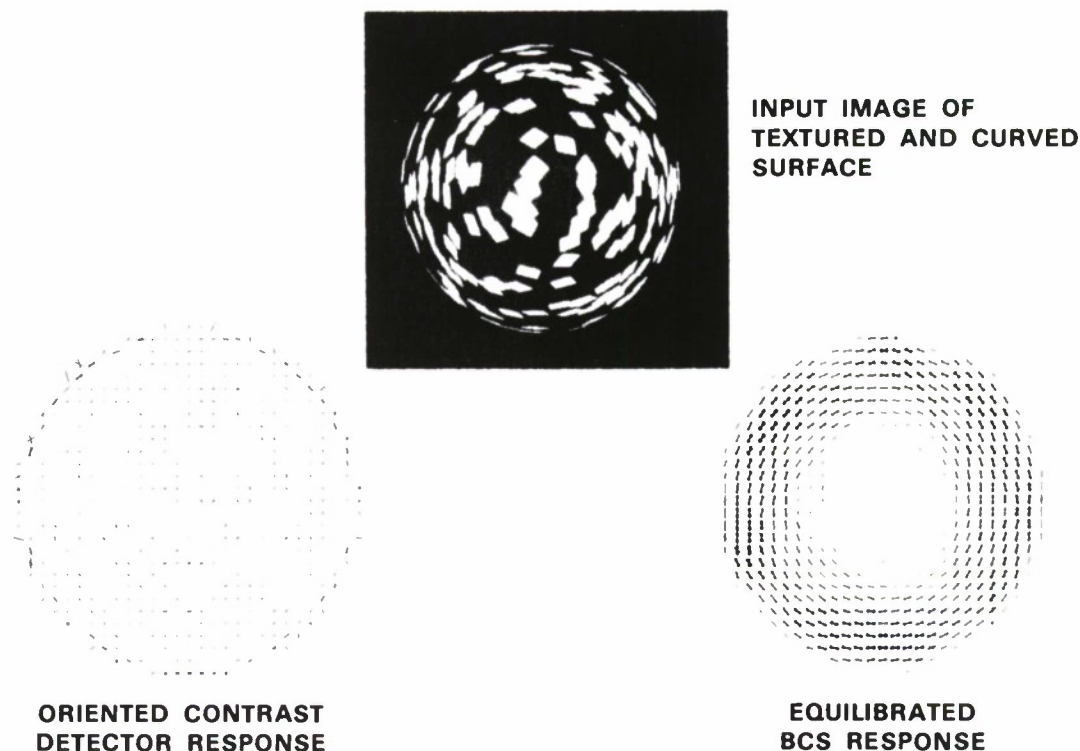


Figure K-6. Multiple-Scale Segmentation. Top: Image of a textured, curved surface [adapted from Todd and Akerstrom (1987)]. Left: Response of oriented contrast detectors of the OC Filter to the image. Right: Equilibrated response of cooperative feedback cells of the CC Loop to the image.

BCS, in response, produces dense webs of activity. These boundary webs in turn support the perception of smoothly curved three-dimensional surface form, as responses from oriented contrast detectors with receptive fields of different sizes are grouped differentially by the CC Loop. The worst correlation between human psychophysical judgments of three-dimensional shape-from-texture and theoretical predictions based upon images such as in Figure K-6 (top) was 0.985.

The macrocircuit depicted in Figure K-1 indicates that a pre-attentively completed segmentation with the BCS can directly activate an Object Recognition System (ORS), whether or not this segmentation supports visible contrast differences within the FCS. The ORS can, in turn, read-out attentive learned

priming, or expectation, signals to the BCS. In response to familiar objects in a scene, the final three-dimensional segmentation with the BCS may thus be *doubly* completed, first by automatic pre-attentive segmentation processes and then by attentive learned expectation processes, as described by adaptive resonance theory [1,2]. This doubly completed segmentation regulates the filling-in processes within the FCS that lead to a percept of visible form.

K.4 INTERACTION OF FEATURE CONTOUR SYSTEM AND BOUNDARY CONTOUR SYSTEM: FUSION OF COLOR-AND-FORM IN FILLED-IN REGIONS

A fundamental issue in visual perception and the design of real-world machine vision systems is how the human visual system transforms the incoming distribution of luminance to generate the perceived brightness distribution. This brightness-from-luminance problem arises because, even in simple two-dimensional displays, brightness does not closely correspond to luminance. The transformation of luminance into brightness is a complex relation, involving compensation for variable illumination levels, and sensitivity for global aspects of the luminance distribution.

Grossberg and Todorović (1987) have developed a theory of two-dimensional brightness perception, which includes a computational characterization of featural filling-in within the FCS (Figure K-7). This theory has been able to simulate, using realistic image inputs, how color-and-form are fused together in filled-in representations of regions. Brightness variations provide information about three-dimensional form, depth relations, surface orientations, and material composition of objects, and thus constitute an essential component of visually-acquired knowledge.

These factors are clearly illustrated by brightness constancy and brightness contrast, two classical brightness phenomena. An example of brightness constancy is the fact that two identical pieces of paper will look about equally bright even if one is well-illuminated and the other is in shadow. Thus surfaces of unequal luminance can have equal brightness, as in Example 2 of Figure K-8 (top right). The visual system “takes into account” the variability of the illumination, and the brightness percept, in this case, correlates with object reflectance (percentage of reflected light) rather than luminance. Therefore, some authors have equated the brightness-from-luminance problem with the reflectance-from-luminance problem [12,13]. However, there are many brightness effects involving surfaces of equal reflectance that look unequally bright. For example, an instance of the phenomenon of brightness contrast is the fact that the same gray piece of paper looks brighter against a black background than against a white background, as in Example 3 of Figure K-8 (bottom left). Thus the appearance of a portion of the visual field depends not only on conditions within that region, but is contextually dependent.

Figure K-9 and Figure K-10 illustrate computer simulations from this theory of two important phenomena: the two-dimensional Craik-O’Brien-Cornsweet effect and a McCann Mondrian in which a brightness contrast is perceived that has not been explained by reflectance-from-luminance theories. The BCS-FCS module thus provides a stand-alone system capable of automatically discounting spurious illumination conditions and filling-in fused representations of color-and-form. Such a preprocessed image representation may be used as a source of input patterns for a subsequent processor, such as ART 2, which is capable of self-organizing a stable pattern recognition code in response to an arbitrary sequence of analog input patterns.

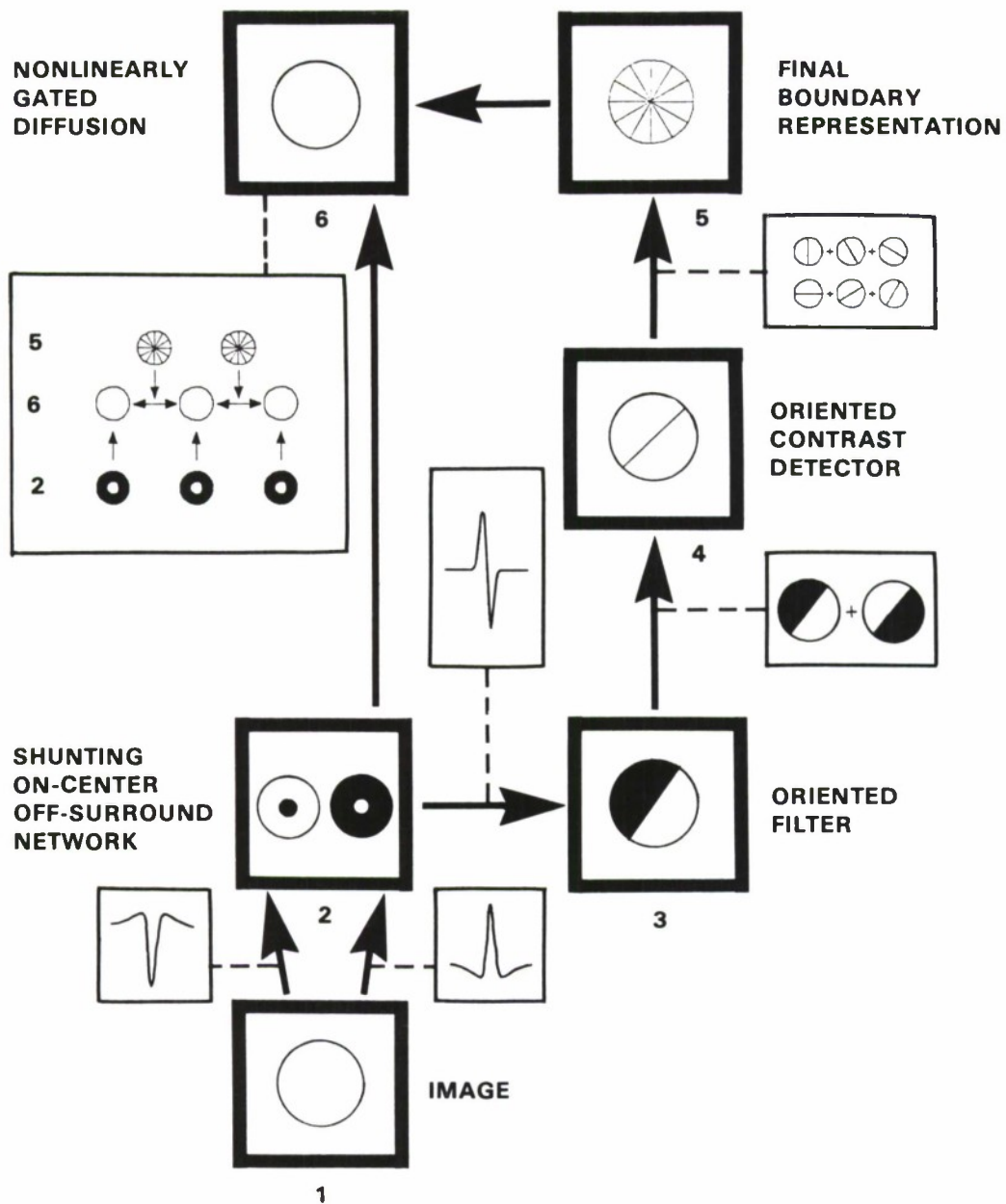


Figure K-7. Flow chart of how the Feature Contour System discounts variable illuminants and regulates featural filling-in: The thick-bordered rectangles numbered from 1 to 6 correspond to the levels of the system. The symbols inside the rectangles are graphical mnemonics for the types of computational units residing at the corresponding model level. The arrows depict the interconnections between the levels. The thin-bordered rectangles represent the type of processing between pairs of levels. This simplified model directly extracts boundaries from image contrasts, rather than generating emergent segmentations from image contrasts. The model's key elements concern how the Level 2 network of shunting on-center/off-surround interactions discounts variable illuminants while extracting feature contour signals, and how Level 5 fills-in signals via a nonlinear diffusion process within the compartments defined by Boundary Contour System output signals.

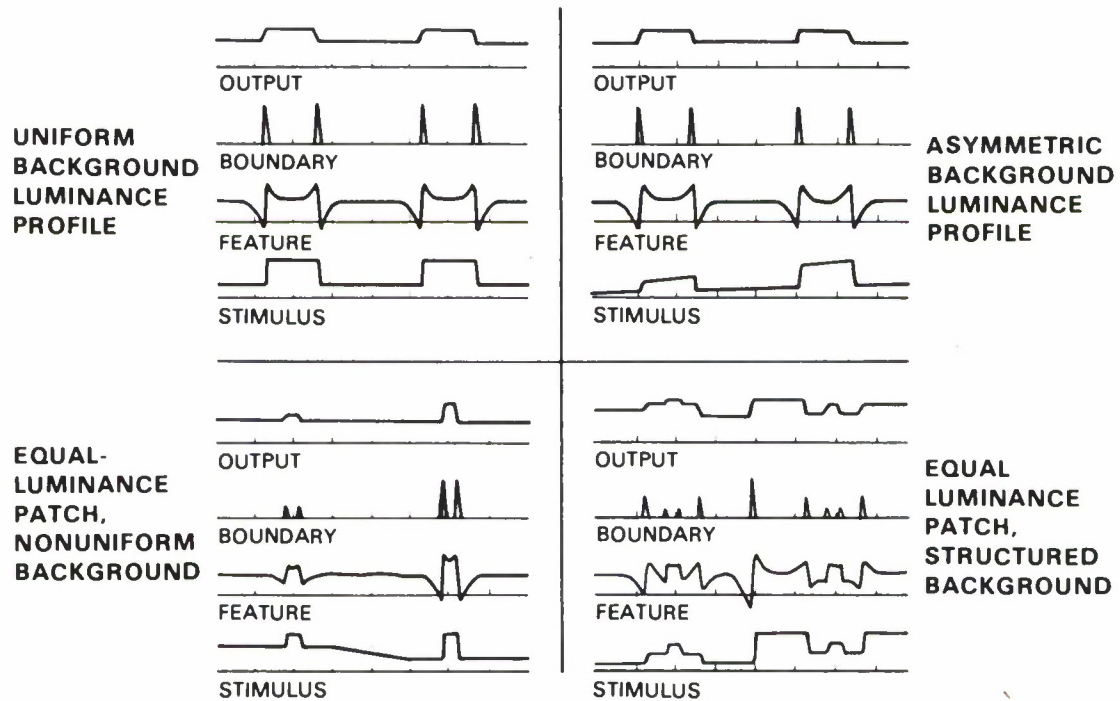


Figure K-8. Boundary and Feature Contour System Interactions Giving Contextually Dependent Brightness Constancy. Simulation of Feature Contour interactions in response to images with a one-dimensional symmetry: The luminance profile (stimulus) in Example 2 (top right) is tilted with respect to Example 1 (top left) due to an asymmetric light source, but the filled-in percept (output) is the same as that in Example 1 (top left), illustrating discounting and constancy. Although the small patches have equal luminance in Example 3 (bottom left), their filled-in percepts are different – in the direction opposite to their backgrounds – illustrating contrast. Although the small inner patches have equal luminance in Example 4 (bottom right), the filled-in percept of the right patch is darker than that of the left patch – in the direction of their surrounding patches – illustrating assimilation.

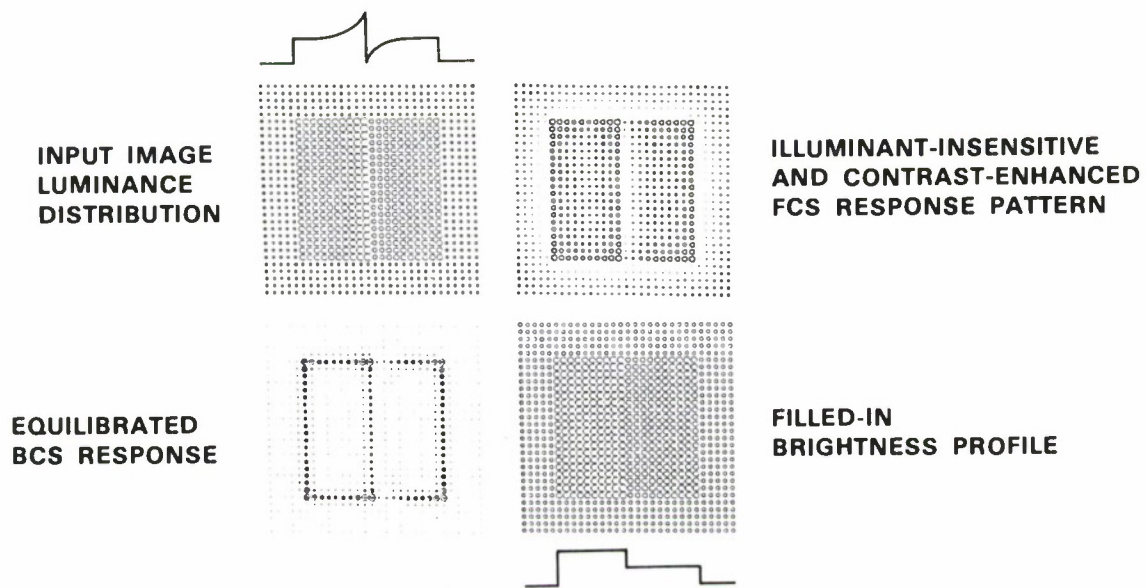


Figure K-9. Simulation of the Craik-O'Brien-Cornsweet Effect. The size of the symbols codes the activity level of units at corresponding locations at different network levels. Top Left: The luminance distribution. Top Right: A contrast-enhance FCS pattern which is insensitive to illuminant variations. Bottom Left: The boundary segmentation within the BCS. Bottom Right: The filled-in brightness profile at a higher level of the FCS.

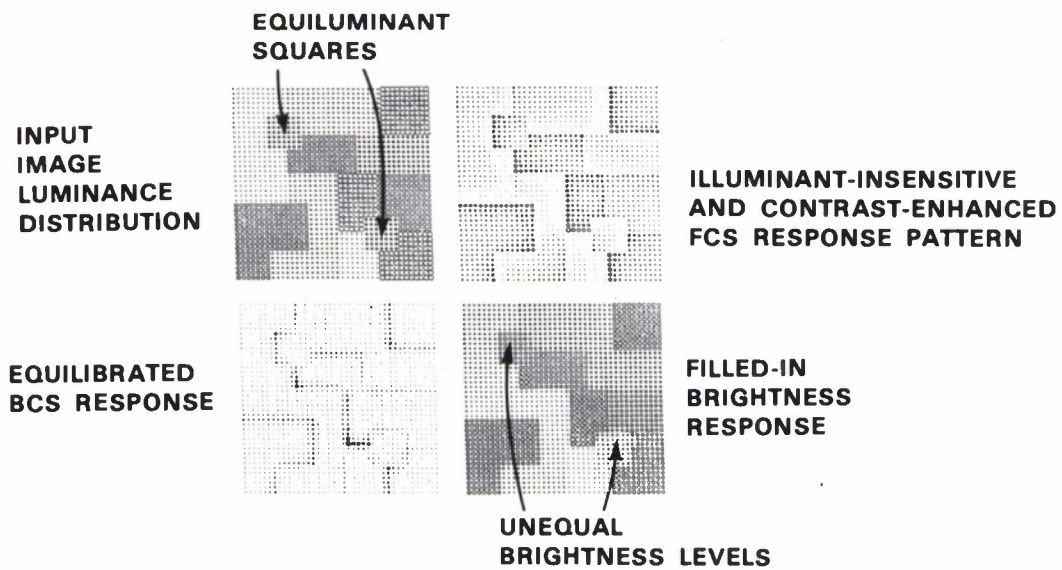


Figure K-10. Simulation of a Mondrian Image. The depicted network levels are as in Figure K-9. Note that the equally luminant square patches on the major diagonal (top left) do not produce equal perceived brightness (bottom right).

REFERENCES

1. Carpenter, G.A. and Grossberg, S., "A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine," *Computer Vision, Graphics, and Image Processing*, **37**, pp. 54-115, 1987a.
2. Carpenter, G.A. and Grossberg, S., "ART 2: Self-Organization of Stable Category Recognition Codes for Analog Input Patterns," *Applied Optics*, **26**, pp. 4919-4930, 1987b.
3. Cohen, M.A. and Grossberg, S., "Neural Dynamics of Brightness Perception: Features, Boundaries, Diffusion, and Resonance," *Perception and Psychophysics*, **36**, pp. 428-456, 1984.
4. Grossberg, S., "Cortical Dynamics of Three-Dimensional Form, Color, and Brightness Perception, I: Monocular Theory," *Perception and Psychophysics*, **41**, pp. 87-116, 1987a.
5. Grossberg, S., "Cortical Dynamics of Three-Dimensional Form, Color, and Brightness Perception, II: Binocular Theory," *Perception and Psychophysics*, **41**, pp. 117-158, 1987b.
6. Grossberg, S. and Marshall, J., "A Computational Model of How Cortical Complex Cells Multiplex Information about Position, Contrast, Orientation, Spatial Frequency, and Disparity," In *Proceedings of the First International Conference on Neural Networks*, San Diego June 21-24, **IV**, pp. 203-213, 1987.
7. Grossberg, S. and Mingolla, E., "Neural Dynamics of Form Perception: Boundary Completion, Illusory Figures, and Neon Color Spreading," *Psychological Review*, **92**, pp. 173-211, 1985a.
8. Grossberg, S. and Mingolla, E., "Neural Dynamics of Perceptual Grouping: Textures, Boundaries, and Emergent Segmentations," *Perception and Psychophysics*, **38**, pp. 141-171, 1985b.
9. Grossberg, S. and Mingolla, E., "Computer Simulation of Neural Networks for Perceptual Psychology," *Behavior Research Methods, Instruments, and Computers*, **18**, pp. 601-607, 1986.
10. Grossberg, S. and Mingolla, E., "Neural Dynamics of Surface Perception: Boundary Webs, Illuminants, and Shape-from-Shading," *Computer Vision, Graphics, and Image Processing*, **37**, pp. 116-165, 1987.
11. Grossberg, S. and Todorović, D., "Neural Dynamics of 1-D and 2-D Brightness Perception: A Unified Model of Classical and Recent Phenomena," *Perception and Psychophysics*, in press, 1987.
12. Horn, B. K. P., "Determining Lightness from an Image," *Computer Graphics and Image Processing*, **3**, pp. 277-299, 1974.
13. Hurlbert, A., "Formal Connections Between Lightness Algorithms," *Journal of the Optical Society of America*, **A3**, pp. 1684-1693, 1986.

APPENDIX L
PRESENTATIONS TO THE APPLICATIONS PANEL

L.1 LIST OF CONTRIBUTORS

VISION

- VIS-1** AC-coupled retina with cooperative receptors
Michael H. Brill, SAIC, Falls Church VA 703-538-3710

- VIS-2** Solution of optimization problems in object-level vision
Richard Elsley, Rockwell International Science Center, Thousand Oaks CA
805-373-4155

- VIS-3** Application of neural networks to target classification
Lynn E. Garn, Night Vision Lab, Fort Belvoir VA 703-664-6066

- VIS-4** Multidimensional image fusion and segmentation
Steven Grossberg, Ennio Mingolla, Boston U., Boston MA 617-353-7857

- VIS-5** Processing of laser radar data with neural networks
Matthew Kabrisky, Steven Rogers, Air Force Institute of Technology, Dayton OH
513-255-5276

- VIS-6** Target classification using a self-organizing neural network
Murali Menon, Lincoln Laboratory, Lexington MA 617-981-5374

- VIS-7** Adaptive edge detection for Automatic Target Recognition (ATR) application
Michael Oyster, Hughes, El Segundo CA

- VIS-8** Multiple sensor fusion for discrimination
Leonid I. Perlovsky, Nichols Research Corporation, Wakefield MA

- VIS-9** Application and implementation of dynamical adaptive systems
Fernando Pineda, Applied Physics Laboratory, Laurel MD 301-792-5000x7260

- VIS-10** Segmentation of textured images using Wigner transforms
T. Reed, U. of Minnesota, MN 612-625-1316

- VIS-11** Industrial parts inspection
D. L. Reilly, Nestor, Providence RI 401-331-9640

VISION - continued

- VIS-12** Japanese (Kanji) character recognition
D. L. Reilly, Nestor, Providence RI 401-331-9640
- VIS-13** Online character recognition
D. L. Reilly, Nestor, Providence RI 401-331-9640
- VIS-14** Realtime 3-d object classification
D. L. Reilly, Nestor, Providence RI 401-331-9640
- VIS-15** OCR recognition of unconstrained hand-drawn numerals
C. L. Scofield, Nestor, Providence RI 401-331-9640
- VIS-16** Computing shape from shading with neural networks
Terry Sejnowski, Johns Hopkins University MD 301-338-8687
- VIS-17** DARPA ATR optical processor
Harold M. Stoll, Northrop, Palos Verdes Peninsula CA 213-377-4811
- VIS-18** Neocognitron evaluation
William Stoner, SAIC, Bedford MA 617-275-2200
- VIS-19** Machine vision
H. Taichi Wang, Bimal P. Mathur, Rockwell International Science Center,
Thousand Oaks CA 805-373-4192
- VIS-20** Distributed processing for invariant recognition and data fusion
Harry Wechsler, U. of Minnesota, MN 612-625-1316

SPEECH

- SPE-1** Learning to recognize speech with neural networks
Jeffrey Elman, UCSD, La Jolla CA 619-534-1147
- SPE-2** Speaker-dependent isolated-word speech recognition (e.g. Intel iSBC 576)
Marcian E. Hoff, Jr., Consultant, Los Altos Hills CA 415-941-3331
- SPE-3** Speech recognition
D. L. Reilly, Nestor, Providence RI 401-331-9640
- SPE-4** Mapping visemes to phonemes for visual speech recognition
Ben Yuhaz, Johns Hopkins University, MD

SONAR

- SON-1** Learned classification of sonar targets using a massively parallel network
Paul Gorman, Allied Signal, Columbia MD 301-964-4094
- SON-2** Sonar target classification with a neural network
J. Harold McBeth, General Dynamics, San Diego CA 619-573-7857

RADAR

- RAD-1** Adaptive radar processing
Dean Collins, Texas Instruments, Dallas TX 214-995-3974
- RAD-2** Satellite orbital motion detector from RCS data
Mitch Eggers, MIT Lincoln Laboratory, Lexington MA 617-981-2664
- RAD-3** Radar target recognition from partial information based on models of neural network
Nabil Farhat, U. of Pennsylvania, Philadelphia PA 215-898-5882
- RAD-4** Multi-target tracking
Robert Kuczewski, TRW, San Diego CA 619-592-3381
- RAD-5** Neural network radar signal processor for IFF
Fred Weingard, Booz-Allen, Washington, DC 703-769-7782

SIGNAL PROCESSING

- SIG-1** Satellite classification
Steven Lehar, TEXTRON Defense Systems, MA 617-381-4799
- SIG-2** Removing random noise from EKG signals
Douglas Palmer, HNC, San Diego CA 619-546-8877

ROBOTICS

- ROB-1** CMAC: the Cerebellar Model Arithmetic Computer—a neural network robot control system
James S. Albus, National Bureau of Standards, Gaithersburg MD 301-975-3418

ROBOTICS - continued

- ROB-2** Control systems that learn
Richard Elsley, Rockwell International Science Center, Thousand Oaks CA
805-373-4155
- ROB-3** Parallel distributed control of robotic systems
Jack Gelfand, David Sarnoff Research Center, SRI, Princeton NJ 609-734-3098
- ROB-4** Tool-use in biological and artificial systems
Neville Hogan, MIT, Cambridge MA 617-253-2277
- ROB-5** Trainable and adaptable neural networks for robot control
Von A. Jennings, Martin Marietta, Baltimore MD 301-682-0892
- ROB-6** Robot Navigation
Chuck C. Jorgensen, Thomson CSF-pro, Palo Alto CA 415-494-8818
- ROB-7** Neural dynamics of adaptive sensory motor coordination
Michael Kuperstein, Wellesley College and Neurogen, Wellesley MA 617-739-2215
- ROB-8** Teaching an adaptive network with visual inputs to balance an inverted pendulum
Viral V. Tolat, Bernard Widrow, Stanford U., Stanford CA 415-723-4949

DYNAMICAL SYSTEMS

- DYN-1** GTE process monitor
Richard Sutton, GTE Laboratories, Waltham MA 617-466-4133

BIOLOGY

- BIO-1** Motion detection in fly and machine vision
Heinrich Buelthoff, MIT, Cambridge MA 617-253-0549
- BIO-2** Central pattern generator for locomotion in Lamprey
Avis Cohen, Cornell University, Ithaca NY 607-255-8997
- BIO-3** Study of neuronal networks artificially grown on microelectronic probe substrates
Mitch Eggers, MIT Lincoln Laboratory, Lexington MA 617-981-2664

BIOLOGY - continued

- BIO-4** Investigating mammalian neural network modeling parameters
Guenter Gross, North Texas State University, Denton TX 817-565-3615
- BIO-5** Motion measurement in the vertebrate retina
Norberto Grzywacz, MIT, Cambridge MA 617-253-0545
- BIO-6** Self-organizing computational maps
John Pearson, David Sarnoff Research Center, SRI, Princeton NJ 609-734-2385
- BIO-7** DARWIN-III: a selective recognition automaton
Gerald M. Edelman, George N. Reeke, Jr., The Neurosciences Institute,
New York NY 212-570-7627
- BIO-8** Spatial-temporal coding in dynamic models of brain function and
potential applications
Gordon Shaw, UC Irvine, Irvine CA 714-856-6620

DECISION SYSTEMS

- DEC-1** Optimal allocation of active sensors
George Fusik, TRW, Redondo Beach CA 213-535-0350
- DEC-2** Mortgage delinquency prediction
C. L. Scofield, Nestor, Providence RI 401-331-9640
- DEC-3** Mortgage loan evaluator
C. L. Scofield, Nestor, Providence RI 401-331-9640

THEORY

- THE-1** Practical results from NN theory
Yaser Abu-Mostafa, CalTech, Pasadena CA 818-356-4842
- THE-2** Phase-locking of neuronal circuits
Frank C. Hoppensteadt, Michigan State U., East Lansing MI 517-355-4473
- THE-3** Random code neural networks
Alexander Jourjine, Analog Intelligence Corp, Cambridge MA 617-524-4858

THEORY - continued

- THE-4** Image and object coding for data compression
Albert F. Lawrence, Hughes Aircraft, CA
- THE-5** Generalities about neural networks
Robert Rosen, Dalhousie U., Halifax NS 716-473-4055

SURVEY

- SUR-1** Neural network Overview on the ESPRIT-II Program
Bernard Angeniol, Thomson CSF, France
- SUR-2** SDI discrimination problem
Mike Cain, Booz-Allen, Washington, DC 703-769-7782
- SUR-3** Neural network applications at Hughes Ground Systems Group
Patrick F. Castelaz, Hughes Command and Control Division, Fullerton CA
714-732-8622
- SUR-4** Modern data storage and retrieval systems
Ching-chih Chen, Simmons College, Boston MA 617-738-2224
- SUR-5** Venture capital financing of a neural network company
Oliver D. Curme, Battery Ventures, L.P., Boston MA 617-367-1011
- SUR-6** Defense applications of neuro computers
Robert Hecht-Nielsen, HNC, San Diego CA 619-546-8877
- SUR-7** Laser radar data survey
Jim Leonard, WPAFB, Dayton, OH
- SUR-8** Future command and control systems
Orin E. Marvel, Hughes Aircraft, Fullerton CA 714-732-5869
- SUR-9** Custom analog VLSI chip development for vision and speech
Carver Mead, CalTech, Pasadena CA 818-356-6568
- SUR-10** Neural network applications at TRW
Mike Myers, Robert Kuczewski, TRW, San Diego CA 619-592-3381
- SUR-11** Artificial adaptive neural network systems
Tom Ryan, SAIC, Tucson AZ

SURVEY - continued

- SUR-12** Cellular automata as neural networks: pattern recognition and tracking
Tommaso Toffoli, MIT, Cambridge MA 617-253-3194
- SUR-13** Applications of neural networks: review of Honeywell's activities
Jim White, Honeywell Systems-Research Center, Minneapolis MN 612-782-7355

L.2 ONE PAGE SUMMARIES OF CONTRIBUTIONS

VIS-1: AC-coupled Retina with Cooperative Receptors

Investigator: Michael H. Brill, SAIC, Falls Church VA 703/538-3710

Problem Area: Develop a model for the photoreceptor dynamics in the retina which is photon-noise resistant and results in a stable percept independent of illumination and suitable for machine vision.

Prior Approach: N/A

Neural Network Approach: Cooperative receptors with three-state activation kinetics using short-lived intermediate photopigment states. The model is "AC-coupled" in the sense that only temporally varying light/dark gradients maintain a sensory excitation. Response of the receptor cells is coded as voltage, and the receptors are resistively coupled.

Status: Completed model simulation for black/white imagery.

Results: The model permits an automatic mixture of time/space integration:

- a) for high photon flux each receptor cell responds rapidly, and the neighboring cells are decoupled, providing the highest spatiotemporal resolution;
- b) for low photon flux cell temporal response is slow, and neighboring cells are coupled to give broader spatiotemporal integration, enhancing light sensitivity at the expense of resolution in space and time.

Future Plans: Apply the key ideas of this model to color perception, and investigate implementation in silicon.

Comments: The model has a strong physiological plausability; specifically, the assumed reaction kinetics is similar to the one proposed for self-organizing computational maps (see John Pearson's contribution: BIO-6).

Conclusions: A photoreceptor model is proposed that accounts for two important biologically observed features:

- Only temporarily varying visual information is encoded;
- Light-flux dependent coupling of time/space integration is demonstrated.

VIS-2: Solution of Optimization Problems in Object-Level Vision

Investigator: Dr. Richard Elsley, Rockwell International Science Center, Thousand Oaks CA
805/373-4155

Problem Area: The analysis of objects seen by imaging systems often involves matching problems which are combinatorially unmanageable. An example is the tracking of targets through a series of frames of imagery data.

Prior Approach: Serial methods, such as the window method which explicitly look for successive blips in windows predicted by previous data.

Neural Network Approach: Treat as optimization problems and solve with special-purpose Hopfield networks.

Status: Simulations of target tracking and several related problems working. Conceptual design of special purpose chip done.

Results: Tracking of 40 simultaneous targets simulated. Deals with missing and extra data, track crossings, off-screen data. Problem segmentation is used for large numbers of targets.

Future Plans:

- Fuse multi-sensor data;
- Integrate into target classification system;
- Implement custom network architecture in hardware;
- Apply method to other problems.

VIS-3: Application of Neural Network to Target Classification

Investigator: Lynn E. Gam, Night Vision Lab, Fort Belvoir VA 703/664-6066

Problem Area: Target classification with sparse and ambiguous data is a common problem, especially with infrared sensors.

Prior Approach: Shape-dependent parameter classification is difficult to implement with such data.

Neural Network Approach: Developed an analytic simulation model with two features and three classes; trained a fully-connected network (two input, two hidden, and three output units) with 50 examples using backpropagation and tested it and a K-neural network classifier with 10 000 examples.

Status: On-going simulation efforts.

Results: For these simulations, the neural network achieved a correct response at 4.5% below theoretically optimal; the K-neural network with K=1 and K=3 achieved 18.4% and 11.1%, respectively.

Future Plans: Study the scaling properties implied by an extended image classification problem.

Conclusions: A neural network can extract a near-optimal decision space from sparse and ambiguous training data without explicit knowledge of the underlying statistics.

VIS-4: Multi-dimensional Image Fusion and Segmentation

Investigators: Steven Grossberg, Ennio Mingolla, Boston U., Boston MA 617/353-7857

Problem Area: Combine different types of locally ambiguous visual information to rapidly generate a globally consistent and unambiguous representation of color-and-form-in-depth. This work is useful, for example, for processing multi-dimensional image data from laser radar sensors.

Prior Approach: AI algorithms for machine vision.

Neural Network Approach: Emergent segmentation and featural filling-in are carried out by nonlinear interactions between a pair of parallel systems called the boundary contour system (BCS) and the feature contour system (FCS). The BCS controls the emergence of a fused three-dimensional segmentation of a scene, while the FCS extracts color and brightness signals that are relatively invariant to changes in illumination conditions and performs featural filling-in of a scene. The neural network architecture possesses a regular structure suitable for compact hardware implementation.

Status: Simulations on a variety of different artificial image data have been completed.

Results: Automatic segmentation, completion, regularization, noise suppression, and filling-in have been demonstrated with this system.

Future Plans: Couple the image representation resulting from the BCS/FCS interaction with a stable, self-organizing recognition code, such as ART II.

Conclusions: A theory of two-dimensional brightness perception has been able to simulate, using varied and realistic image inputs, how color- and-form are automatically fused together in filled-in representations of regions.

VIS-5: Processing of Laser Radar Data with Neural Networks

Investigators: Matthew Kabrisky, Steven Rogers, Air Force Institute of Technology, Dayton OH 513/255-5276

Problem Area: Performance comparison of neural network and conventional approaches for target recognition of an existing actual laser radar database. All data are preprocessed by a Zernicke transformation to achieve scale and rotation invariance resulting in a 22-dimensional input vector for each scene.

Prior Approach: Standard nearest-neighbor classifier.

Neural Network Approach: A four-layer, fully-connected network with (22, 200, 60, four) nodes per layer; about 200 training scenes were presented about $25\,000\times$ using backpropagation learning; the remaining 40 scenes represented the test sample.

Status: Finished research project.

Results: Within the noise of the data, the conventional and neural network approaches performed equally well; certain test classes with few examples had rather high error rates.

Future Plans: Repeat comparison with a more extensive, higher-quality database.

Comments: Since the network has a large number of hidden weights and a rather small training set, it is possible that the neural network stored the actual training set explicitly (this would have to be tested with a more extensive database).

Conclusions: Preprocessed laser radar data can be classified with a neural network; at the present stage, a clearcut advantage of the neural network approach is not apparent.

VIS-6: Target Classification Using a Self-Organizing Neural Network

Investigator: Murali Menon, M. I. T. Lincoln Laboratory, Lexington MA 617/981-5374

Problem Area: Classification of preprocessed patterns (wire diagrams).

Prior Approach: Rule-based feature classifier, for example.

Neural Network Approach: Fukushima's neocognitron with four levels, eight layers.

Status: Completed demonstration simulations.

Results: Classified three different wire models demonstrating shift tolerance and orientational invariance; for illustrative purposes, the self-selected features of the intermediate layers were also studied. Input data were 128×128 pixels; classification still worked when 35% of pixels were corrupted by random noise.

Future Plans: Apply the model to actual, preprocessed laser radar data.

Comments: The model is computationally intensive; current simulations are run on a VAX/8600: training of 164 000 nodes with 32 000 weights took four hours of CPU time; for real-time applications, special-purpose hardware would be required.

Conclusions: Once the system parameters have been properly selected, the system can robustly classify different input classes – no re-tuning of the parameters is required. The neocognitron is an example of an unsupervised, hierarchical feature detector.

VIS-7: Adaptive Edge Detection for Automatic Target Recognition (ATR) Application

Investigator: Michael Oyster, Hughes, El Segundo CA 213/607-0838

Problem Area: Find low-level features in FLIR images with high-contrast ratio and low signal/noise.

Prior Approach: Fixed-threshold pixel processing; some features may be lost in noise.

Neural Network Approach: A three-layered network with fixed weights; the final layer executes a Grossberg-type relaxation with a threshold value that is locally adaptable.

Status: A conceptual example was discussed.

Comments: This presentation attempted to give the Neural Network Study's System Applications Panel a unique example of a visual processing function which could only be performed using a neural network approach.

Conclusions: ATRs are a very significant defense application. It is easy to define examples where neural network algorithms will dramatically enhance the performance of state-of-the-art ATR systems. The crucial issues are: which neural network algorithms will produce the most improvement and what is the best R&D route for achieving the improved performance. The issue is not whether neural networks will improve ATR performance.

VIS-8: Multiple Sensor Fusion for Discrimination

Investigator: Leonid I. Perlovsky, Nichols Research Corporation, Wakefield MA

Problem Area: Discriminate RVs among decoys using multiple sensors and classification information from kill assessment.

Prior Approach: AI and statistical pattern recognition techniques.

Neural Network Approach: Combined unsupervised/supervised learning classifier network using maximum likelihood approach for information fusion and parameter estimation. Neural network has no hidden layers and uses adaptively adjusted nonlinear decision regions for flexible classifier boundary construction. Two-layer neural network weights are fuzzy classification variables equivalent to estimated Bayesian probabilities.

Status: Conceptual stage of development; 12 man-months expended to date.

Results: Examples of classifier training in simulated scenarios have been executed.

Future Plans: Optimization of number of neurons and connections for various problems need to be studied with limited training data, variable number of classes and types, and increased dimensionality of classification space. At present, the classifier retains all prior training information; the capability for "forgetting" individual patterns may be a necessary development.

Estimated computational requirements for SDI applications:

Conventional processor (10 MIPS for 1 500 seconds) could calculate the classification parameters on 10^3 targets of 10 types using three discrimination features (equivalent to 10^5 neural network connections; 10^5 targets of 100 types using 10 discrimination features would require 10^7 MIPS or 10^8 neural network connections.

Comments: A unique method of neural network weight estimation is claimed using nonlinear hyper-boundaries in feature space. To the degree that this method results in nonlinear boundaries, there is a conceptual relationship to the Nestor RCE network.

Conclusions: Noteworthy, yet preliminary work which has still to focus on some of the important details of implementation in high-dimensional spaces.

VIS-9: Application and Implementation of Dynamical Adaptive Systems

Investigator: Fernando Pineda, Applied Physics Laboratory, Laurel MD 301/792-5000 x7260

Problem Area: Find simple and stable algorithms for neural network models which are well suited for implementation as physical systems – e.g., analog VLSI or optical hardware.

Prior Approach: Standard backpropagation based on finite difference equations is suited only to implementation on digital machines.

Neural Network Approach: Differential-equation-based learning rule which reduces to conventional backpropagation in the limits of discrete time and feedforward connectivity. DE approach amenable to implementation in analog VLSI.

Status: On-going research funded by JHU/APL and AFOSR.

Results: Obtained a general method for deriving differential-equation-based learning rules for a general class of neural network models. Applied one of the simplest algorithms to train a two-layer associative memory. Demonstrated a robust pattern recognition module composed of a four-layer network with feedback connections between the second, third, and output layers. The first two layers act as an associative memory and account for the robustness of the system. Limited shift and rotational invariance as well as noise suppression capabilities could be demonstrated for three different pictorial inputs of 55×24 pixels.

Future Plans: Dr. Pineda believes strongly that neural network algorithms are ill-suited for digital machines and that optimized conventional algorithms will outperform neural networks running on digital machines if the task is a naturally symbolic manipulation task – e.g., traditional AI, NETtalk, etc. Signal processing tasks are not symbolic, so he expects a significant impact here. Also, some AI tasks might be better accomplished by not going to a symbolic representation (see, e.g., the presentation by Ben Yuhaz [SPE-4, p. 154]), in which case neural network techniques can be expected to have significant impact.

Comments: Development of massively parallel analog neural machines is essential. (In collaboration with Andreas Andreou, JHU, and Robert Jenkins, APL).

Conclusions: “Conventional” backpropagation training on a digital computer does not allow realistic time estimates for learning on analog chips, which will be orders of magnitude faster.

VIS-10: Segmentation of Textured Images Using Wigner Transforms

Investigators: T. Reed, Harry Wechsler, U. of Minnesota, MN 612/625-1316

Problem Area: Unsupervised analysis of pictures with random structures – i.e., texture analysis using minimal *a priori* information.

Approach: Use of the Wigner transform for spatial/spatial-frequency representations.

Status: Ongoing research.

Results: Simple test patterns could be segmented on 64×64 pixel fields; a Cray is recommended for faster turnaround times.

Future Plans: The proposed method is sensitive to non-uniform lighting, and segmentation of slanted or curved surfaces needs to be demonstrated.

Comments: For each pixel, this is a strictly serial processing approach, no feedback is required.

Conclusions: A Wigner distribution has been examined as a tool for clustering and grouping.

VIS-11: Industrial Parts Inspection

Investigator: D. L. Reilly, Nestor, Providence RI 401/331-9640

Problem Area: Industrial processes; pattern recognition.

Prior Approach: Human operators.

Neural Network Approach: The Nestor Learning System was trained by showing it where to look for distinctive differences between parts (but not what to look for) in the image.

Status: Completed.

Results: Training and tests on three different types of automatic transmission stators yielded an accuracy of 97%.

Future Plans: This work will be extended to a wide range of industrial parts.

Conclusions: The Nestor Learning System is able to detect subtle distinctions among otherwise similar parts (automatic transmission stators).

VIS-12: Japanese (Kanji) Character Recognition

Investigator: D. L. Reilly, Nestor, Providence RI 401/331-9640

Problem Area: Pattern recognition, classification.

Prior Approach: Human readers.

Neural Network Approach: Use of the Nestor Learning System to train on a large data set to recognize approximately 2 500 individual characters.

Status: Completed.

Results: Recognition of 2 500 distinct characters (classes) on a test set of previously unseen examples, with a base accuracy of 92% for Kanji and Hiragana, 95% on Katakana, and 97% after additional, user- specific training on problem characters.

Conclusions: The Nestor Learning System has been successfully trained to recognize patterns in a set of approximately 2 500 distinct classes.

VIS-13: Online Character Recognition

Investigators: D. L. Reilly, D. Ward, Nestor, Providence RI 401/331-9640

Problem Area: Pattern recognition, classification.

Neural Network Approach: Use of the Nestor Learning System to train a neural network in the recognition and classification of unconstrained, handwritten letters, or other symbols.

Status: Completed.

Results: Accuracies of 98% are obtained on test sets of previously unseen characters.

VIS-14: Realtime Three-dimensional Object Classification

Investigator: D. L. Reilly, Nestor, Providence RI 401/331-9640

Problem Area: Object recognition, target recognition, quality inspection.

Prior Approach: Rule-based systems.

Neural Network Approach: Use of the Nestor Learning System to train neural networks on randomly selected images (100 per object).

Status: Completed.

Results: 88% accuracy in forced recognition with 100% throughput; better than 98% recognition for 70% throughput (30% not classified, 2% error).

Future Plans: Extension of this work is planned to more complex environments.

Conclusions: Three-dimensional object recognition, using the Nestor Learning System, has been established.

VIS-15: OCR Recognition of Unconstrained Hand-drawn Numerals

Investigator: C. L. Scofield, Nestor, Providence RI 401/331-9640

Problem Area: Pattern recognition, classification.

Prior Approach: Human operators.

Neural Network Approach: Use of the Nestor Learning System to train a neural network on an existing database (from a European postal service).

Status: Completed.

Results: 97.5% accuracy on previously unseen test sets.

Conclusions: The Nestor Learning System can be used in a variety of applications that involve recognition of numbers on documents.

VIS-16: Computing Shape from Shading with Neural Networks

Investigator: Terry Sejnowski, Johns Hopkins University MD 301/338-8687

Problem Area: Find the three-dimensional shape from two-dimensional shading information in images.

Prior Approach: Explicitly, this is an ill-posed problem.

Neural Network Approach: Use a three-layer feedforward network with 122 input units, 36 intermediate, and 24 output units; network is trained by backpropagation; representation is the principal curvatures; raw data are preprocessed with a Mexican-hat-type receptive field with considerable overlap; the output layer represents a distributed coding of the curvatures.

Status: Demonstration simulations completed.

Results: Cells in the hidden layer organize as edge detectors.

Comments: DARPA should coordinate a better infrastructure for neural network research. The development of standard neural network databases would be very useful for neural network research.

Conclusions: A small neural network by itself does not intrinsically learn; the problem and the network need to be structured properly to permit a solution. However, neural networks are not just a look-up table; they can find features in data with unknown statistics and can 'generalize' by nonlinear interpolation. Neural networks, so far, treat only a small piece of the problem of how to intelligently design learning systems.

VIS-17: DARPA ATR Optical Processor

Investigator: Harold M. Stoll, Northrop, Palos Verdes Peninsula CA 213/377-4811

Problem Area: Build a high-throughput, fully optical processor.

Neural Network Approach: "Optical neurons" are realized using saturable two-beam amplification in photorefractive barium titanate; three-dimensional interconnects are realized by using volume holograms in photorefractive lithium niobate.

Status: On-going program.

Results: System concept developed; optical component subsystems have been successfully tested; associative recall has been demonstrated with high-resolution in an optical cavity.

VIS-18: Neocognitron Evaluation

Investigator: William Stoner, SAIC, Bedford MA 617/275-2200

Problem Area: Demonstrate the classification capabilities of the Fukushima neocognitron.

Prior Approach: N/A

Neural Network Approach: Neocognitron as described in Fukushima's article.

Status: Completed simulation.

Results: The character recognition capability of the model were verified as described by Fukushima; certain internal parameters need to be adjusted properly to make the system work; limited translational invariance was also demonstrated.

Future Plans: Considering the application of the model to gray-level images on a parallel machine.

Comments: The model was implemented on an IBM PC/AT, which severely limited the application potential; the model should be implemented on a parallel machine.

Conclusions: Fundamentally, the neocognitron works as well as advertised by Fukushima.

VIS-19: Machine Vision

Investigators: H. Taichi Wang, Bimal P. Mathur, Rockwell International Science Center, Thousand Oaks CA 805/373-4192

Problem Area: How to extract higher-level information from image data. Areas of concern are edge detection, motion detection, edge orientation, stereopsis, and motion field computation. Both algorithms and analog VLSI implementation are being addressed.

Prior Approach: Fast, special-purpose serial computers.

Neural Network Approach: A resistive network has been developed to perform spatial filtering and edge detection. Hopfield networks have been used to implement edge detection, stereopsis and a novel motion field computation network.

Status: Simulations of edge detection and motion field computation using natural images have been performed. Chip design for edge detection is in progress.

Results: Simulations of motion field computation have been successful. A new way for fusing oriented-edge and motion information has been suggested.

Future Plans: Develop new stereopsis algorithm by using edge orientation information. Extend motion field network to full three dimensions. Chip development for extracting oriented edges.

Comments: For both algorithm and circuit development, proper choice of neural representations is crucial for success.

Conclusions: Machine vision is a natural area for neural network application.

SPE-1: Learning to Recognize Speech with Neural Networks

Investigator: Jeffrey Elman, UCSD, La Jolla CA 619/534-1147

Problem Area: The recognition of speech by machine is made difficult by a number of problems, which include:

- Enormous variability in the acoustic signal;
- Interspeaker differences in pronunciation;
- Coarticulation (“blurring” together of adjacent sounds);
- Ambient noise; and
- Difficulties training the system to recognize new speakers.

As a consequence, machine-based recognition systems have had extremely limited success, and typically work in very restricted environments.

Prior Approach: Most recognition systems severely limit the speaker database and vocabulary size; many also require very high signal-to-noise ratios and distinct pauses between words. Most systems depend upon fixed acoustic templates for entire words, and make use of dynamic programming to compare inputs against stored templates.

Neural Network Approach: The neural network approach attempts to capitalize on the architecture used by humans to solve the problem. Much of the variability in speech (both within a given speaker, and between different speakers) is lawful, and reflects the dynamics of articulation. This variability can be exploited, since it provides a high degree of redundancy. The neural network approach, in addition, provides a mechanism for integrating information from many different sources (e.g., acoustic, phonetic, morphological, lexical, syntactic, semantic) in a speedy and elegant manner. Finally, learning algorithms can be applied to neural networks in order to allow for rapid and efficient training.

Status: Work in this area has been underway since 1981 (funded by ONR, NSF, and Army Avionics). A large-scale computer simulation of one neural network (the Trace model) has been completed, and is described in the literature. More recent work is in progress which focuses on the use of new parallel architectures for processing sequential inputs, and the use of neural networks for speech compression and encoding.

Results: The Trace model exhibits a wide variety of behaviors which closely resemble experimental findings regarding how humans recognize speech. The model demonstrates in a dramatic way how the neural network can turn the variability of the signal to its advantage. Results of neural networks for speech compression and encoding are also promising.

SPE-1 — continued

Future Plans: Future work involves a hardware implementation of Trace with the goal of achieving realtime performance. (The system will be based on a network of Transputer processors.) Other work will focus on the representation of time and serial inputs in neural networks.

Conclusions: There are a number of activities which humans do well that machines – to date – do very poorly. Recognition of speech is one of these activities. It is likely that one significant obstacle to machine performance is use of an inappropriate architecture for processing. The neural network approach is far better suited to solving the needs of tasks which involve (a) simultaneous interactions of many pieces of data, (b) flexible responses to dynamically changing inputs, and (c) need for rapid learning. There are some very serious challenges ahead, and neural network technology remains at a relatively early stage. Still, there are certain tasks that can be accomplished within the short-term future (two years) with existing techniques; and there is much more that will be done in the longer term as the consequences of the new approach are explored.

**SPE-2: Speaker-Dependent Isolated-Word Speech Recognition
(e.g. Intel iSBC 576)**

Investigator: Marcian E. Hoff, Jr., Consultant, Los Altos Hills, CA 415/941-3331

Problem Area: Develop a speech recognition module capable of at least a 200-word vocabulary. Speaker dependent (i.e., training required for each user) and isolated-word (i.e., user must provide a short pause between words) requirements acceptable.

Prior Approach: N/A

Neural Network Approach: Speech input is preprocessed to provide spectral information which is then time sampled to produce a two-dimensional data array for each utterance. During training, each array is stored as the coefficients of one template of a matched filter. Each stored template is the equivalent of one "neuron."

Status: The system was realized in the form of a standard Intel SBC printed circuit board (6.75in. × 12.0in.). The spectral preprocessing was performed by two 2920 processors, and the matched filter was implemented by an 8086 microprocessor running a program stored in ERPROM memory. In addition to implementing the matched filter, the microprocessor also performs several other duties such as formatting the templates, communicating with a host computer, and implementing a state sequence machine which controls vocabulary and response. This product was announced in 1982 and has been successfully used in numerous installations.

Results: For the speaker who trained the system, the performance is better than 99% correct recognition. At any one time (i.e., at any one node of the state sequence), the typical number of acceptable words is less than 50. Response time for 50 words is less than one-half second.

Future Plans: Allow self-adaptation by making modifications to the stored templates, and modify the algorithm to allow more continuous speech input.

Comments: A major application for the system has been inspection of automobiles during manufacturing. The use of speech recognition allows the inspector to have his hands free while he maneuvers about the vehicle. Prior methods required him to carry a clip board and pencil.

Conclusions: A limited vocabulary speech recognition system can significantly improve efficiency for certain data collection processes where the user needs to have his hands free for other functions.

SPE-3: Speech Recognition

Investigator: D. L. Reilly, Nestor, Providence RI 401/331-9640

Problem Area: Pattern recognition and classification.

Prior Approach: Rule-based and statistical systems.

Neural Network Approach: Use of the Nestor Learning System to train a neural network on the so-called T.I. Database. A proprietary third-party feature extraction process reduced the data description of each word utterance to a single 512 bit pattern.

Status: Completed.

Results: Speaker dependent test: 97.7% accuracy; speaker- independent test: 97.0% accuracy.

SPE-4: Mapping Visems to Phonemes for Visual Speech Recognition

Investigator: Ben Yuhaz, Johns Hopkins University, MD

Problem Area: Visems are characteristic mouth shapes associated with normal speech production; it has been found that under noisy conditions speech perception is significantly enhanced by visual input.

Neural Network Approach: Train a fully-connected three-layer network to associate a 25×20 pixel representation of the mouth area as input to a 30-unit phoneme representation in the output layer via one hidden layer of 20 units; the phonemes are expressed as a vocoder representation.

Status: On-going research project.

Results: In a preliminary test 70 pictures and associated sounds have been trained and compared to a nearest-neighbor classifier with promising results.

Future Plans: Finish the project, a Ph.D. thesis, under supervision of T. Sejnowski.

Comments: It is not clear whether the most optimal neural network structure and data representation has been found so far.

SON-1: Learned Classification of Sonar Targets Using a Massively Parallel Network

Investigator: Paul Gorman, Allied Signal, Columbia MD 301/964-4094

Problem Area: For the design of classifiers in intelligent sensor systems, suitable features must be found in the signal; in collaboration with T. Sejnowski, the ability of a neural network to find features is explored in the following sonar target classification problem: low-noise sonar signatures were obtained in seawater from a metal cylinder and a rock of equivalent size. About 200 signatures were used as a training/test set.

Neural Network Approach: Three-layer, fully-connected, feedforward network, taught by backpropagation, with 60 input units, two classification outputs, and one hidden layer with 0-24 units. The signals were represented as a power spectral envelope over a sliding time window.

Status: Completed research project.

Results: For training, about 100 signatures of the sample set were presented $300\times$; a sensitivity-to-random-weight initialization and to training set sequence was noticed; results are averages over a total of 10 trials. (These results are from the aspect-angle-dependent experiments.) Twelve hidden layer units were sufficient to give better than 99% accuracy on training sets and about 90% on the test sets; a two-layer network gave 73%, and human operators gave about 88% correct classification. Analysis of the weight distribution for the hidden layers supports the notion that the network primarily isolates by location of principle attack, bandwidth of the main feature, and decay; humans seem to decide on similar principles; the slightly better performance of network versus human is attributed to the presence of about 10-15% "non-standard" signals which do not follow the simple rules and which the network can store explicitly.

Future Plans: Optimal encoding of data, develop a dynamical model based on the present work.

Conclusions: A multi-layer network can be trained as a classifier and may help in the discovery of structure or hidden features implicit in unknown data.

SON-2: Sonar Target Classification with a Neural Network

Investigator: J. Harold McBeth, General Dynamics, San Diego CA 619/573-7857

Problem Area: Classify underwater sonar targets according to type of ship.

Neural Network Approach: The FFT of the received signal is split into 20 mutually-overlapping frequency bins which are introduced in temporal order into a forward-connected avalanche network.

Status: A PC-based simulator model has been developed.

Results: The neural network has been trained on approximately 13 recorded waveforms (each 600 ms duration, divided into 20 overlapping temporal segments) of different ships. A waveform from one of these 13 classes can be properly classified, even when the waveform is corrupted with artificial noise.

Future Plans: A five-year development time is projected for a field- deliverable unit.

Comments: Presently the demonstration lacks realistic test data; but realtime use of live test data is in development. The system immunity against real-world perturbations of the received signal needs to be demonstrated.

Conclusions: Conclusive evidence for a successful application has not been demonstrated, although it is working under laboratory-controlled conditions.

RAD-1: Adaptive Radar Processing

Investigator: Dean Collins, Texas Instruments, Dallas TX 214/995-3974

Problem Area: Anti-radar tracking problem: passively locate and identify a number of Radar emitters; prioritize action according to radar importance. Data are typically very noisy.

Prior Approach: Conventional signal processing.

Neural Network Approach: Brain state in box (BSB) model to achieve clustering in a multi-dimensional decision space.

Status: Ongoing program sponsored by Wright Patterson AFB.

Results: 500 pulses, closeness encoded, were fed to the neural network; 10 out of 10 transmitters were properly located and identified.

Future Plans: Still 20 months (of 24) to go; direct comparison of neural network approach to conventional techniques is to be emphasized.

Comments: This is a program close at the edge of being classified. Dr. Collins felt that too many classification restrictions on as young field as neural networks would be very counterproductive.

Conclusions: In this early stage of development, the neural network approach gives reasonable results, as compared to standard techniques, and shows good operation in handling the War Reserve Mode of operation.

RAD-2: Satellite Orbital Motion Detector from RCS Data

Investigator: Mitch Eggers, MIT/Lincoln Laboratory, Lexington MA 617/981-2664

Problem Area: Classify a database of radar cross-section returns from six satellites which perform three distinguishable orbital maneuvers.

Prior Approach: Classical classification algorithms.

Neural Network Approach: Two-layer perceptron model with restricted connectivity; presently the data are preprocessed to extract a single decision parameter. An enhanced (as compared to backpropagation) learning algorithm for multi-layer networks has been developed.

Status: On-going program.

Results: For the present database, the trained neural network model was sufficient to identify all orbital motion transitions; for the case of a Gaussian distributed variable, the model was shown to perform as well as a classical Neyman-Pearson classifier.

Future Plans: Extend the methodology to include more information extractable from the radar cross section data.

Conclusions: The strength of the neural network classifier is primarily that data with unknown statistics can be easily accommodated; the optimal decision threshold is implicitly extracted from the training examples.

RAD-3: Radar Target Recognition from Partial Information Based on Models of Neural Networks

Investigator: Nabil Farhat, U. of Pennsylvania, Philadelphia PA 215/898-5882

Problem Area: Recognition of non-cooperative radar targets from limited information.

Prior Approach: Images formed by microwave diversity imaging systems are recognized by human operators. Imaging radars are prohibitively costly and/or can not operate in real time.

Neural Network Approach: Super-resolved automated recognition from sketchy (incomplete and/or noisy) information employing heteroassociative storage and recall. Sinograms extracted from realistic microwave scattering data are utilized as target representations. They offer potential for distortion-invariant recognition from a small number of looks (as low as three for three objects stored in a neural network of 32×32 neurons has been demonstrated). For practical applications, recognition from one look regardless of aspect is desirable. This approach could obviate the need for expensive microwave imaging systems.

Status: On-going research program.

Results: Neural network of 32×32 neurons performs the function of storage, processing, and labeling simultaneously and discriminates between three radar targets from as low as 10% of the sinogram information.

Future Plans: Make self-organizing neural networks form their own representations of the radar echoes and recognize any one of them at a later time with generalization.

RAD-4: Multi-Target Tracking

Investigator: Robert Kuczewski, TRW, San Diego CA 619/592-3381

Problem Area: In the absence of continuous surveillance, multiple, possibly crossing, target tracks need to be constructed from sparse information.

Prior Approach: Traditionally, the combinatorial explosion is overcome by heuristic decision algorithms which reduce the permissible solution space but also may introduce error. An Hopfield network approach was also used, but was dropped when it achieved about 20% illegal solutions at best.

Neural Network Approach: Developed a biologically-inspired track-filtering mechanism based on a temporal extension of the boundary contour system such that time continuity of moving objects is maximized. A distributed representation, assigning one neuron to each node in a spatial grid with a dipole-type receptive field (defining the dynamical constraints of the objects and) transferring the influence of neighboring positions, leads to a fully parallel cellular automata-type approach (interpolative probability field, IPF).

Status: Complete simulation system operational.

Results: Many track scenarios have been simulated; tracks stabilize in as little as five consecutive time frames.

Future Plans: The IPF filter needs to be integrated with a higher-level cognitive structure (as, e.g., Grossberg's masking field or adaptive resonance).

Comments: A fundamentally parallel approach which is independent of the number of assigned tracks since the individual processing elements completely span the covered space. Object constraints are fully expressed in the functional form of the probability distribution for the receptive field.

Conclusions: Approach leads naturally to a unified two-step solution of multi-target tracking where filtering (object constraints) and correlation (object recognition) form a self-consistent, mutually reinforcing, resonant structure.

RAD-5: Neural Network Radar Signal Processor for IFF

Investigator: Fred Weingard, Booz-Allen, Washington, DC 703/769-7782

Problem Area: Need an adaptable classifier allowing new patterns to be classified as the threat environment changes.

Prior Approach: Compete with AOSR-96 radar.

Neural Network Approach: Consider seven radar features that map onto a Grossberg masking field (giving a combinatorial set of nodes for each possible configuration of the features). Novelty is to be handled by an ART network as an additional logic layer. The resulting network should lead to fast learning and novelty classification.

Status: Program sponsored by Avionics Lab, WPAFB.

Results: Concept stage. Component simulations need to be done; for this purpose, a computer-aided design simulation system is being developed. Once the multi-layered network can be constructed on the simulator, the bulk of the one year effort will be spent on adjusting the parameters and retrying the network with training and test data.

SIG-1: Satellite Classification

Investigator: Steven Lehar, TEXTRON Defense Systems, MA 617/381-4799

Problem Area: The recognition of satellites by their Long-Wave Infrared (LWIR) signatures.

Prior Approach: Human expert.

Neural Network Approach: Generalized delta rule backpropagation algorithm with 500 input, 500 hidden, and three output units.

Status: Preliminary tests complete. This work was supported under the AMOS/MOTIF GFY 86-90 contract F30602-85-C-0179.

Results: The system was trained in less than an hour-and-a-half on unprocessed (other than range and atmospheric attenuation compensation) LWIR satellite signatures (one-dimensional temporal plots of non-spatially- resolved intensities) for three different satellites using nine, 10, and 11 distinct signature examples, respectively. Blind tests with two signatures for each satellite yielded six out of six correct classifications with over 90% certainty on the first trial, and five out of six correct on a second trial with different signatures withheld for the blind test.

Future Plans: Conduct more extensive tests on larger database, optimize performance by varying system parameters, investigate preprocessing feature representations and investigate different network algorithms.

Comments: What was started initially as a pure research program produced a practical and much needed analysis tool ready for immediate implementation. This technology surpassed all expectations and further study is recommended.

Conclusions: Neural network algorithms show immediate advantages over existing algorithms for certain application areas.

SIG-2: Removing Random Noise from EKG Signals

Investigator: Douglas Palmer, HNC, San Diego CA 619/546-8877

Problem Area: Train a backpropagation network on a continuous EKG data stream to achieve a low-noise waveform estimate from the noisy input data.

Prior Approach: To compare the neural network performance, a finite impulse response (FIR) filter with 40 adaptive taps was trained using the same training set as for the back-propagation network.

Neural Network Approach: A moving window frame is formed from the last 40 samples of the input, each consecutive sample being $200 \mu\text{s}$ long; hence, the network looks at about $1/5$ of the heartbeat period at any one time. A fully-connected three-layer network with (40, 10, one) units was used; the single output unit is trained to give the noise-free waveform prediction for the present sample time. For supervised training, the same waveform is presented to the neural network with and without artificially added noise.

Status: Finished demonstration program.

Results: The neural network showed higher fidelity than the FIR filter approach, even for waveforms for which it had not been trained.

Comments: It is not clear whether the neural network has “learned” the shape of the EKG spike and is able to reproduce this low-noise spike at the appropriate time.

Conclusions: The backpropagation network implements a nonlinear filter and can change its frequency response dependent on the input signal; this property allows many filtering tasks that are impossible to perform with linear filters.

ROB-1: CMAC: The Cerebellar Model Arithmetic Computer – A Neural Network Robot Control System

Investigator: James S. Albus, National Bureau of Standards, Gaithersburg MD 301/975-3418

Problem Area: An adaptive learning neural network for sensory- interactive robot control.

Prior Approaches:

- In neural networks: Perceptron, Adeline.
- In brain modeling: Marr-Albus theory of cerebellum.
- In servo control: State-space adaptive control.

Neural Network Approach: CMAC is a feedforward mapping network, with fast response times, which utilizes a distributed input representation by mapping the input state vector into a $100\times$ higher-dimensional space. Weights are adjusted by error correction of the mapped variables. Learning by teaching a desired trajectory leads to topological generalization. Only a few training samples are needed to support a smoothly varying control or decision surface. CMACs can be hierarchically chained allowing high-level to low-level command decomposition.

Status: Original work completed and published in 1975. Current work under way at University of New Hampshire for self-learning of feedforward control for robot dynamics [Thomas Miller]. Proof of convergence of training algorithm at Cranfield [P. C. Parks].

Results: Looks very good for learning sensory-interactive tasks and for learning dynamic system parameters. Needs to be embedded in larger control system architecture to be useful. [See also V. Jennings: ROB-5, p. 168.]

Future Plans: Apply to high-performance, dynamically-complex robot control problems, such as manipulator force/stiffness control, or driving high-speed land vehicles.

Conclusions: CMAC may allow programming by teaching of complex sensory-interactive tasks. This could have significant impact on software development for advanced robotic applications.

ROB-2: Control Systems that Learn

Investigator: Dr. Richard Elsley, Rockwell International Science Center, Thousand Oaks CA
805/373-4155

Problem Area: Control of nonlinear systems that have unknown, computationally intractable, or time-varying properties.

Prior Approach: Explicit calculation of inverse kinematics/dynamics; adaptive control.

Neural Network Approach: A controller architecture based on a backpropagation neural network that learns to control the system, beginning with no knowledge of system properties and using no external teacher. Uses inverse Jacobian formulation.

Status: Simulation of a robot learning to reach for objects it can see (i.e., kinematic control) beginning with no knowledge of what effect commands to its arm joints will have and receiving no direct feedback on the success of individual joint commands.

Results: After training, robot will reach goal object with arbitrary accuracy. A single presentation of about 100 randomly chosen goal objects is sufficient for training. After training, changing the system, cutting synapses, etc., is followed by relearning.

Future Plans: Extend to dynamic control, flexible robots with variable loading, cooperating robots, other systems.

ROB-3: Parallel Distributed Control of Robotic Systems

Investigator: Jack Gelfand, David Sarnoff Research Center, SRI, Princeton NJ 609/734-3098

Problem Area: Present commercial hardware cannot do realtime inverse kinematics, and dynamics for force and position control.

Prior Approach: Conventional methods provide precision and speed for low-degree-of-freedom manipulator in a structured environment.

Neural Network Approach: Combine distributed processing of reflexive control strategies with high-level hierarchical control for execution of complex arm movements; design strategies inspired by biological control systems which are studied and simulated in concurrent efforts.

Status: On-going multi-disciplinary research effort.

Results: Computer simulations of a multi-joint "ball-throwing" action with adaptive learning.

Future Plans: Implementation of control strategies on custom robot arm hardware. Plans to build mechanical systems and compare them with experimental data from motor physiology.

Conclusions: Engineering efforts inspired by biology will lead to effective and efficient control strategies for complex robotic systems.

ROB-4: Tool-use in Biological and Artificial Systems

Investigator: Neville Hogan, MIT, Cambridge MA 617/253-2277

Problem Area: Sensory guided control of artificial manipulator arms without control-loop-induced instabilities.

Prior Approach: Conventional closed-loop control leading to contact instabilities and sub-optimal computation-limited performance.

Neural Network Approach: Not a neural network approach per se; yet the design philosophy evolved from studying the performance parameters of animal and human limb motion. The central tenet is to control the “impedance” of the mechanical system (not the force or motion, as in conventional systems).

Status: On-going program.

Results: A video tape was shown of a robot arm designed for extremely high-speed response; the arm will move with high speed without colliding with arbitrarily moving obstacles which are recognized by an optical beacon system.

Future Plans: Extend applications to control of cooperating articulated, kinematically redundant robots.

Conclusions: The system emulates important architectural features of biological systems (e.g., local force/velocity relations as impedances) and demonstrates their effectiveness in an engineering system specifically designed to exceed biological performance.

ROB-5: Trainable and Adaptable Neural Networks for Robot Control

Investigator: Von A. Jennings, Martin Marietta, Baltimore MD 301/682-0892

Problem Area: Robot control needs to integrate high-level and low-level control with environmental input and feedback for effective action.

Prior Approach: Current control techniques attempt to model the system explicitly. The difficulties of this approach include: computation too slow for realtime control; all model parameters are not known. High-level control using state table task decomposition needs an unmanageable collection of explicit rules.

Neural Network Approach: Implemented a CMAC (cerebella model arithmetic computer [ROB-1, p. 164]) approach for integrating optical range sensors (uncalibrated and at various roughly orthogonal directions) with a six-degree-of-freedom fork-lift robot arm. The CMAC is taught by explicit example given by a human operator.

Status: On-going research effort (internally sponsored).

Results: Sensory-guided object acquisition demonstration performs the task of picking up a pallet; the robot will adapt in real time to positional changes of the pallet.

Future Plans: Evaluate a fully-connected multi-layer network for control functions presently implemented by CMAC. Extend CMAC control structures to the operation of an anthropomorphic robot arm and a vehicle obstacle avoidance problem, all taught by example of a human operator.

Comments: This program has been submitted to DARPA for funding. The need to test controllers on real robot hardware is stressed.

Conclusions:

- Control functions are stored implicitly without equations or rules as stored patterns of weights and connections and acquired by learning.
- Inherent parallelism of the CMAC approach allows multiple control functions to be performed simultaneously in real time.

ROB-6: Robot Navigation

Investigator: Chuck C. Jorgensen, Thomson CSF-pro, Palo Alto CA 415/494-8818

Problem Area: Use of sonar maps for autonomous guidance of a robotic vehicle and anticipation of terrain in new environments.

Prior Approach: AI techniques.

Neural Network Approach: Sensor is a phased array of six four-element Polaroid ultrasound transmitters; raw data of sonar map are stored in a Hopfield-type neural network with sensor data used as recall cues.

Status: A demonstration unit is operational.

Results: Robot can build up a 'world map' of a laboratory environment with simple obstacles and plan a navigation path.

Future Plans:

1. Apply self-organizing maps to robot path planning and control;
2. Study joint neural network/expert system robotic learning.

Conclusions: Anticipation of partially learned environments is a key problem in merging local and global sensor data for autonomous robotics. Neural networks provide a potential solution method.

ROB-7: Neural Dynamics of Adaptive Sensory Motor Coordination

Investigator: Michael Kuperstein, Wellesley College and Neurogen, Wellesley MA 617/739-2215

Problem Area: Design a computational map representation that learns to coordinate a multi-joint robot arm and stereo cameras for manipulating novel objects.

Neural Network Approach: Use of an ordered, highly distributed representation of the input variables both for the visual and the motor control system. These distributions can be mapped together by an environment-driven, self-consistency requirement; no explicit teacher is needed to monitor successful mappings.

Status: Simulations for a static robot model and for a dynamic one- joint model with viscous forces have been completed.

Results: The error of this self-consistent eye/arm map simulation is 4% of arm's length and 4 degrees of arc. Presently, this approach is applied to a standard robot arm and camera system.

Future Plans: Develop a payload-independent mapping which also includes dynamical properties of a robot arm in motion.

Comments: The model is actively pursued for its commercial application potential.

Conclusions: An unsupervised computational mapping model has been developed which relates favorably to known aspects of equivalent biological systems.

ROB-8: Teaching an Adaptive Network with Visual Inputs to Balance an Inverted Pendulum

Investigators: Viral V. Tolat, Bernard Widrow, Stanford U., Stanford CA 415/723-4949

Problem Area: Demonstrate that a skilled person can teach an adaptive network to make real-time control decisions based on the dynamic analysis of visual input data.

Prior Approach: Realtime control of an inverted pendulum using quantized state variables as inputs was done by Widrow and Smith in 1963. More recently, adaptive network control of an inverted pendulum was done by Barto, Sutton, and Anderson, where learning was achieved by punishment/reward learning.

Neural Network Approach: Quantized visual images of the present and recent past positions of the cart and pendulum are fed to the adaptive network. Training signals (desired responses) are obtained from a skilled person (the teacher). The adaptive network learns to recognize the important features of the visual images for purposes of control.

Status: Control of a simulated inverted pendulum (broom-balancer) by an adaptive network has been successfully demonstrated using a Macintosh to generate the display and to implement the differential equations of the cart and pendulum.

Results: When training on only 20% of all possible visual input images, 96% of the control decisions for the remaining visual input images were correct. With limited training, the adaptive network was able to stabilize the pendulum indefinitely in spite of small errors (4%) in its decisions.

Future Plans: New problems and applications are being explored for adaptive network control. Emphasis is placed on cases where a human expert is available to teach the adaptive controller. By studying enough examples, general concepts will appear making it possible to develop a science of trainable expert systems.

Conclusions: It has been demonstrated that an adaptive network trained by a human expert has been able to generalize, from limited training, to successfully control an inverted pendulum on a cart. The adaptive network has been able to abstract from visual inputs the features (state variables) that were critical for control purposes. It was not told what to look for. This work may lead to a new form of man-machine interaction, where a person trains the machine rather than programming it.

DYN-1: GTE Process Monitor

Investigator: Richard Sutton, GTE Laboratories, Waltham MA 617/466-4133

Problem Area: Predict from 100-200 process sensors the yield of a fluorescent bulb process line in real time.

Prior Approach: Weekly batch processing of data off-line.

Neural Network Approach: On-line neural network weight update (every 15 minutes) using a novel temporal-difference learning algorithm.

Status: Conceptual development.

Results: System under installation in 1988.

Future Plans: Move from process monitoring toward adaptive process control.

Comments: The temporal-difference learning rule offers new applications of neural networks for dynamic control problems.

Conclusions:

- Efficient, low-memory utilization of large data flow in a non-stationary monitor situation.
- Implicit neural network model of process developed from incremental weight updates based on partial information.
- Can be readily compared with conventional methods.

BIO-1: Motion Detection in Fly and Machine Vision

Investigator: Heinrich H. Buelthoff, MIT, Cambridge MA 617/253-0549

Problem Area: Study biological motion detection in flies as a model system for a high-performance visual system. Behavioral studies of the optomotor and fixation response of flies have founded a mathematical theory of the orientation behavior of flies. Open-loop studies of the optomotor behavior in a “fly-flight-simulator” can be well explained by a movement detector model based on multiplication of image intensities after asymmetric temporal filtering (Correlation-Model; Hassenstein and Reichardt, 1964). By combining neuropharmacology and electrophysiology, an effort was made to distinguish between different neuronal models of this essential multiplication-like interaction. By blocking inhibitory interactions with picrotoxinin, an antagonist of the inhibitory neurotransmitter GABA, most of the directional selectivity of a large-field movement-sensitive neuron (H1- cell) in the third optical ganglion of the blowfly *Calliphora erythrocephala* could be abolished (Schmid and Buelthoff, 1988). These modifications are similar to changes in the optomotor response of the fruitfly *Drosophila melanogaster* observed in flight- and locomotion-simulators after application of picrotoxinin (Buelthoff and Buelthoff, 1987a,b). These results are compatible with inhibitory synaptic interactions at the level of elementary movement detection as proposed by Torre and Poggio (1978).

A new parallel and fast algorithm for computing the optical flow and its implementation on the Connection Machine has been described (Buelthoff and Little, 1986; Little *et al.*, 1987). This algorithm is motivated by the study of simple biological motion detection mechanisms as described above. It is based on a regularization technique that exploits a simple assumption – that the optical flow is locally uniform. It can be easily translated into the following description: Consider a network of elementary motion detectors holding the results of multiplying (or logical “AND-ing”) image features (intensity or edges) for different displacements. Each detector collects a vote indicating support that a patch of surface exists at a certain displacement in the second image. The final step is to choose the velocity $v(x, y)$ out of a finite set of allowed values that has maximum vote (non-maximum-suppression or winner-take-all scheme). The corresponding $v(x, y)$ is taken as the velocity of the point (x, y) .

Status: On-going research sponsored by ONR.

Future Plans: Develop ideas based on biological mapping functions (retinotopic and cortical mapping) to reduce the two-dimensional search in the voting algorithm to a one-dimensional search. This should increase the performance of the voting algorithm (four seconds on the Connection Machine) to realtime performance even on standard von Neumann architectures.

Comments: The study of biological hardware and biological information processing strategies and much more careful mathematical analysis of neural network algorithms is essential for any progress in neural network research.

BIO-2: Central Pattern Generator for Locomotion in Lamprey

Investigator: Avis Cohen, Cornell University, Ithaca NY 607/255-8997

Problem Area: Investigate experimentally the locomotive pattern generation capabilities of the Lamprey spinal cord and compare to model based on a system of coupled limit-cycle oscillators.

Prior Approach: N/A.

Neural Network Approach: Isolated sections of the spinal cord can be prepared for studies *in vitro* lasting several days; detailed electrical and chemical stimulations can be performed and cellular analyses performed.

Status: Presently funded by AFOSR URI program.

Results: Model system of a distributed neural network capable of generating coherent non-local oscillatory states.

Comments: Funding of this program was recently curtailed by more than 25%, which eliminates the planned post-doctoral investigator and will seriously reduce future progress. Dr. Cohen feels that across-the-board cuts are inappropriate and inefficient for small programs, since a minimum funding threshold exists for continued research capability.

Conclusions: A case where close integration of biological research with mathematical modeling efforts can produce an engineering description of the properties of a complex system of interacting neuronal circuits.

BIO-3: Study of Neuronal Networks Artificially Grown on Microelectronic Probe Substrates

Investigator: Mitch Eggers, MIT/Lincoln Laboratory, Lexington MA 617/981-2664

Problem Area: Collect simultaneous multi-channel recordings of artificially grown neuronal networks of limited size.

Prior Approach: N/A

Neural Network Approach: Different types of nerve cells can be grown directly on the planar, microelectronic recording substrate.

Status: A modular electronics cell with sterile nerve cell carrier has been developed and built; nerve tissue growth techniques are being developed at MIT.

Comments: This work is similar in approach to the work reported by G. Gross [BIO-4, p. 177]. The distinction lies in the attempt to control the architecture of the living neural network (ideally one neuron per electrode) to enable true modeling.

Conclusions: The collective properties of artificially-selected and -grown nerve cell systems will be studied *in vitro*.

BIO-4: Investigating Mammalian Neural Network Modeling Parameters

Investigator: Guenter Gross, North Texas State U. Denton TX 817/565-3615

Problem Area: Study the electrical and chemical characteristics of neuronal network *in vitro*.

Prior Approach: Single-cell response measurements of *in-vivo* or *in vitro* neurons.

Neural Network Approach: Develop techniques to grow and prepare two- dimensional, artificially grown neuron networks; study these preparations of 100-400 fully isolated neurons as a function of electrical and chemical stimulation and by selectively removing neuron connections with a laser scalpel; 32 electrode channels can be recorded simultaneously.

Status: Operational system, presently upgraded to include 64 channels of high-speed extracellular recording; any 32 channels can be selected for processing.

Results: Simultaneous multi-channel recordings of such neuron systems have shown a pharmacologically induced change of firing rate. Characteristic burst patterns have been manually classified to give a simplified representation on which to look for inter-neural correlations.

Future Plans: Develop techniques to automatically scan the data and find the model parameters of the neuron network. Correlate circuit structure with circuit electrical activity. Stimulate electrically to study signal storage.

Comments: It will perhaps be necessary to invent a neural network computer to successfully relate and interpret the large amount of data generated by these studies.

Conclusions: Techniques have been developed to study in full detail the temporal dynamics of interconnected networks of real neurons. With these techniques, one can study network dynamics without a need for specific input and output circuits (i.e., "system dynamics"). It is the analog pattern generation and pattern recognition properties of neuronal networks that may hold the key to new computer architectures which specialize in rapid pattern recognition. Dr. Gross strongly feels that digital architectures will not be efficient for rapid pattern recognition and classification.

BIO-5: Motion Measurement in the Vertebrate Retina

Investigator: Norberto Grzywacz, MIT, Cambridge MA 617/253-0545

Problem Area: At the early vision level, what are the proper models to extract, e.g., motion from the visual field? Specifically:

- a) What is the delay mechanism?
- b) What are the interactions involved and where do they occur?

Prior Approach: Machine vision techniques.

Neural Network Approach: Direct stimulation experiments on rabbit retina.

Status: Ongoing program.

Results:

- a) The delay mechanism is the result of excitatory (not sustained) and inhibitory (sustained) interaction to give a speed-independent direction-selective cell response; cells in a preferred direction, however, are very speed-selective.
- b) Experiments have been performed to distinguish between a multiplicative shunting-type model (inhibition shunts excitatory dendrite inputs) and a hyperpolarizing thresholding-type model (e.g., hyperpolarization reduces synaptic transmission from amacrine cells); the latter model is consistent with the data.

Comments: A good example of “applied biological science” with the goal to elucidate working models of visual information processing in biology.

Conclusions:

- The same cells are interconnected to give both direction-sensitive and speed-sensitive information.
- A hyperpolarizing model for the amacrine cells accounts for retinal stimulation experiments.

BIO-6: Self-Organizing Computational Maps

Investigator: John Pearson, David Sarnoff Research Center, SRI, Princeton NJ 609/734-2385

Problem Area: Study the biological organization and theoretical modeling of computational maps; a specific problem is the sensory fusion of visual and auditory information in the barn owl.

Prior Approach: N/A

Neural Network Approach: Defined a competitive, sensor-stimulated, reaction-kinetic model for demonstrating adaptive computational map organization.

Status: On-going program.

Results:

- Simulated sensory fusion—automatic registration of visual and acoustic receptive fields.
- Found dynamically stable clustering of receptive fields in a computational map model.

Future Plans: Apply such computational models to real-world engineering problems.

Comments: This work is based on a fruitful cooperation of biologists, physicists, computer and cognitive scientists.

Conclusions: Preliminary results show that computational maps can self-organize and are stabilized (and modified) by external sensory input (changes).

BIO-7: DARWIN-III: A Selective Recognition Automaton

Investigators: Gerald M. Edelman and George N. Reeke, Jr., The Neurosciences Institute, New York NY 212/570-7627

Problem Area: In understanding perception and the higher cognitive capacities of the brain, the organization and operating principles of the nervous system are fundamental. According to the theory of neuronal group selection, these capacities are thought to emerge from epigenetically determined, distributed networks of excitatory and inhibitory nerve cells. Networks are structured into maps and develop adaptive functionality as the result of interaction with the environment without explicit computational or learning algorithms. Adaptation is dynamic and competitive and allows the organism to respond continually to changes and novelty in the environment.

Prior Approach: N/A

Neural Network Approach: A general-purpose simulator for networks of cells with biologically realistic properties has been written. In a typical application, 8 000 neurons with 200 000 connections are arranged to form a functional automaton, Darwin III, with integrated units for vision, arm movement, search, and categorization. Darwin III is a "creature" based on biological principles. It has a phenotype, nervous system, and behavior. It generates a categorization of nearby objects based on visual and tactile cues, and it responds differently to stimuli in different categories depending on their adaptive value for it.

Status: Working code.

Results: Successful demonstration of visual saccades, arm reaching, and multi-modal categorization of objects with appropriate behavioral responses.

Future Plans: Continue development to achieve higher performance and more complex functions, including behavioral conditioning of the "creature."

BIO-8: Spatial-Temporal Coding in Dynamic Models of Brain Function and Potential Applications

Investigator: Gordon Shaw, UC Irvine, Irvine CA 714/856-6620

Problem Area: Find a workable model to describe the temporal and spacial dynamics of biological neuron activities; how does the nervous system encode and process information with these patterns?

Prior Approach: Some neural network models (e.g., the Hopfield model) neglect such temporal dependence.

Neural Network Approach: Rather than starting with individual neurons, the fundamental unit of a neural network is postulated as a group of 30-100 tightly-coupled neurons, called a Trion; such a collection represents a tri-state device: a background firing activity, and activity above or below background.

Status: On-going research program.

Results: Computer simulations show that networks of a few Trions give rise to a large variety of dynamically stable patterns which can be selected and enhanced by Hebb-type learning; a certain structure, a balance of inhibition and excitation, and statistical fluctuations are essential in producing stable firing patterns; this work is successfully being applied to an interpretation of multi-electrode data from mammalian primary visual cortex.

Future Plans: Three potential applications of the Trion model:

1. Motor systems – identify the stable patterns with the activation patterns necessary for reflexive motions.
2. Smart sensors – process sensory data as complex signals extended in space and time and “resonate” with the Trion states for identification.
3. Integrating (1) and (2) provides an interface for complex sensory and motor systems.

Conclusions: Such structured, Trion-like neural networks give rise to synchronous firing patterns; the spatial-temporal firing patterns are non-factorizable.

DEC-1: Optimal Allocation of Active Sensors

Investigator: George Fusik, TRW, Redondo Beach CA 213/535-0350

Problem Area: SDI—Optimal allocation of defense assets.

Prior Approach:

1. Human decision making.
2. Knowledge-based systems (AI programs on traditional computers).
3. Optimal assignment algorithms executed on traditional computers.

Neural Network Approach: Three-layer network with fixed interconnection weights; negative feedback from second and third layer into first: used to reduce list of possible optimal candidates for “next best to look at with sensor” to one. Interconnection weights are determined by neural network emulation on TRW’s MARK-III neural network emulator. Overall performance of network is also evaluated with the MARK-III emulator.

Status: Investigated behavior of neural networks implementing Hopfield’s “traveling salesman” approach and a modified Hopfield “traveling salesman” approach (Hopfield with simulated annealing) to optimal visitation of targets. Developed current approach, which attempts to elicit desired mathematical behavior from the neural network.

Results: Hopfield approaches found to work well (not great) but not as well as traditional optimal assignment algorithms. The new approach, it is felt, will work better.

Future Plans: Test out new approach – determine values of all constants in network formulas. Develop other approaches, if necessary, and select the best. Develop electro-optic neurocomputer to implement selected the selected optimization approach.

Comments: Real progress depends on funding levels.

Conclusions: Potential for application of neural networks to optimization problems exists, although work needs to be done to find the best approaches (theoretical development). Optical implementation of neural networks is essential for network speed and to manage the complex interconnections required.

DEC-2: Mortgage Delinquency Prediction

Investigator: C. L. Scofield, Nestor, Providence RI 401-331-9640

Problem Area: Risk Analysis.

Prior Approach: Statistical methods.

Neural Network Approach: Use of the Nestor Decision Learning System to train neural networks on home mortgage applications classified as Good risks or Bad risks on the basis of their performance over a period of 2.5 years.

Status: Completed.

Results: Significant reduction of delinquencies compared to performance of human underwriters.

Future Plans: Extension of this work to other risk analysis areas.

Comments: This application is presented to illustrate the use of the Nestor Decision Learning System to Risk Analysis.

Conclusions: The Nestor Decision Learning System is able to perform the risk analysis more systematically than human operators and thereby is able to reduce, in the cases tested sofar, undesirable outcomes.

DEC-3: Mortgage Loan Evaluator

Investigator: C. L. Scofield, Nestor, Providence RI 401/331-9640

Problem Area: Risk analysis.

Prior Approach: People trained in the art of good underwriting.

Neural Network Approach: Use the Nestor Decision Learning System of neural networks to learn and match the underwriter behavior implicitly contained in a database of 10 000 mortgage applications. The system utilizes several three-layer networks with restricted connectivity (RCEs) which are adaptively allocated by a system controller that also provides system output to the user through an application-specific user interface.

Status: Operational system.

Results: The system is configurable from “low-risk” to “high-throughput” mode of operation. The neural network model matches the underwriter from 97% to 87% of the time when performing simple yes/no decisions in low-risk to high-throughput mode, respectively. Where the system differs with the human underwriter, its decisions are better as judged by actual loan performance.

Conclusions: The Nestor Decision System offers the following advantages:

- Automatically trainable multi-level, hierarchical system.
- Efficient training times for supervised learning.
- Fully expandable to larger databases without retraining.

THE-1: Practical Results from Neural Network Theory

Investigator: Yaser Abu-Mostafa, CalTech, Pasadena CA 818/356-4842

Problem Area: Summarize theoretical results pertinent to neural network applications:

1. What are neural network limitations?
2. How do neural networks compare with conventional methods?
3. What problems are suitable for neural networks?
4. How difficult is it to learn?
5. Are there advantages for analog computations?

Results:

- (1) What are neural network capabilities?
 - (a) Storage: Capacity estimates exist for Hopfield and feedforward networks and are relatively low.
 - (b) Computational power: Perceptrons can do any linearly-separable decision; a fully-connected three-layer network can map any Boolean transformation.
- (2a) “Robustness” of neural networks: there are no general results;
- (2b) “Structured” problems (for which an algorithm exists, such as the traveling salesman problem) are not suitable – to solve such an NP-complete problem would require an exponentially increasing number of neurons.
- (3) Problems suitable for neural networks are those which cannot be described by a short algorithm, such as random and pattern-recognition problems; in such cases, the neural networks are “programmed” by example.
- (4) “To learn” means here to define a given I/O mapping; a general mapping is NP-complete, therefore prior knowledge and existing structure of the problem need to be embedded as useful “hints” via proper choice of representation.
- (5) There exist no theoretical results concerning the advantages of analog computations; experimentally, however, analog devices give fast decisions and high accuracy when used with feedback.

THE-2: Phase-Locking of Neuronal Circuits

Investigator: Frank C. Hoppensteadt, Michigan State U., East Lansing MI 517/355-4473

Problem Area: Mathematical methods have been developed to model the frequency-response properties of neurons and to show that neural networks can support stable patterns of synchronized firing.

Neural Network Approach: The neuron is described by a voltage-controlled oscillator model.

Status: Ongoing program.

Results: Synchronization of neuron firing have been modeled in small and large networks of neurons. Memory is described by the phase-locked response of a network of such oscillators; collective properties such as a Hebbian learning rule have been demonstrated.

Comments: The main results have been summarized in Dr. Hoppensteadt's book, *An Introduction to the Mathematics of Neurons* (Cambridge University Press, London, 1986).

Conclusions:

- A model for the frequency-response properties of neurons has been developed.
- A neural network system of such neurons is capable to form stable patterns of synchronized firing.

THE-3: Random Code Neural Networks

Investigator: Alexander Jourjine, Analog Intelligence Corp., Cambridge MA 617/524-4858

Problem Area: Develop a hardware oriented neural network theory, based on distributed binary micro-states which correspond macroscopically (i.e., for longer-term time averages) to analog state variables of the network. Implement learning as minimization (at each node of the network) of a disorder functional dependent on the correlations within input and internal signals of the node.

Prior Approach: N/A

Neural Network Approach: This novel approach considers elemental neurons which receive binary pulses as address and information carriers simultaneously via two channels: a direct channel and a modulated channel (modulated subject to a certain update rule); pulses are transmitted only via the modulated channel: thus the generated microstates are a function of average correlations between these microstates and the specific update rule for the modulation. Depending on the rule, the macroscopic behavior (long time average) of the system reduces to several common neural network models. In effect, the model develops its own 'preprocessor' to recognize the deviations from randomness inherent in the data without external supervision – learning and recognition happen simultaneously.

Status: On-going research project.

Results: Analysis of the theory is extensive; small-scale simulations have been run and analytic results verified.

Future Plans: Tie this work directly to applications.

Comments: Unusual and innovative work.

Conclusions: A novel model for unsupervised learning has been developed. The neural network operates as a collection of binary devices which all contribute to represent the operational states of the network as analog values; for N neurons only order(N) interconnects are needed; only local synchronization of units is needed.

THE-4: Image and Object Coding for Data Compression

Investigator: Albert F. Lawrence, Hughes Aircraft, CA

Problem Area: Image analysis, signal processing.

Prior Approach: These mainly involve decomposition of the data into orthogonal series such as: (1) Fourier series. Image data is written as Fourier series. Terms whose coefficients do not exceed a pre-determined level are neglected. (2) Karhunen-Loève transforms. The statistical properties of the data are used to determine a series similar to the principle components. Terms of small norm are neglected. Other approaches have exploited the nature of the digitized data as in run-length encoding or, more generally, recording only the differences in succeeding terms in a more or less continuous series.

Neural Network Approach: Barnsley and his co-workers at Georgia Tech have shown that data can be coded by systems of affine transforms at compression ratios of 10 000 to 1 or better. The determination of the most efficient representation, however, is an enormous computational problem. This problem can be reduced to an explicit optimization problem. Such optimization problems can be solved in neural network machines by standard methods which have been reported in the literature. Furthermore, one may emulate the data coding properties of retinal interconnections and retina to visual cortex mappings found in nature. These yield interconnect architectures which give simple realizations of affine transformations.

Status: Effort is in the early stages of mathematical analysis. Some developmental work has been performed toward the specification of architectures.

Results: Networks embodying exponential or logarithmic point-to-point mappings have been shown to yield simple schemes for the calculation of affine transformations on two-dimensional data sets. Reduction of the dimensionality of the optimization problem to a more manageable level by use of the method of moments has been shown.

Future Plans: The dimensionality of the problem is still an obstacle. Methods for reducing the degree of connectivity required, particularly through hierarchical interconnect schemes seem to constitute the most promising approach at present.

Comments: One of the major obstructions to further progress in neural network theory is the lack of a detailed theory of the dynamics of nonlinear systems, particularly as relates to computation-like processes. Some information from neurophysiology might give some useful hints in this area. Another gap is in the device technology available to the researcher. Development of devices, such as Josephson junctions, which operate at the quantum level might prove useful here.

Conclusions: The work here is preliminary. Although neural networks seem to afford a convenient solution for the interconnect problem, more mathematical analysis is necessary to reduce the scale for implementation in present-day technology.

THE-5: Generalities about Neural Networks

Investigator: Robert Rosen, Dalhouse U., Halifax NS 716/473-4055

Problem Area: A historical and conceptual overview of neural networks and how they relate to other fields.

General Points:

1. Neural networks are transducers from afferent to efferent.
2. Neural networks are pattern recognizers and classifiers.
3. Neural networks are pattern generators.
4. Neural networks are realizable in other contexts (operon networks, Ising models).
5. Neural networks are generalizable from dynamical systems (which are simple systems).
6. Neural networks are “informational” networks (complex systems) which are infinitely open and cannot be described by a syntactic description.

Comments: For a more complete account, see “Organisms as Causal Systems which are not Mechanisms: An Essay into the Nature of Complexity,” in *Theoretical Biology and Complexity*, R. Rosen, ed., Academic Press, New York 1985, p. 165–203.

Conclusions: Complex systems are “open” in the sense that a complete syntactic description cannot be given. Furthermore, complex systems do not allow a distinction between hardware and software. The fundamental tenet is that living systems in biology present vivid examples of complex systems, as opposed to simple systems; only the latter can be fully defined by a formal description or algorithm.

SUR-1: Neural Network Overview on the ESPRIT-II Program

Investigator: Bernard Angeniol, Thomson CSF, France

Problem Area: Multi-national effort to develop and apply neural network technology primarily for image processing and speech recognition.

Neural Network Approach: Both hardware and software development of a neural network simulator and standard language is planned.

Status: Phase I funding of about M\$ 15 is expected during spring of 1988 with about 70% probability.

Results: Various demonstrations have been completed; specifically, a solution to the 'Traveling Salesman Problem' was highlighted using a newly developed form of the self-organizing map for a 1 000-city tour.

Future Plans: Phase II funding is planned at about M\$ 45.

Comments: Work is to be performed by engineering departments of universities and industry.

Conclusions: Program with a broad scope; specific program goals are hardware-oriented.

SUR-2: SDI Discrimination Problem

Investigator: Mike Cain, Booz-Allen, Washington, DC 703/769-7782

Problem Area: Overview on multi-sensors (passive, active, interactive) for specific target identification. Since signatures change as a function of time adaptive processing techniques are needed; where could neural networks be useful?

Approach: AI most suitable for data fusion and clustering.

Neural Network Approach: Neural networks potentially applicable for optimization (simulated annealing), coordinate transformations (perceptron), and feature detection (Grossberg).

SUR-3: Neural Network Applications at Hughes Ground Systems Group

Investigator: Patrick F. Castelaz, Hughes Command and Control Division, Fullerton CA 714/732-8622

Problem Area: Neural networks are suitable for optimization and learning problems; Dr. Castelaz presented a list of 12 small-scale demonstrations which show promise for future system applications. Here, these problems will be summarized briefly:

Problem 1: De-ghosting (both digital (d) and analog (a)) – radar triangulation from several emitters without respective range information will give false target locations due to beam overlap.

Prior Approach: 36 actual targets result in 4 070 ghosts; standard linear programming techniques would require long solution times (years!).

Neural Network Approach: (Class: optimization) Special-design relaxation network with cost function such that overlap is minimized.

Status: Simulation results (d: design checked, a: qualitative).

Results: Neural network settles in milliseconds on a solution: within 1-10 mile radius 55%–100% of potential targets correctly identified.

Comments: Approach is tested only to 40 targets.

Problem 2: Plot/Track Correlation (d1, a) — Sort out conflicting information on multiple tracks.

Neural Network Approach: (Class: optimization) Relaxation processor.

Status: Simulation results (d1: design checked, a: qualitative).

Results: Tested to 16 tracks.

Problem 3: Plot/Track Correlation (d2)

Neural Network Approach: (Class: analogue) Use a cellular automata to construct an analogue micromodel of problem.

Status: Proof-of-concept hardware.

Problem 4: Analog Coordinate Transform — Build a hardware table look-up.

Neural Network Approach: (Class: analogue) use a cellular automata to perform, e.g., a $(x, y) \iff (\theta, r)$ transformation.

Status: Proof-of-concept hardware.

Comments: Expect submicrosecond transform, versus best (CORDIC) response of 20-40 μ s.

SUR-3 — continued

Problem 5: Infrared Tank Discrimination and Centroid Finding – Smart shell: during half-rotation ($1 \mu\text{s}$) discriminate and locate target; 16×16 single-bit data.

Neural Network Approach: Use the temporal dynamics of a cellular- automata-type thinking algorithm, gated at fixed time delay (for a given height above target), to reduce to a single pixel (the centroid), if, and only if, a target was present in the initial picture.

Status: Simulation results and VLSI chip mask.

Problem 6: 1-D Signal Classification of a General, Arbitrary Pulse Shape

Status: Qualitative simulation results.

Problem 7: 1-D Signal Classification of a Continuous-Wave Sinusoid -- Detection and classification of two signals within a waveform.

Neural Network Approach: 32 neurons, 32 examples of each class presented $300 \times$.

Status: Quantitative simulation results.

Results: 80–90% correct classification.

Problem 8: Infrared Point Source Discrimination and Multi-target Tracking — (class: one-dimensional and two-dimensional pattern recognition and optimization)

Status: Contract with USASDC since October 1987.

Future Plans: Follow-on proof-of-concept hardware.

Problem 9: Speaker ID from Speech — (class: 1-D signal class.) Speaker identification using time-amplitude waveform.

Neural Network Approach: Extended backpropagation.

Status:

Results: Quantitative simulation results.

two speakers could be reliably distinguished by pronouncing the word “Hughes”.

SUR-3 — continued

Problem 10: Doppler Radar Target Classification

Problem Area:	(class: two-dimensional signal class.)
Neural Network Approach:	Tried a (16, 16, 16) backpropagation network successfully. 'Time' encoded into signal as additional channel (two-dimensional representation); 48-neuron network trained on 32 exemplars from each of four classes.
Status:	Quantitative simulation results.
Results:	92% correct classification.
Comments:	One of the applications in maturest development state.

Problem 11: Preferential Defense — (class: n-D pattern class.) Fire control problem.

Neural Network Approach:	Backpropagation network: 22-neuron network, four-neuron output.
Status:	Quantitative simulation results.
Results:	Achieved 92% intercept rate: single threat, single target, multiple (fixed) weapons.
Comments:	One of the applications in maturest development state; video demonstration.

Problem 12: Backpropagation Implementation Issues — What are the component tolerances needed to achieve satisfactory performance from a feedforward network?

Neural Network Approach:	Backpropagation networks.
Status:	Quantitative simulation results.
Results:	Resistor values of 10% accuracy are OK, graceful degradation beyond 10%.
Comments:	One of the applications in maturest development state.

SUR-3 — continued

General Comments: Dr. Castelaz indicates that in the present state of development all these applications are of small size; how to scale up to realistic systems is not always clear; one possibility might be to segment the complete problem into chunks small enough for processing by dedicated small-scale hardware.

Conclusions: Multiple applications of optimization and learning/pattern recognition networks to real problems have been addressed. Both quantitative and qualitative results have been reported. Representation and implementation issues have also been addressed. Current research is very focused on scaling and implementation issues. Extensions of backpropagation approaches to decrease size are being investigated.

SUR-4: Modern Data Storage and Retrieval Systems

Investigator: Ching-chieh Chen, Simmons College, Boston MA 617/738-2224

Problem Area: Large-scale information processing in centralized databases has the problem of most-suitable database selection for a given inquiry; present so-called “gateway programs” are being developed based on simple AI-like decision systems. Large databases, lexical and pictorial, can now be stored and distributed on optical disk; a current example is a detailed archeological record related to the First Emperor of China, specifically the terracotta figures of warriors and horses at Xian, including high-resolution digital images. To use this type of information effectively, efficient feature extraction algorithms need to be developed; currently, these algorithms are rule-based – but neural network approaches may be more suitable.

Status: On-going research program.

Future Plans: More work will be devoted to electronic digital imaging and the computer visual input and output in terms of retrieval.

Conclusions: The dynamic development of various hybrid new technologies (such as optical and microcomputer) have great potential for large-scale multi-media data and information storage, retrieval, and utilization. It creates new exciting research areas for neural network approaches as well.

SUR-5: Venture Capital Financing of a Neural Network Company

Investigator: Oliver D. Curme, Battery Ventures, L.P., Boston MA 617/367-1011

Problem Area: How will venture capital support neural network startup companies?

Status: Battery Ventures currently supports Hecht-Nielsen Corp. (HNC), San Diego, CA.

Future Plans: Battery Ventures does not finance research; a company must have a demonstratable product in an identifiable niche position in order to attract financial support. Financing is primarily intended for the purpose of broad-based marketing of products with short-term profit prospects.

Comments: Break-even for Battery Ventures is about one in three commitments; payoff is expected within five years.

Conclusions: With respect to the neural network field, Mr. Curme shares an outlook very consistent with Robert Hecht-Nielsen of HNC: substantial applications of neural network in many areas are imminent; short-term profit potential exists in marketing neural network simulation tools and know-how.

SUR-6: Defense Applications of Neurocomputers

Interview: Robert Hecht-Nielsen, HNC, San Diego CA 619/546-8877

General Comments:

1. Neural network technology is a weak, embryonic field; over-promising, 'hype', and anthropomorphism need to be avoided.
2. Neural network technology should be biology-inspired, but not constrained to model nature explicitly.
3. HNC recognizes that "neural networks are a solution looking for a problem" and markets the expertise and hardware to train domain experts in neural network methods and capabilities in the hope of broadening the applications potential.
4. At present, there are no HNC-installed, commercial neural network applications; a promising financial application, presently under development by HNC, is expected to achieve market maturity in two-to-five years.
5. DOD applications are more difficult: "DARPA should commit to long-term funding." Initial demonstrations should be modest, contractor-proposed, and closely refereed. For military systems, development tools are needed – i.e., standard languages and standard graphics interfaces.

Problem 1: Doppler Radar Return Classifier — classify targets by internal motion-induced chirp signatures imposed on a quasi-cw radar pulse; dynamic time-warping and matched filter operation is needed.

Prior Approach:	De-modulate the signal to the audio band for human interpretation.
Neural Network Approach:	Chop 0.5-s pulses into 40 slices, compute the spectrum for each slice, and propagate through an avalanche network; parallel matched filters sort out the target classes.
Status:	Conceptual stage.
Future Plans:	TRW's Mark-IV could implement 10 000 matched filters running almost in real time. With a small group (four people for about one year), one could build an operational demonstration with perhaps 20 targets.
Comments:	Ambiguities exist due to velocity and angle of target; also, such a system would have to deal with battlefield noise.

SUR-6 — continued

Problem 2: Signal-from-Signal Extraction (Cocktail Party Problem) – pick and track a specific signal wave form among many: “How to focus attention?”

Prior Approach: FM-carrier tracking.

Neural Network Approach: Fukushima’s neocognitron and Grossberg’s ART solve somewhat similar problems in the visual domain; representation is a sliced and Fourier-transformed spatio-temporal version of the input signal (“atomic spatio-temporal sequences”); there must be variable-gain stages which are set by the system at a higher level.

Status: No clear concepts developed.

Comments: It may be difficult to “tune-up” such a system initially.

Other Problems:

- Sonar target identification,
- Image processing, and
- Noise reduction.

SUR-7: Laser Radar Data Survey

Investigator: Jim Leonard, WPAFB, Dayton, OH

Problem Area: CO₂ laser radar with Doppler processing, as well as infrared and visible information require sensor fusion; such systems allow feature detection, correction, and identification; they give three- dimensional information directly and show promise for model matching.

Conclusions: Powerful vision algorithms are needed to render all the information inherent in such multi-sensors useful.

SUR-8: Future Command and Control Systems

Investigator: Orin E. Marvel, Hughes Aircraft, Fullerton CA 714/732-5869

Problem Area: BMC³ (battle management, command, control, and communication) is a very complex problem, constrained in addition by such requirements as functioning with existing NATO equipment. Duties are roughly partitioned as C³ for humans and BM for computers. The key to a war in the future are: short time lines, very high speeds, large data flow, and a large number of simultaneous engagements.

Prior Approach: Standard algorithms are fast enough to perform decisions; most are done manually using O.R. techniques.

Neural Network Approach: Required optimization problems (correlation, strategy, assignments) are too large for current state-of-the- art technology and may be solvable by neural network approaches.

Future Plans:

1. Resource optimization;
2. Weapon allocation;
3. Battle Management Associate;
4. Intelligence Processing.

Conclusions: Requirements suggest that a large automated engagement like SDI cannot be accomplished without neural networks.

SUR-9: Custom Analog VLSI Chip Development for Vision and Speech

Investigator: Carver Mead, CalTech, Pasadena CA 818/356-6568

Problem Area: Special-purpose analog chips that mimic closely processing functions present in a biological retina or cochlea are under development; such chips are deemed essential components of an autonomous vision or speech recognition system.

Approach: The central tenet of Dr. Mead's design philosophy is to merge knowledge about the biological system (and a full appreciation of its detailed structure and function) with an understanding of how to implement the essential features in analog hardware (emulating the biological with equivalent electronic functions).

Status: On-going project with about k\$400 per year.

Results: See "A Silicon Model of Early Visual Processing", *Neural Networks*, 1, Nr.1, 91-99 (1988).

Future Plans: Work on a vision segmentation chip (in collaboration with C. Koch) may require another five years of development; in 10 years a chip set for a complete vision system may have been developed.

Comments: MOSIS is being used for all chip fabrication; DARPA should be congratulated for its work in establishing MOSIS.

SUR-10: Neural Network Applications at TRW

Investigator: Mike Myers, Robert Kuczewski, TRW, San Diego CA 619/592-3381

Problem Area: The dedicated team at TRW has developed a flexible and powerful neural network simulator and has addressed a wide range of potential application areas; a brief outline of these applications follows:

Problem 1: Airplane Identification

Neural Network Approach: Present raw spatial image to neural network. Use neural network to center, take polar and Fourier transforms and classify using a three-layer trained network.

Results: Simulations for three different airplane shapes completed.

Future Plans: Expand to classify all aspect angles, use for trajectory extrapolations.

Conclusions: Needs considerable more work to use in real systems, but the basic approach has been shown to work well.

Problem 2: Edge Detection

Neural Network Approach: On-center/off-surround two-dimensional differentiation.

Status: Used same approach for temporal image change detection.

Problem 3: Recognize Doppler Modulation of Radar Returns from Helicopters

Prior Approach: Directly demodulate Radar returns to audio for recognition by human operators.

Neural Network Approach: 2 500 neuron spectrogram map + 50-neuron Fourier transform + 100-neuron classifier + 1 000-neuron image outstar.

Status: Demo complete.

Results: Eight helicopters distinguished from simulated data.

Future Plans: Results will be applied in smart weapons programs.

SUR-10 — continued

Problem 4: Threat Maneuver Detection — Recognize specific time sequences of airplane banking angles to anticipate, e.g., evasive maneuvers; important for “smart” missile guidance.

Prior Approach: None.
Neural Network Approach: Spatio-temporal correlation of detected aspect angles using prior neural network.
Status: Small demo built and working.
Results: Not tested extensively.
Future Plans: Will pursue if funding source exists.
Comments: Build from spatio-temporal information a model to predict future actions; important for dynamic system modeling.

Problem 5: On-Line Hand-Written Character Recognition

Neural Network Approach: Four-dimensional input (x, y, \dot{x}, \dot{y}) , train BEP.
Results: Works on characters independent of size, needs more training examples.
Conclusions: This demo was completed in one morning, and showed the power of the emulator design tool MARK III-1.

SUR-10 — continued

Problem 6: Radar Emitter Identification from Single Pulses – Identify radar emitter from characteristic micro-features present in the pulses; use real pulse samples from three different sources: two emitters, one recorded six months apart.

Prior Approach: Classified.

Neural Network Approach: Fully-connected three-layer network with (500, 16, three) cells; out of a 1 000-pulse data set, 200 pulses were presented $5\times$ as training set.

Results: 98% correct identification between the two different emitters; 90% correct between same emitter recorded at different times.

Future Plans: This will be pursued on IR&D.

Comments: While an impressive demonstration, it is not clear if the simple backpropagation neural networks could achieve equally good results with a more realistic number of possible emitters.

Problem 7: On-Line Threat Assessment — neural networks applied to cognitive processing: build up an internal “world model” for threat assessment and effective force deployment.

Neural Network Approach: Threats and targets are modeled as repulsive and attractive force fields, respectively; the optimal path sequence follows from the resulting equation of motion.

Status: On-going program sponsored by AFWAL AAAL.

Results: Final demo given 1/88.

Future Plans: Use as “inner model” for threat response evaluation and planning.

SUR-10 — continued

Problem 8: Backward-Error Propagation as a Self-organizational Structure — Apply back-propagation to an unsupervised learning situation where the input pattern is equated to the output pattern with the goal to achieve stable hidden layers of significantly lower dimensionality.

Neural Network	Five-layer network with (255, 16, three, 16, 255) nodes.
Approach:	
Status:	Network was trained and tested on 255-dimension image data.
Results:	Clustered in three dimensions.
Future Plans:	Bandwidth reduction.

SUR-11: Artificial Adaptive Neural Network Systems

Investigator: Tom Ryan, SAIC, Tucson AZ

Problem Area: Description of several current projects which can be simulated with the SAIC-developed GINNI software simulator and also Connection Machine applications.

Prior Approach: Varies with project.

Neural Network Approach:

1. In an AI-like system use, the ART I model as a content-addressable memory.
2. Development of a hierarchical model for machine vision.
3. Use of a Kohonen-type neural network for vector quantization of visual information.
4. Application of a multi-layer backpropagation networks to image processing; primary appeal is that the system can be taught by data – explicit knowledge engineering is not needed.
5. The following four applications use a Connection Machine CM-1:
 - (a) Geman/Geman-type processing for image segmentation;
 - (b) Cellular automata for broad area search with 5×5 or 7×7 pixel window;
 - (c) Hopfield-type associative memory for feature detection;
 - (d) Parallel VQ training.

Status: On-going programs.

Future Plans: Focus on machine vision applications.

Conclusions: Neural network technology can be applied directly in low-level signal processing/pattern recognition applications. More work is needed on complexity, scaling and invariance issues. We should take as many cues as possible from the biological studies. The application of neural networks to symbolic processing is on shaky ground but deserves adequate funding. Knowledge/facts representation is a major concern.

SUR-12: Cellular Automata as Neural Networks: Pattern Recognition and Tracking

Investigator: Tommaso Toffoli, MIT, Cambridge MA 617/253-3194

Problem Area: How to segment texture in an image.

Prior Approach: Computer vision techniques developed in AI.

Neural Network Approach: Use a cellular automata network (CAN) to find the large-scale macrofeatures emerging from specific microscopic update rules; whenever these large-scale features correspond to similar features in the image, the CAN can find them in noisy input images.

Status: A dedicated CAN-board for the IBM PC has been developed and is commercially available with fast realtime video update.

Results: Several kinds of macrofeatures emerging from micro-rules were demonstrated; while some specific results exist, no complete and general theory exists to predict the full relationship between macro- features and micro-rules and vice versa.

Future Plans: A high performance CAN (512 modules, each with 16 planes of 1024×1024 points) is in development.

Conclusions: Whenever a process is fully described by local interactions, it can be efficiently simulated using a CAN.

SUR-13: Applications of Neural Networks: Review of Honeywell's Activities

Speaker: Jim White, Honeywell Systems Research Center, Minneapolis
MN 612/782-7355

Problem Area: Honeywell has taken a careful look at neural network technology from both a software (algorithmic) and a hardware perspective. An integration of neural networks, expert systems, and numeric processors is needed.

Problem 1: Feature Learning in Simple Sentences

Neural Network Multi-layer backpropagation networks.

Approach:

Results: Network learned certain domain concepts; generalization was observed; shortcomings were identified; and alternatives suggested.

Problem 2: Genetic Classifier System for Acoustic Signal Recognition

Approach: A rule-based system for recognizing sequences of acoustic segments.

Results: Simplified problems were handled successfully using a credit assignment algorithm.

Problem 3: Semantic Control of Neural Network Associative Search

Neural Network Proprietary mechanism for establishing an externally supplied, context-sensitive control of associative search.

Results: Basic operation demonstrated.

Problem 4: Digital Associative Content-addressed Memory

Neural Network Off-the-shelf RAM combined with custom ICs to implement a high-speed content addressable memory.

Results: Design analysis and simulations completed.

Problem 5: Neural Network Coprocessor for Hybrid Computing System

Neural Network Neural cluster concept and basic architecture has been defined.

Approach:

Results: Key data and control representation issues were identified.

SUR-13 — continued

Future Plans: Honeywell sees a symbiosis of conventional and neural network systems: neural networks of different types will cooperate in solving a problem from different aspects.

Comments: Honeywell knows how to reduce some of its neural network work to silicon, but believes it is, currently, premature.

Conclusions: Neural networks are useful for front end applications to process raw data; neural networks will also be useful as back ends for generalization to reduce the rule-explosion problem. For successful neural network applications, the right data and knowledge representations are crucial. Algorithmic and symbolic processing elements will be combined in future systems.

L.3 QUESTIONNAIRE TO FORECAST NEURAL NETWORK APPLICATIONS DURING THE NEXT QUARTER CENTURY

L.3.1 Introduction

One of the major goals of the Neural Network Study was to identify application areas to which a national neural network program could make meaningful contributions. For concreteness, a time interval of 25 years was chosen. As part of this effort, a questionnaire was designed to show possible options and alternatives, referenced to such questions as “What will be?” and “What is possible?” Special attention was given to identifying the technology barriers and the required breakthroughs.

For purposes of designing the questionnaire, it was necessary to standardize the questions that would be asked of various experts so that the forecasts would have a common foundation. The questions were grouped into two main categories:

- “**What will be?**” was asked as an alternative to such questions as “What is probable?” or “What is 80% probable?” etc. Implicit in the question “What will be?” is a very high likelihood that the capability will indeed be achieved.
- “**What is possible?**” was asked as a question generally intended to be unconstrained by consideration of funding limitations. Having determined “what is possible,” the expert was asked to identify the controlling factors.

Next, it was desirable to quantify, using a histogram, the responses so that the results of the questioning could be easily aggregated and summarized. It was assumed that there would be about 25 to 50 expert respondents; there were, in fact, 40 responses. This suggests that a histogram should have about five categories as a balance between resolution and accuracy.

All the inputs to this questionnaire were used in some fashion. It is only natural that conflicting forecasts would be made and, for that reason – as well as the need to aggregate many of the forecasts to a higher level – final accountability for the actual forecasts rests not with the questionnaire’s respondents but with the Neural Network Study’s *System Applications Panel*. For reference, the raw histograms are given to enable independent conclusions to be drawn. Some respondents answered more than once per question or not at all. Multiple answers were counted equally. For these reasons the total number of responses varies from question to question.

L.3.2 Results

What Will Be?

With \$10M to \$100M per year (63%) (Figure L-1) for five years (55%) (Figure L-2), neural networks are promising for generating new applications (88%) (Figure L-3) and (Figure L-4). The applications will be in robotics (43%) and man-machine interfaces (43%) (Figure L-5) primarily in classification/recognition (49%), sensors (21%), and motion control (18%) (Figure L-6). These applications will exhibit increased information throughput (33%) and quicker responses (27%) – i.e., be faster systems (Figure L-7). Moreover, about one half of the neural network applications are expected to replace existing technologies (Figure L-8). There is a significant civilian fallout to neural network work (55%) (Figure L-9), with the biggest challenge being product development (65%) – not manufacturing, marketing or competition (Figure L-10). Finally, neural network products are expected to create social changes (65%) (Figure L-11).

What is Possible?

There are fundamental limits on neural network technology remaining to be discovered (82%) (Figure L-12). The main problem areas are a better mathematical theory of neural networks (42%) and design rules for complex neural networks (27%); validating a theory or transferring known theories to practice do not appear to be as important (Figure L-13). There may be a psychology developed for neural networks (42%) (Figure L-14), but its application to human psychology in the foreseeable future is debatable (Figure L-15). In any case, neural networks are not expected to approach the capability of the human mind except for specialized functions (66%) (Figure L-16). Finally, on a purely speculative note, conjectured phenomena of the human mind, such as psychokinesis, are very unlikely (85%) (Figure L-17).

**AT THIS TIME AN APPROPRIATE LEVEL OF
GOVERNMENT SUPPORT PER YEAR TO NN APPLICATIONS IS:**

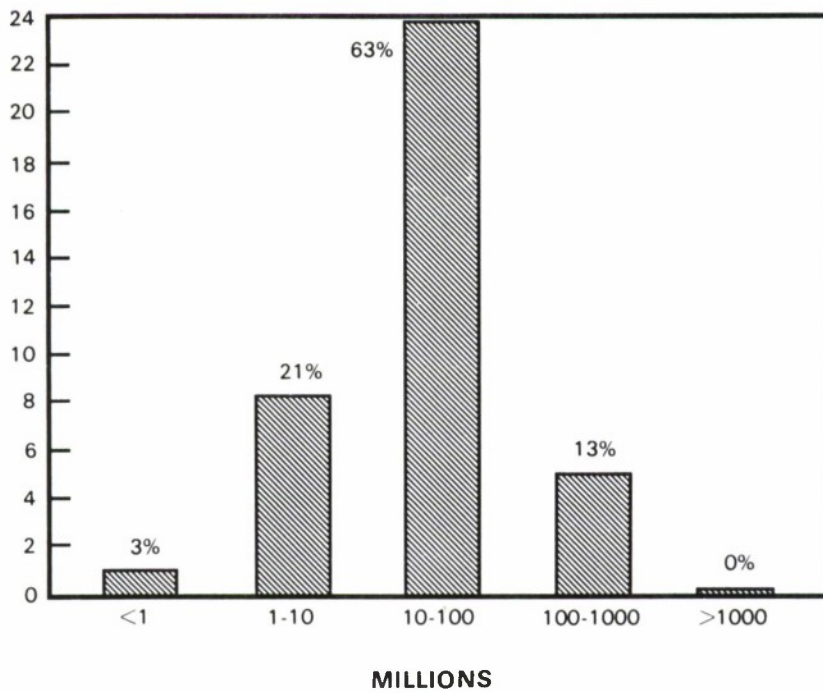


Figure L-1. Level of Support Appropriate for Neural Network Applications Research.

NN WILL BE MAJOR TECHNOLOGY IN:

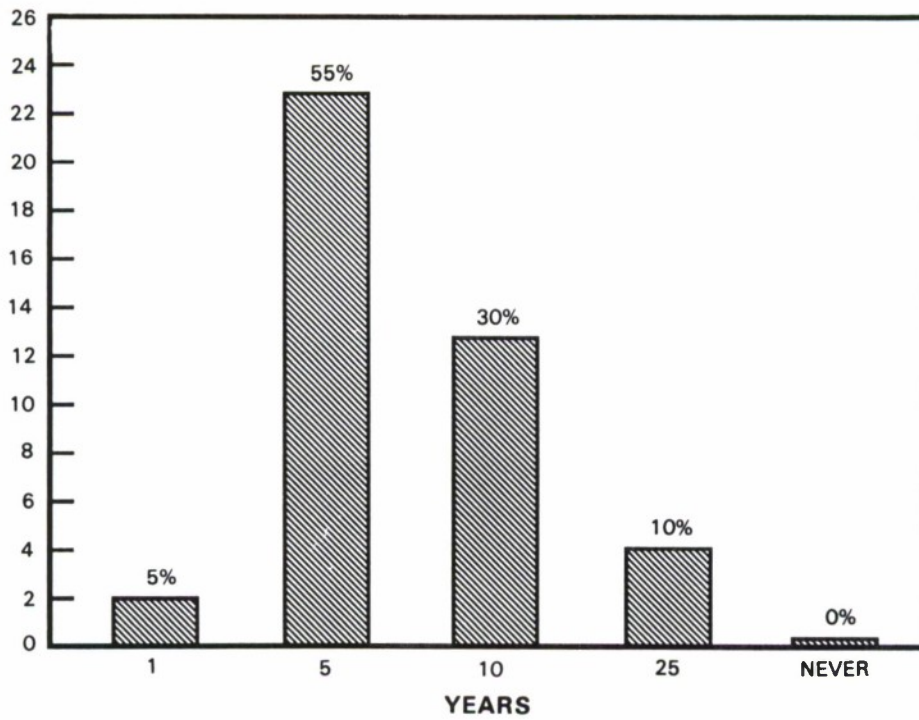


Figure L-2. Development Time of Neural Networks as a Major Technology.

102972-18

102972-17

NN IS VERY PROMISING FOR GENERATING NEW APPLICATIONS

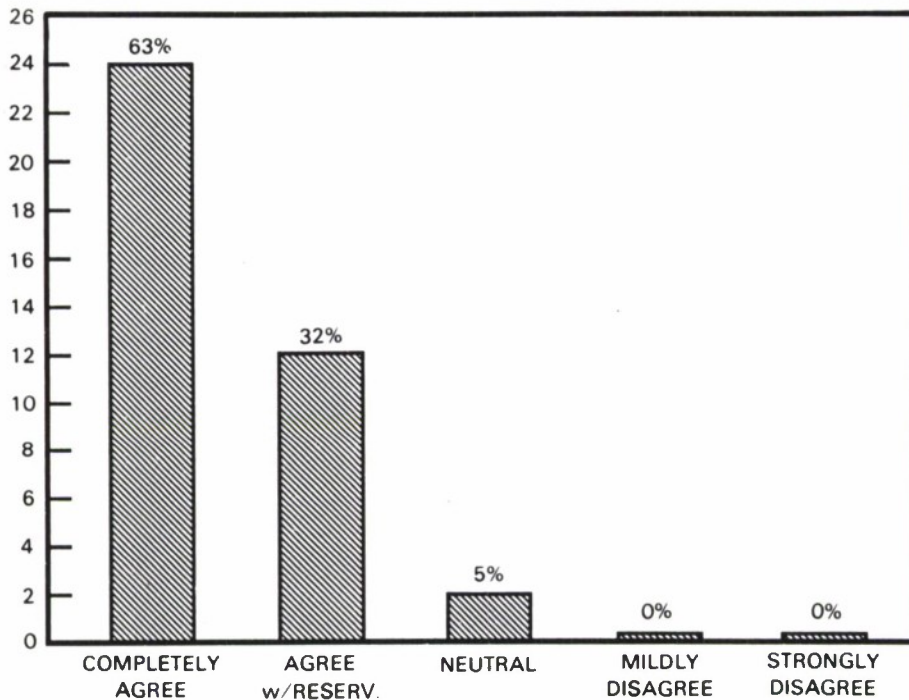


Figure L-3. Potential of Neural Networks for Novel Applications.

THE MOST PROMISING NN APPLICATIONS ARE IN:

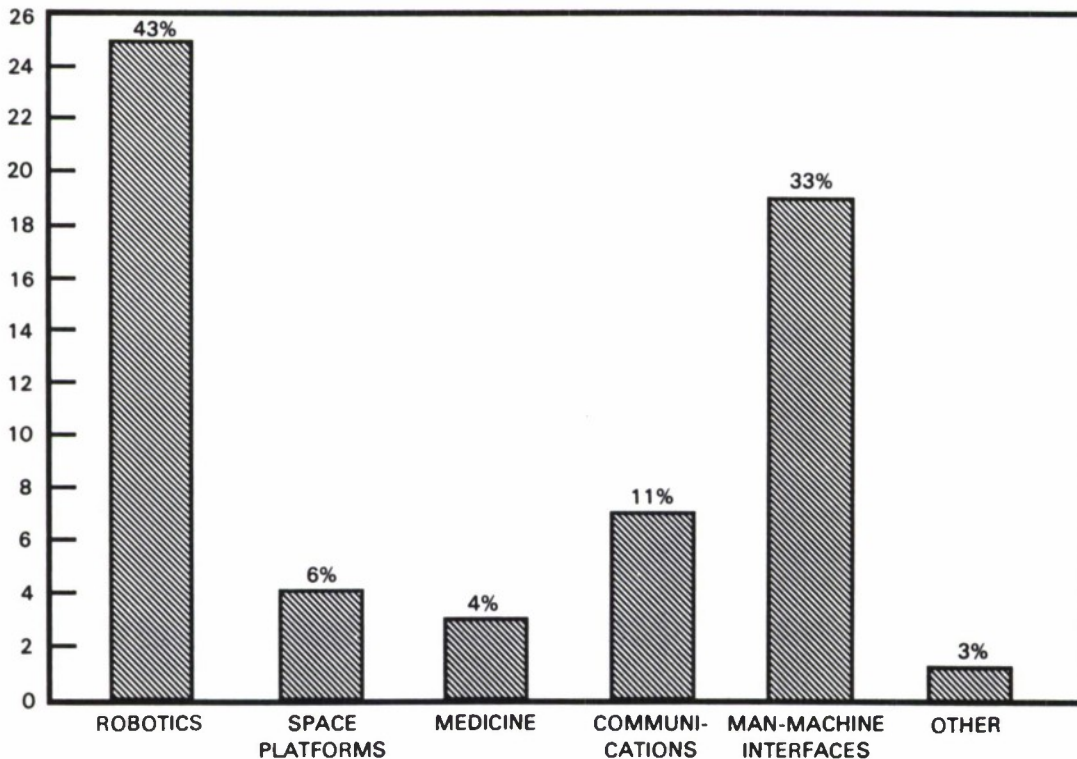


Figure L-4. Potential of Neural Networks for Enhanced System Performance.

**THERE ARE SEVERAL APPLICATION AREAS
WHERE NN COULD PRODUCE A BREAKTHROUGH**

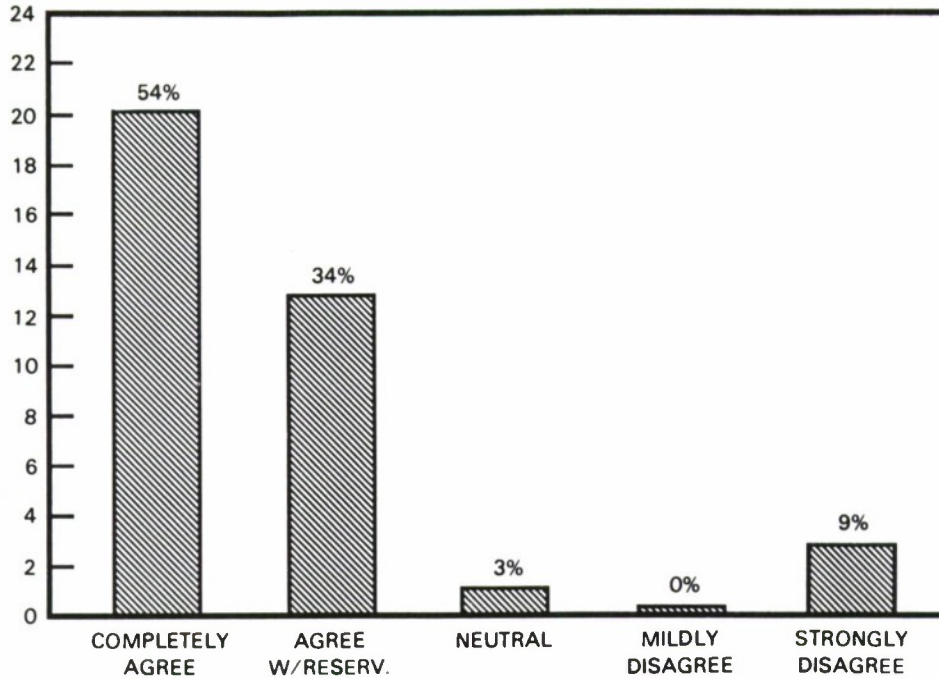


Figure L-5. Promising Application Areas for Neural Networks.

**THE SUBSYSTEM AREA WHICH OFFERS THE BEST PROMISE
FOR NN IMPACT IN THE NEXT 25 YEARS IS:**

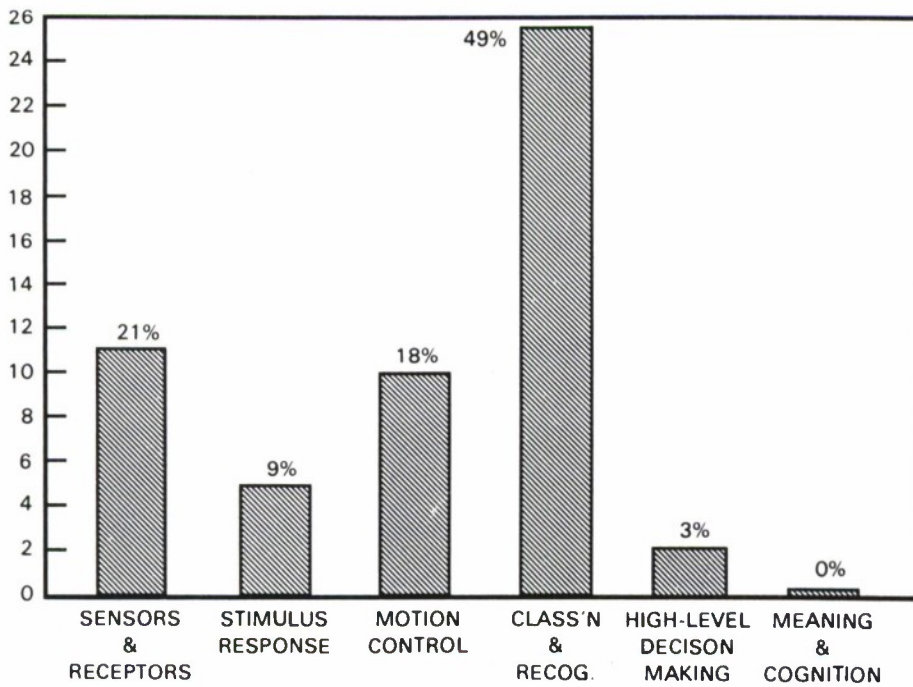


Figure L-6. Promising Application Areas for Neural Network Subsystems.

**NN APPLICATIONS TO IMPROVE PERFORMANCE
ARE MOST PROMISING FOR:**

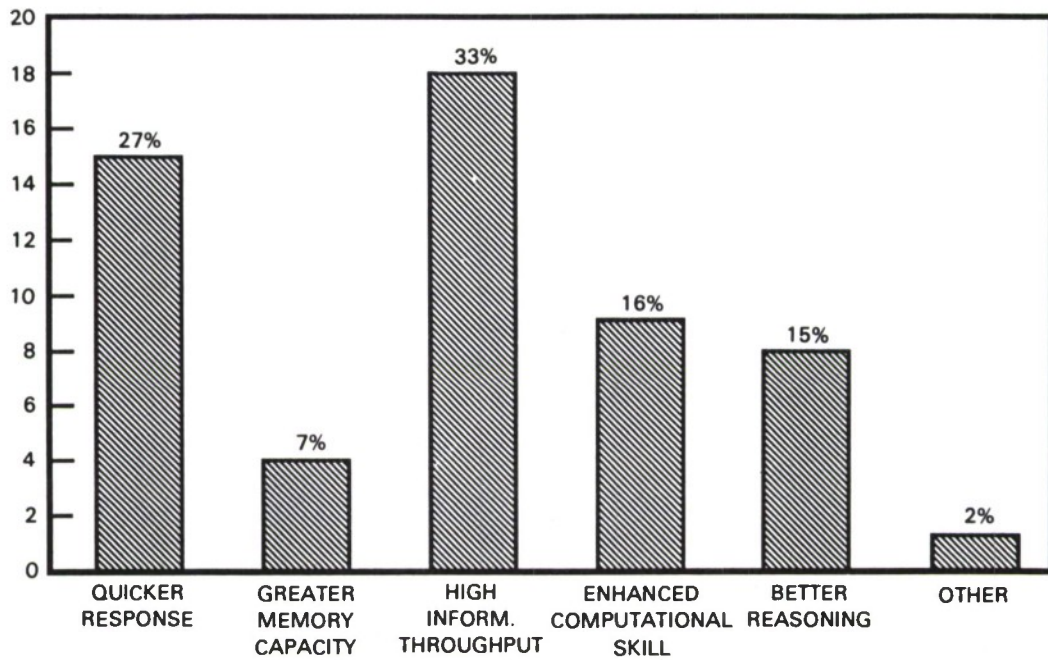


Figure L-7. Performance Advantages of Neural Networks.

NN IS VERY PROMISING FOR REPLACING EXISTING TECHNOLOGY

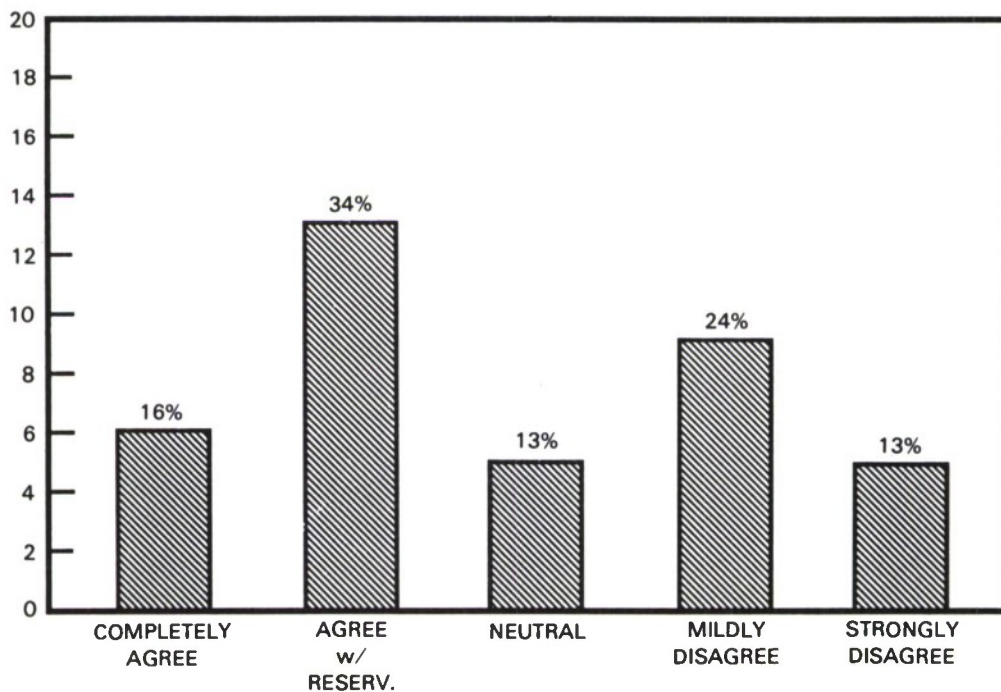


Figure L-8. Potential for Replacing Existing Technology.

102972-12

102972-11

**THE MOST PROMISING SECTOR FOR
NN APPLICATIONS IN THE NEXT 25 YEARS IS:**

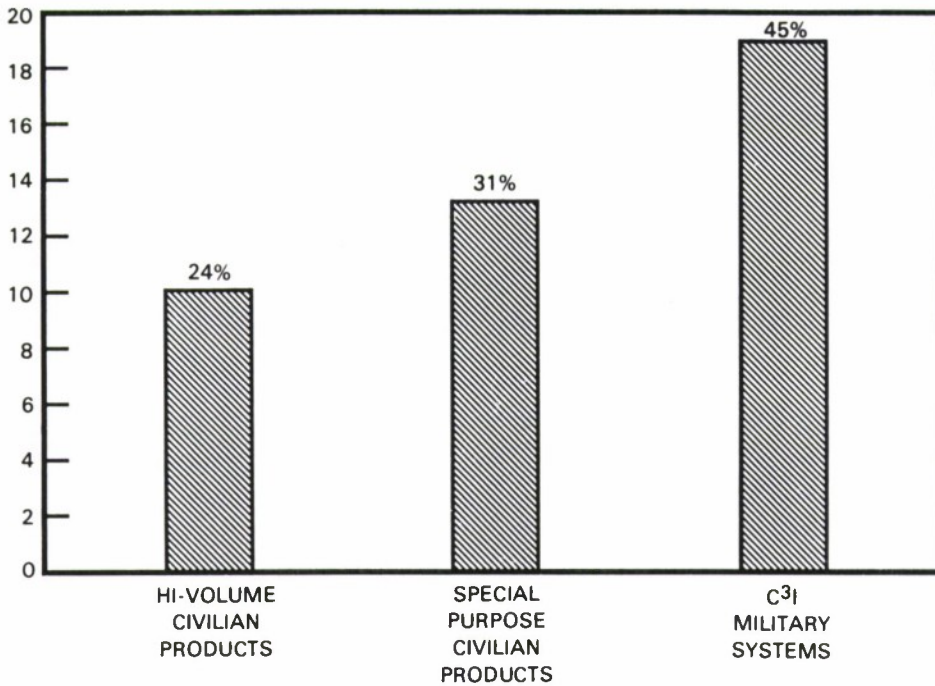


Figure L-9. Share of Civilian versus Military Applications.

THE BIGGEST CHALLENGE TO NN APPLICATIONS IS:

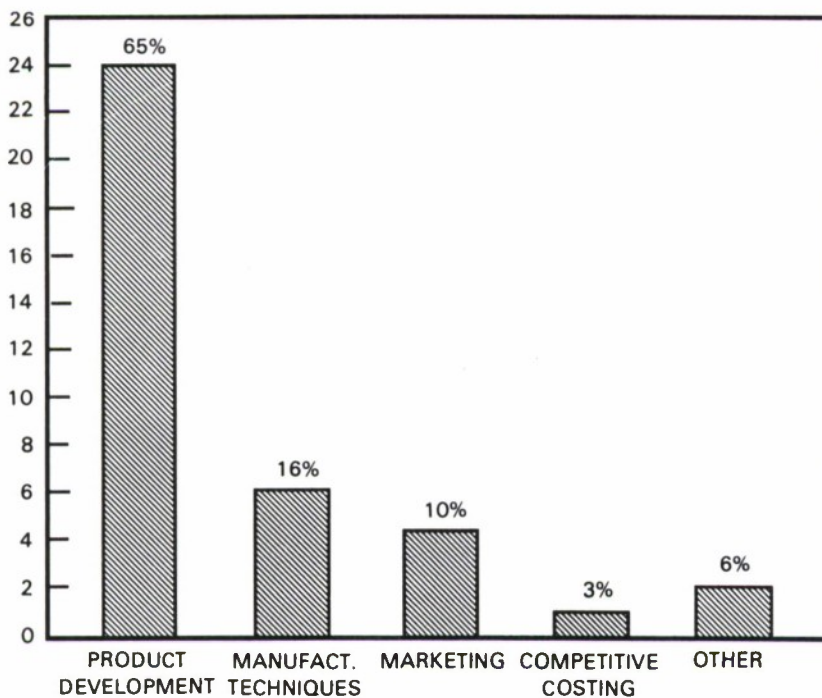


Figure L-10. Application Challenges.

THERE WILL BE SOME CRITICAL SOCIAL IMPLICATIONS IN NN APPLICATIONS IN THE NEXT 25 YEARS

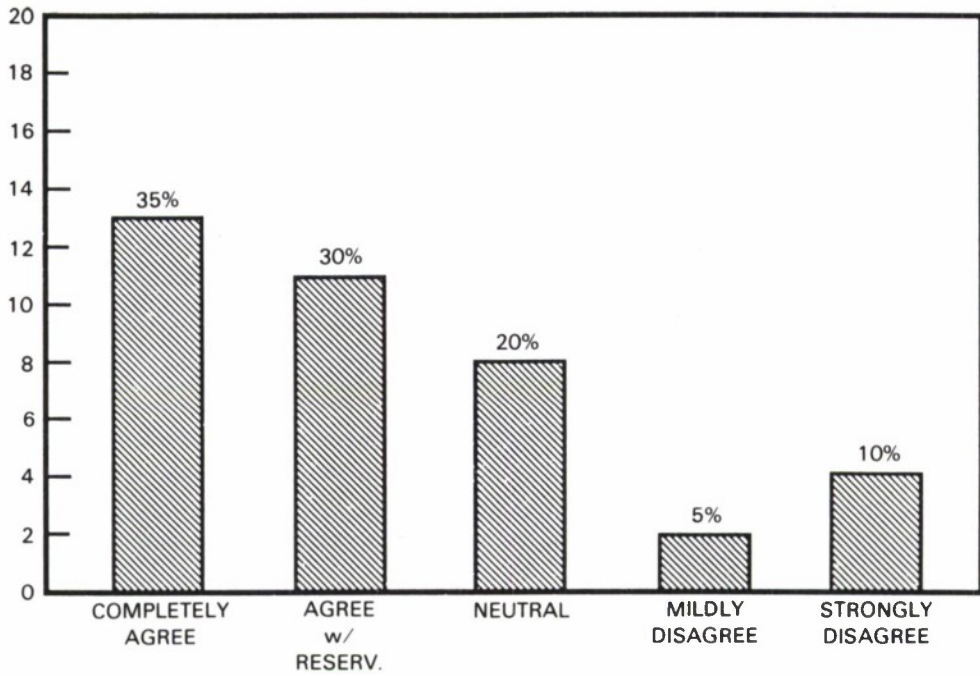


Figure L-11. Potential Social Impact.

ALL THE FUNDAMENTAL LIMITS ON NN TECHNOLOGY ARE ALREADY KNOWN

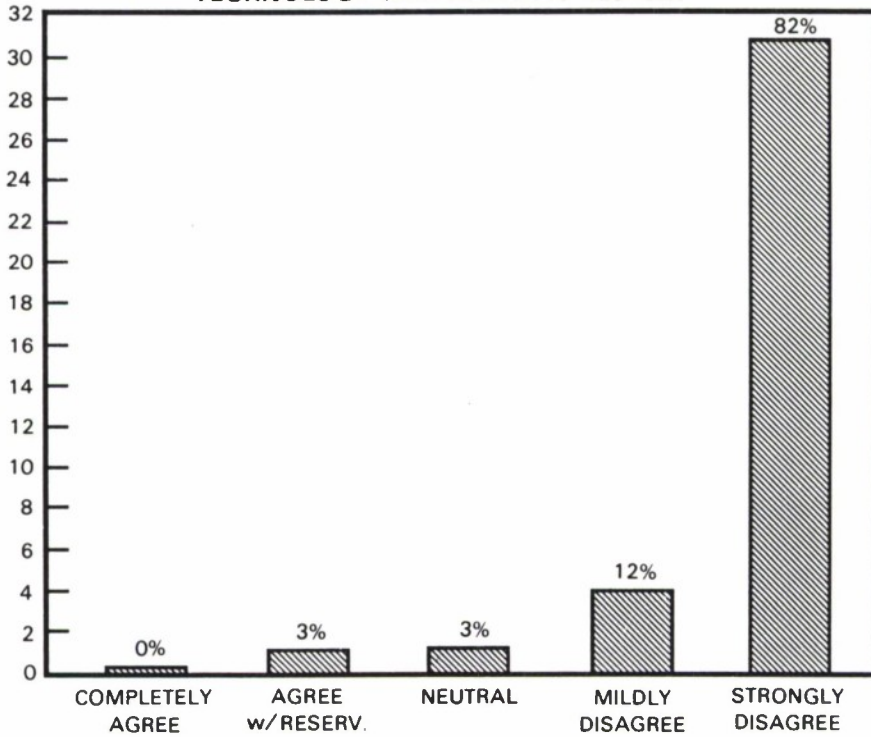


Figure L-12. Fundamental Limits.

THE CRITICAL LIMITING FACTOR TO NN APPLICATIONS IS:

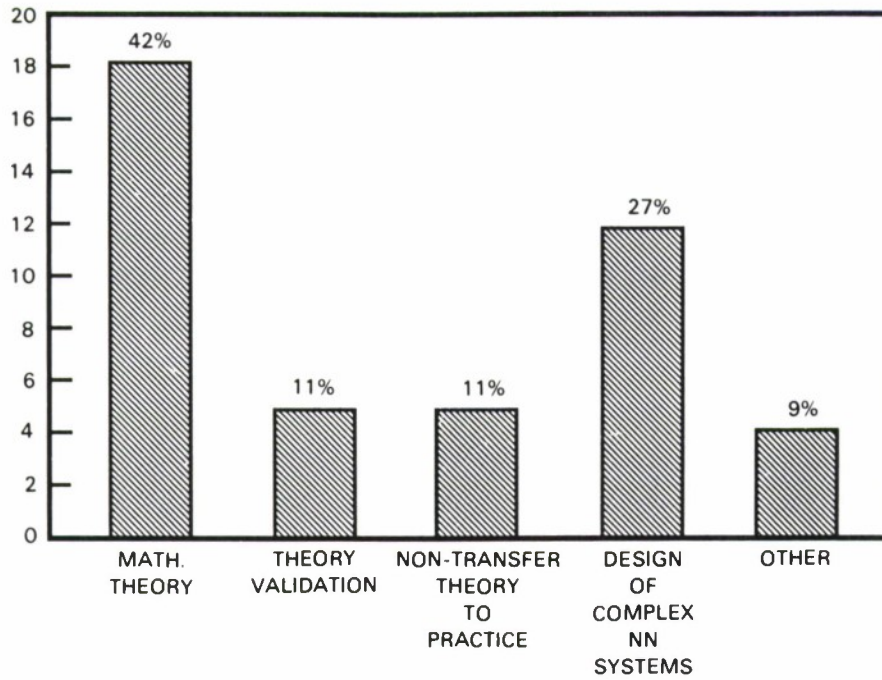


Figure L-13. Limiting Factors.

THERE WILL BE A PSYCHOLOGY OF NN'S DEVELOPED IN THE NEXT 25 YEARS

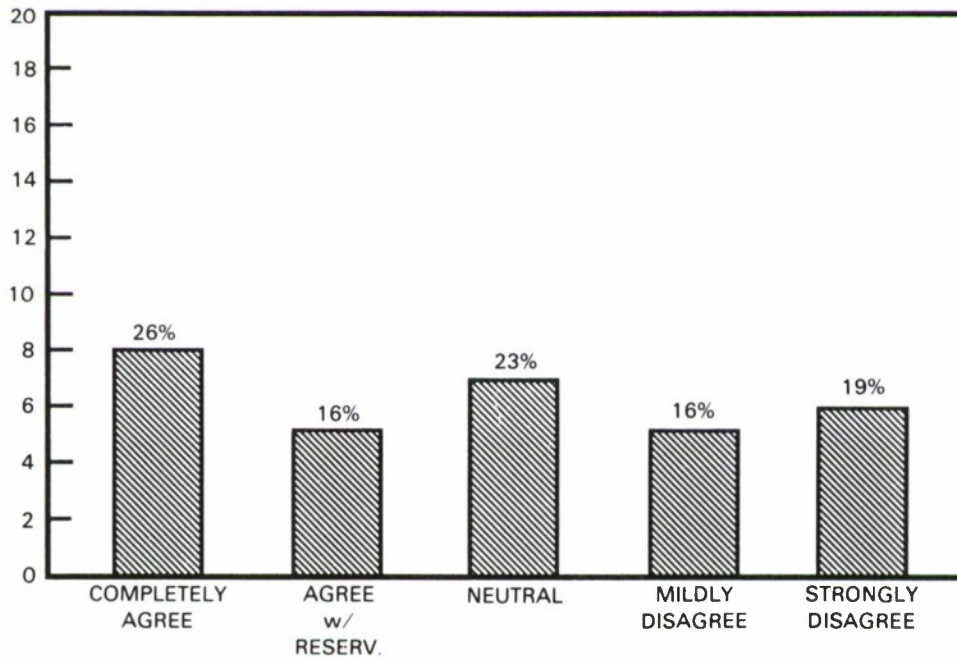


Figure L-14. Neural Network Psychology.

102972-6

102972-5

**PSYCHOLOGY WILL BE CONVERTED FROM A
SOFT TO A HARD SCIENCE IN THE NEXT 25 YEARS**

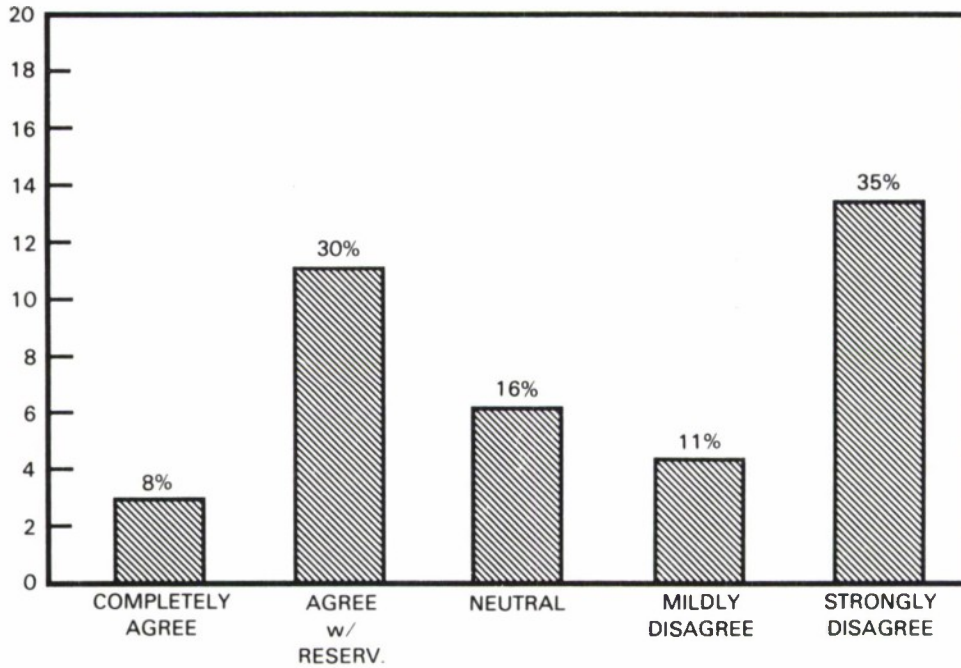


Figure L-15. Applicability to Human Psychology.

**IN 25 YEARS THE PROCESSING CAPABILITIES OF
NN WILL EQUAL OR EXCEED THAT OF THE HUMAN BRAIN**

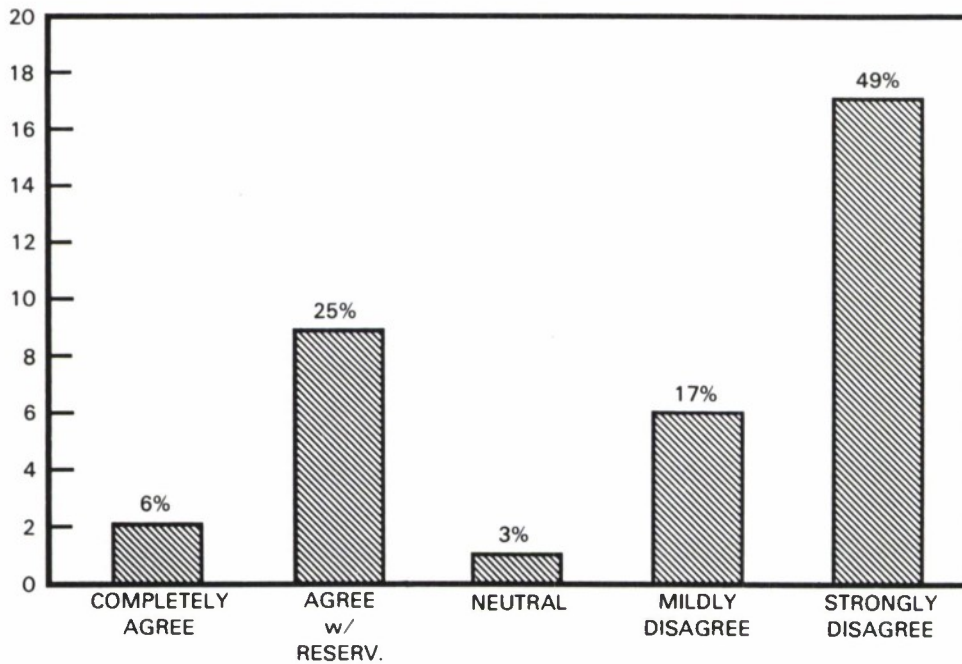


Figure L-16. Human versus Neural Network Processing Potential.

**IN THE NEXT 25 YEARS EXTREMELY ADVANCED NN'S
WILL MANIFEST PSYCHOKINESIS**

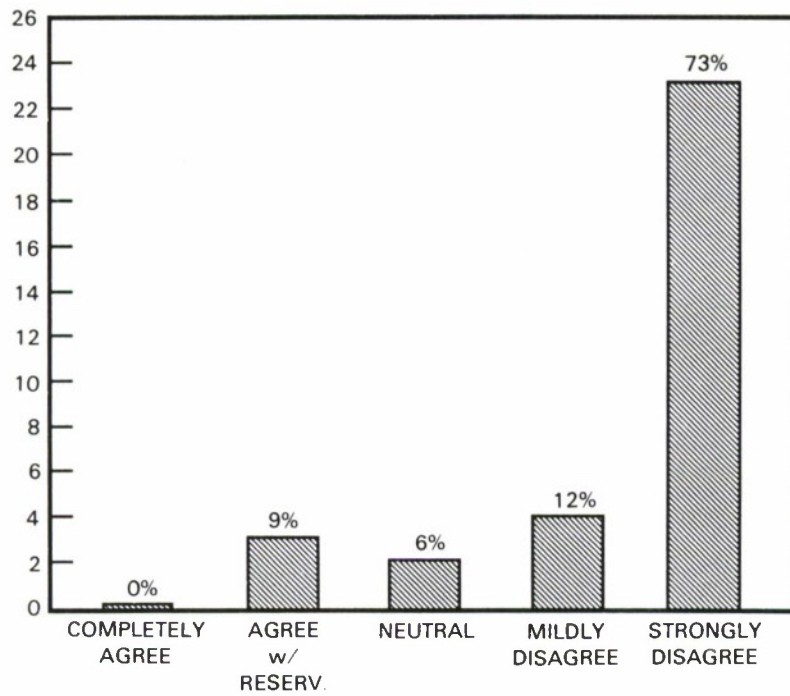


Figure L-17. Speculative Phenomena.

102972-2

Part V

**SIMULATION/EMULATION: TOOLS
AND TECHNIQUES**

Edited by Paul J. Kolodzy

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS	iv
LIST OF TABLES	v
V SIMULATION/EMULATION: TOOLS AND TECHNIQUES	
1. OVERVIEW	1
1.1 Contents of Part V	1
1.2 The Panel Charter	1
2. ALGORITHM AND SOLUTION REQUIREMENTS	3
2.1 Processing Definitions of Neural Network Algorithms	3
2.2 Learning versus Execution Networks	7
2.3 Computation Requirements	8
2.4 Solution Methods	9
2.5 Biological Computational Requirements	13
2.6 Simulation of Implementation Device Characteristics	14
2.7 Summary	17
3. EXISTING HARDWARE AND SOFTWARE	19
3.1 Introduction	19
3.2 Hardware Classes of Neural Network Simulations	19
3.3 Software Tools for Neural Network Modeling	26
3.4 Survey of Commercial Software Packages	27
3.5 Technology Issues in Neural Network Implementations	27
3.6 Summary	32

4. APPLICATION COMPUTATIONAL REQUIREMENTS	33
4.1 Signal Processing	33
4.2 Robotics	34
4.3 Speech	35
4.4 Pattern Recognition – Vision	35
4.5 Application Examples and Current Hardware	39
4.6 Computational Requirements of Current Applications	39
4.7 Summary	42
5. CONSIDERATIONS FOR FUTURE SIMULATIONS	43
5.1 Projections of Standard Hardware	43
5.2 Design Criteria	43
5.3 Vision-Capable Simulator Development	44
5.4 Access to Databases	44
6. CONCLUSIONS CONCERNING NEURAL NETWORK SIMULATION/EMULATION TOOLS AND TECHNIQUES	47
6.1 Summary	47
6.2 Conclusions	48
REFERENCES	50
APPENDIX A – PARALLEL PROCESSING HARDWARE	51
A.1 Butterfly	51
A.2 CAPP	51
A.3 Connection Machine	52
A.4 GAPP	52
A.5 iPSC	53

A.6	Massively Parallel Processor	53
A.7	RP3	54
A.8	Warp	55
APPENDIX B – SIMULATION QUESTIONNAIRE RESULTS		57

LIST OF ILLUSTRATIONS

Figure No.		Page
2-1	Neural Network Connectivity	5
2-2	Multielectrode Recording Arrays Used for Neuroanatomical Measurements	15
2-3	A Proposed Data Acquisition/Data Analysis System	16
3-1	Types of Processors Used in Neural Network Simulations	20
3-2	Graphical Representation of Neural Network Simulation Capabilities	23
4-1	Signal Processing Application Example	33
4-2	Robotic Arm Movement Example	35
4-3	Isolated Word Recognition Example	36
4-4	Vision Example Schematic	38
4-5	Hardware Capabilities and Application Example Requirements	40
4-6	Simulation Requirements of Current Applications	41
5-1	CCD Multiply/Accumulate Accelerator	45

LIST OF TABLES

Table No.		Page
3-1	Selected Neural Network Hardware Simulators	21
3-2	Neural Network Simulation Engine Comparison	22
3-3	Capacity of Network	29
3-4	Module Parameters	30
4-1	Computational Requirements for Signal Processing Application	34
4-2	Computational Requirements for Speech Application	36
4-3	Computation and Storage Requirements for Vision Example	37
4-4	Simulation Requirements for Vision System Example	39
4-5	Computational Requirements for Selected Applications	42

1. OVERVIEW

1.1 CONTENTS OF PART V

Part V of the Neural Network Study Technical Report consists of six chapters. The first gives an outline of Part V and discusses the charter of the Study's *Simulation/Emulation Tools and Techniques Panel* (Panel 3).

The second chapter, *Algorithm and Solution Requirements*, defines the requirements, mathematical and computational, for performing simulations of neural networks.

The third chapter, *Existing Hardware and Software*, discusses the current status of hardware and software available and the technological issues that must be addressed when designing a simulator.

The fourth chapter, *Application Computational Requirements*, outlines four examples of typical applications the Department of Defense (DoD) community might be interested in solving and their computational requirements.

The fifth chapter, *Considerations for Future Simulations*, lists a number of technological considerations this Panel agreed would need to be addressed to produce advanced simulators for applications listed in the fourth chapter.

The sixth chapter, *Conclusions Concerning Neural Network Simulation/Emulation Tools and Techniques*, summarizes the finding of this Panel, from algorithm and application requirements to existing and future hardware capabilities, and includes recommendations concerning how simulators might be developed for DoD applications.

1.2 THE PANEL CHARTER

The Simulation/Emulation Tools and Techniques Panel's responsibility is to delineate techniques available to neural network investigators using digital (serial and parallel) machines as well as special-purpose hardware, and to define future requirements for both hardware- and software-bound simulators. Neural network simulation needs range from the development and testing of **algorithms** to the development and testing of special-purpose hardware **implementations**.

With respect to algorithm development and algorithm testing, simulation is generally accepted as a way to allow the most flexibility with "reasonable" computational throughput. This is accomplished by using conventional digital hardware and thus distinguishes this panel's role from that of the *Advanced Implementation Technology Panel* (Panel 5).

At this time, neural network simulation of applications is generally overlooked. However, most (if not all) of the neural network applications defined by the *System Applications Panel* (Panel 4) are run on simulators. For that reason, the computational requirements for near-term applications should be investigated by the Simulation/Emulation Tools and Techniques Panel. When designing implementation devices, there are two ways to take into account device characteristics:

- Build the device, analyze its characteristics, and then modify the design; or,
- Model the device characteristics and analyze the design using a simulator.

Thus, this Panel also investigated the possibility of incorporating device characteristics into the simulator.

2. ALGORITHM AND SOLUTION REQUIREMENTS

This chapter defines the processing requirements for the simulation of neural networks. There are many definitions of neural networks, each one of which can lead to a different set of computations. These computations can range from iterating a signal through a sequence of maps to the solving of a massive number of nonlinear differential equations. Outlined below are the types of neural network definitions and their corresponding computational needs.

2.1 PROCESSING DEFINITIONS OF NEURAL NETWORK ALGORITHMS

One of the main difficulties in designing a general-purpose simulator for neural networks is the overabundance of descriptors used in their definition. This section will group algorithms according to four descriptors:

- Equation type,
- Connection topology,
- Processing schema, and
- Synaptic transmission mode.

These descriptors have been selected for their effect on the hardware and software requirements of simulators as well as on implementation hardware.

2.1.1 Equation Types

The *equation type* descriptor determines the ability of the network to be simulated on special-purpose software or hardware. Difference equations (e.g., iterative maps) can be implemented on high-speed special-purpose hardware due to the inflexibility of the computations involved. Differential equations as well as algebraic/optimization networks perform a variety of calculations determined primarily by the state of the network. Those type of networks require more flexibility in the software and hardware and are therefore more difficult to implement on special-purpose machines.

Differential

Algorithms are described either as sets of difference equations or differential equations. One example of a *differential* equation description is the Oriented Feedback layer of the Grossberg/Mingolla Boundary Contour System:

$$\frac{d}{dt}v_{ijk} = v_{ijk} + h(z_{ijk}) - v_{ijk} \sum_{(p,q)} H(z_{pqk})W_{pqij}. \quad (2.1)$$

Often one is interested in integrating the differential equation solution to steady-state for a given set of initial conditions, although there are many examples where the detailed dynamics of the equations are critical (see for example, [3]).

Difference

Every differential equation can be rewritten as a *difference* equation. For instance, Kohonen describes his Tonotopic Feature Map in either of the following forms:

$$\frac{d}{dt}m_i(t) = \alpha(t)[x(t) - m_i(t)] \quad (2.2)$$

or

$$m_i(t_{k+1}) = m_i(t_k) + \alpha(t_k)[x(t_k) - m_i(t_k)]. \quad (2.3)$$

The dynamics of the differential equation determine the applicability of using the difference equation form for simulation. The difference equation is identical to solving the differential equation using Euler's Method with a fixed step size:

$$x_i(t_{k+1}) = x_i(t_k) + \Delta t \frac{d}{dt}x_i(t_k) \quad (2.4)$$

where Δt is the step size. If the system of equations is considered stiff (see Gear) – that is, if the ratio of the largest to smallest eigenvalue is larger than 100 – then Euler's Method is an incompatible way of representing the differential equation as a difference equation. Under these conditions, other techniques must be used to solve the equation. Those techniques are described later in this section.

If the algorithm can be rewritten as a difference equation, then the computations are reduced to an updating of the values of each neuron by the set of operations given in its description. There is a group of algorithms that are always described as iterative maps. Those algorithms, such as CMAC, are also computed in the same manner as difference equations.

Algebraic/Optimization

Work done in the study of backpropagation and its application to various empirical problems takes a different form. In this case, a neural network is a multi-layer (usually three) network of nested sigmoids. That is,

$$o_i = f_i(\langle \sum_j w_{ij} I_j \rangle) \quad (2.5)$$

where o_i is the output of the i th neuron, which is a sigmoid function (f_i) depending on the neuron of a weight vector (w_j) associated with the neuron and an input vector (\mathbf{I}). The inputs are either fixed external inputs or outputs from other neurons or constants. The weight vectors are either constants or regarded as free parameters whose values are at the discretion of the modeler. Some node or nodes of the system are designated as output nodes. For fixed inputs, there is usually some known desired output. A cost function is determined (usually the sum of squares of the discrepancy of the output of the system from the desired input). The weights of the system are adjusted so as to minimize the cost function. This approach has a great deal of utility for specific applied problems as long as the function to be minimized (*energy landscape*) is not too complicated and the optimization technique is sufficiently powerful.

2.1.2 Connection Topology

General-purpose computers with global random access memory (RAM) can address any data element in memory with uniform speed. This ability to access any data element in memory places constraints on the hardware. Due to device characteristics, either the amount of memory available for global access must be small, or the speed with which it can be accessed must be lowered. Going off-chip or off-board for additional memory incurs a penalty in access time. High-speed simulators attempt to exploit the uniform topology and locality of data storage to allow computation pipelining and/or simultaneous access to multiple data points. The topology of the interconnections in neural network algorithms (now viewed as architectures) determines if such a uniform topology exists.

The connectivity of a network is the measure of how many processors on one level communicate to each of the processors at the next level. If the layers contain processors in a one-dimensional space, then the connectivity between the layers can be represented as a matrix. Each dimension of the matrix corresponds to either the input layer (L^i) or output layer (L^{i+1}) of processors. A *fully-connected* network has a fully-populated matrix (see Figure 2-1). Each input processor is connected to every output processor.

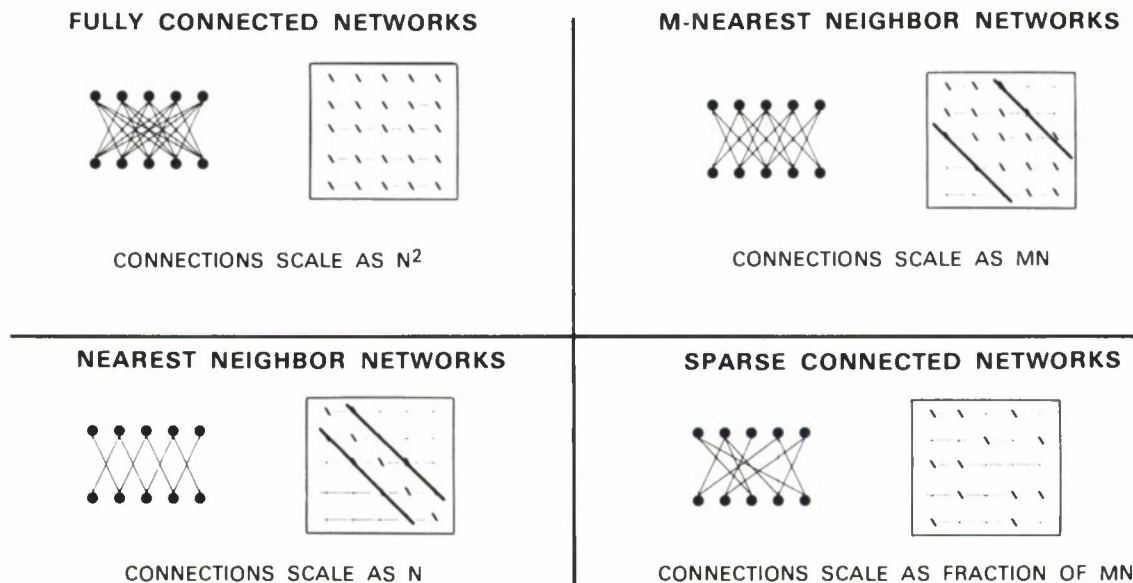


Figure 2-1. Neural Network Connectivity.

A network that is used in preprocessing imagery may use on-center/off-surround interactions. Each processor is connected to its corresponding processor in the next layer with an excitatory weight and to a surrounding region with inhibitory weights. This is a *locally-connected* topology and for a one-dimensional space the matrix will be banded diagonally (see Figure 2-1).

A Markov Random Field network is an example of a system that uses connections only to *nearest-neighbor* processors. There are only three connections for a one-dimensional system: to itself, the neighbor on the left, and the neighbor on the right. Locally-and nearest-neighbor-connected networks are identical when the radius of the surrounding region for a locally-connected network is one neuron. Therefore,

a nearest-neighbor connectivity matrix is also banded diagonally (see Figure 2-1).

The final connectivity category is for connections that are *randomly* distributed between the two layers. These networks may or may not have topological importance to their connections. Their connectivity matrix is randomly populated (see Figure 2-1).

The extension of the connectivity matrix from one-dimensional systems is in two directions: higher-dimensional systems (2-, 3-D for vision, etc.), and to allow connections within a layer (intra-level). The connectivity matrix for higher-dimensional systems becomes a connectivity tensor. The banded diagonal representation remains a banded diagonal in a higher dimension. An alternate representation of the tensor is a series of matrices. The connectivity of each matrix will be a banded diagonal. When intra-level connections are allowed, the former representation becomes one quadrant of an expanded matrix. The dimensions of the matrix become a combination of input-and-output-layer neurons. The former matrix is the quadrant depicting input-to-output neurons.

Except for randomly connected networks, the connection topology of all other networks is compact. The requirement for global RAM for the entire network is unnecessary. Therefore, hardware can take advantage of the finite need for accessing global memory in the processor. Randomly connected networks require complete random access and most likely can be simulated on general-purpose hardware processors.

2.1.3 Processing Schema – Synchronous, Asynchronous

Nodes in a network are said to be updated *synchronously* if the output at the next iteration in the network depends entirely on the prior state of the network. Nodes in a network are said to be updated *asynchronously* if they can be updated on a component, or block-by-block, basis. Virtually all neural network models have in the past assumed synchronous updating, but asynchronous updating has advantages in terms of ease and speed of computation.

However, asynchronous processing in numerical processing has a more limited domain of stability (see [9] for examples). At the Jet Propulsion Laboratory, Barhen has recently begun to investigate asynchronous networks. Future hardware considerations may make such networks a promising alternate approach. Asynchronous operation may relax overhead, such as single- and double-buffering of data lines as well as the additional control lines needed for synchronous operation.

Of course, the processor can work asynchronously and simulate synchronous updates with the cost of additional memory storage. There is thus a complicated tradeoff of stability, memory access, and update speed which is, unfortunately, model-dependent. For the near-term, simulators may not need the ability for such operations due simply to a lack of interest. The driving force in the future for such a class of algorithms, and thus simulators, may be implementation device limitations.

2.1.4 Temporal and Non-Temporal Synaptic Transmission

The general computation model for neural networks is neuronal values multiplied by synaptic weights summed across the input to a neuron. The neuron acts on the summed value and its output is multiplied by weights and used as an input to other neurons. The output of the neuron has either a **binary, integer,**

or **floating point** value. The propagation time (or delay) between neurons is considered negligible. Computations are performed assuming *non-temporal synaptic transmission* of data. Researchers interested in modeling biological tissue have a slightly different computation model. The output of a biological neuron is a series of pulses whose frequency can be related to the output value of the general computation model. The propagation time between neurons has an impact on the phase of the received signal. The *temporal synaptic transmission* characteristics are essential to the performance of such systems.

Bower, at the California Institute of Technology (CIT), and others have elaborate models that require the simulator to keep track of the phase of all the pulses within the network.

2.2 LEARNING VERSUS EXECUTION

Networks that allow changes in the synaptic weights are called *learning networks*. An example of such a network is

$$\frac{d}{dt}x_i = -x_i + f_i\left(\sum_j z_{ij}x_j\right). \quad (2.6)$$

The value of z_{ij} represents the synaptic weight between the recipient node, whose activity is x_i , and the node whose activity is x_j . Typically, when z_{ij} is varying, it obeys an equation of the following form:

$$\frac{d}{dt}z_{ij} = -h(z_{ij}) + \alpha x_i(t)x_j(t - \tau_j') \quad (2.7)$$

where h is typically a linear function and called a *forgetting function*. These equations are called *modified Hebbian rules* of learning. If x_i and x_j are replaced by $\frac{dx_i}{dt}$ and $\frac{dx_j}{dt}$, the equations are known as *modified differential Hebbian rules* of learning.

In Equation 2.5 as well, one can distinguish between the phase of operation where the weights change and the phase of operation where the weights are adjusted so the desired transduction is achieved. In both cases, the learning phase requires much more complicated and computation-intensive operations per step. For example, in a fully-connected network of dimension n , weight updates require order n^2 operations.

Also, the dynamics of learning are typically slow: weight adjustment requires orders of magnitude more time than operation of the network in the steady-state condition. It is possible to compute a rough measure of the necessary number of presentations of a training pattern before steady-state occurs. The number of presentations is proportional to the number of patterns in the training set. Therefore, to train a network with 250 patterns, each pattern should be presented roughly 250 times. Training time is over two orders of magnitude longer than execution time!

Once, however, a satisfactory learned value of the weights is achieved, the dynamics are much more rapid and, especially in a sparsely-connected case, the networks transduce an output for a fixed input orders-of-magnitude more rapidly than in the learning phase. It follows that after learning a specific application, the application often can be “hard-wired” and the result computed much more rapidly than in the learning phase.

2.3 COMPUTATION REQUIREMENTS

Note, again, that the descriptors used for the neural network definition translate into the type of computations necessary to perform the simulation. Only two descriptors are needed to distinguish the simulation tasks: equation type and synaptic transmission characteristics. Three tasks evolve from those descriptors:

- Multiply-accumulate operations,
- Solving differential equations, and
- Pulse-code storage with link scanning.

2.3.1 Multiply-Accumulate

Due to the nature of neural networks – many parallel processors with a high degree of connectivity – it is not surprising to find that the bulk of their computations are in multiply-accumulate operations. In every network equation there is an operation such as

$$\sum_i^N w_{ij} x_i \quad (2.8)$$

where N is the number of interconnections to neuron j . This is a typical example of a multiply-accumulate operation.

The relative importance of the multiply-accumulate operation to the neuronal computation is overwhelming. For a network with an average connectivity of ten,

$$10 = \frac{\sum \text{Connections}}{\sum \text{Neuron}} \quad (2.9)$$

and when the number of operations for computing the neuronal output is four, the number of multiply-accumulate operations is 2.5 times that of the neuronal operations. Average network connectivity can range from one connection per neuron to 10^4 connections in biological tissue.

2.3.2 Solving Differential Equations

The dominant methods of solution involve linear vector sums of the values of the answer at prior points and may involve multiple function evaluations. Since the function's evaluations are normally dominated by multiply-and-accumulates, this leads to a higher dominance of multiply-and-accumulates. Note that many of the reliable differential equation solution methods (e.g., Gear's Predictor Corrector Method) require the storage of many back sets of values. Thus, using Gear's nonstiff method (see [4]), which is a very widely used method to solve differential equations, may require up to 14 times the number of data points as the size of the system.

2.3.3 Memory and Link Scanning

For many simple neural networks whose network topologies are one-dimensional, a simple vector pipeline or systolic array will provide adequate addressing for the multiply/add/accumulate operations which constitute the bulk of the computations done in a neural network. However, many – perhaps the majority – of neural networks need more complex data access capability because often local relationships are multi-dimensional. Research is needed in devising a local access scheme which is general enough for most applications but restrictive enough to provide memory access at a higher rate than general-purpose computers.

It is worth noting that since the dominant operation in neural network computation is add/multiply/accumulate, with relatively infrequent stores to memory, read-and-fetch operations need to be much faster than the relatively infrequent write operations.

2.4 SOLUTION METHODS

In this section, the classical techniques for solving problems in unconstrained optimization and integrating differential equations are outlined. Where noted, new advances in technique promise both increased stability and speed of analysis. These advances should be more widely used and adapted to neural network research for a number of reasons.

The first and most obvious reason is that the best numerical technique yields an increase in precision for a given computational effort, or the same precision for less computational effort. However, it is often remarked that little precision in output is needed in neural network research. If this is the case, why should one be concerned with numerical methodology? Even when one is only interested in a result with a small output precision, one should realize that error often propagates in an explosive manner if one uses an inferior numerical technique, so that the output of the system may be in fact orders of magnitude in error. In this case, the output of the system will be dominated *by the details of the numerical technique used*, not by the neural model. This is hardly a desirable situation.

Techniques that numerically integrate differential equations are at a relatively mature state of development. One probably should not expect major breakthroughs in algorithmic efficiency or accuracy at this time. However, problems in optimization are still poorly understood and breakthroughs in the efficiency and the reliability of algorithms may be occurring and may be fostered by an infusion of well-placed funding.

2.4.1 Classical Methods for Solving Differential Equations

Most methods for integrating differential equations involve constructing formulae which approximate the Taylor Series Expansion of the integral of the differential equation at a point. The number of terms in the series minus one is called the *order* of the method. The step increment used is called the *stepsize of the method*. We will discuss three such classes of methods.

Euler's Method

Euler's method approximates the solution of a differential equation by a first-order difference equation. Thus if the differential equation is $\dot{x} = f(x, t)$, in the differential equation of the neural network to be modeled, Euler's method replaces the differential equation by the difference equation

$$x_{i+1} = x_i + f(x_i, t_0 + ih)h \quad (2.10)$$

where x_i is the approximate value of the integral equation at $t_0 + ih$. This method has the advantage of simplicity and ease of calculation. It has the disadvantage of instability and low precision for the computational effort. In one simple problem, Euler's method required more than *two orders of magnitude more iterations to obtain the same error* as a simple Taylor Series expansion of order 2 with the same error. [7].

For this reason, neural network simulators should be built which implement more sophisticated techniques than Euler's method.

Runge-Kutta Methods

Runge-Kutta methods occupy a unique place in differential equation techniques in that they are self-starting and easily adjust to different step sizes. Runge-Kutta methods accomplish this by using sums of nested function evaluations to derive the approximation to the Taylor Series expansion. Large order Runge-Kutta methods require large amounts of working memory and many function evaluations to store intermediate results. In addition, for problems which have very high dimensions Runge-Kutta methods are generally less efficient than the predictor-corrector methods described below. In summary, Runge-Kutta methods are suitable for small to intermediate size systems or for producing the initial values for large size systems.

Predictor-Corrector Methods (Gear's Method)

If $\dot{y} = f(y, t)$ is the general form of a differential equation, then the most general form of a linear predictor-corrector method is:

$$\sum_{j=0}^k \alpha_j y_{n+j} = h \sum_{j=0}^k \beta_j f_{n+j}, \quad n = 0, 1, 2, \dots, \quad (2.11)$$

where $f_k = f(y_k, t + kh)$, α_j , and β_j are constants and $\alpha_k \neq 0$. Such a method is called a *linear k-step method*. In order to generate the sequence of approximations, it is necessary to obtain k starting values y_0, y_1, \dots, y_{k-1} . Then the computation takes one of two possible forms. First, if $\beta_k = 0$ then y_{n+k} is immediately obtained and such a method is called an *explicit multi-step method*. If $\beta_k \neq 0$, then such a method is called an *implicit multi-step method*. In the case of an implicit method, an approximation is found to y_{n+k} by iterating the defining equation until the difference in the iterates for y_{n+k} is sufficiently small. Because the implicit multi-step methods are more stable and therefore more accurate than the explicit ones, one typically proceeds as follows: first one uses the explicit multi-step method to predict a

starting iterate to a corresponding implicit multi-step algorithm, which is then iterated a number of times to produce a final output. Such a method is called a *predictor-corrector* method.

Gear's method is a predictor-corrector method which adaptively adjusts the order of the method (the number of steps in the summation) and the stepsize in order to minimize the error for a given iteration. Values between steps are extrapolated if necessary. Gear's method is generally fairly efficient over a wide range of error bounds and requires a fairly minimal number of function evaluations and is suitable for large systems. Drawbacks include the need to store often 15-to-20 times the number of output points as working storage, and many iterations for a high degree of accuracy.

2.4.2 Classical Methods of Solving Optimization Problems

The classical methods for solving optimization problems are modifications and adaptations of *Newton's method* and the *method of steepest descent*. We will concentrate on *conjugate gradient methods*, which "... although they are far from ideal are the only reasonable method available for a general problem in which the number of variables is extremely large" [6].

Although there are methods which theoretically will find the local minimum with probability one, should one compute for an indefinitely long period of time (see for example [5]), these methods are not tractable at the present time for most large minimization problems.

The Method of Steepest Descent

The method of steepest descent, known in the neural network literature as *backpropagation*, updates the weights of the network in the direction of steepest descent. That is,

$$w_{n+1} - w_n = -\lambda \nabla_w f \quad (2.12)$$

where f is the function to minimize and λ is a parameter specified to complete the description of the algorithm. This method has the advantage of simplicity and reliability. However, convergence is often excessively slow near local minima. Substantial speedups in convergence can be achieved by using classical minimization techniques.

Newton's Method

Newton's method for a minimization problem can be written:

$$w_{n+1} - w_n = -\lambda \mathcal{H}^{-1}(\nabla_w f) \quad (2.13)$$

where \mathcal{H} is the Hessian of the function f to be minimized. This is an attempt to apply Newton's method to the equation $\nabla_w f = 0$. Newton's method converges extremely rapidly in a neighborhood of the minimum. However, this method requires the computation of a matrix of second derivatives and then a matrix inversion which is very computationally costly. For a large system, the amount of storage which may be necessary for this matrix or the Hessian matrix may be prohibitive. Far from the minimum point, Newton's method may be unstable and require many iterations with small stepsize λ for convergence to the minimum.

Quasi-Newton Methods

These methods attempt to replace the Hessian Matrix or its inverse by an approximation which is calculated from successive values of the gradient $\nabla_w f$. The methods shares rapid convergence near the local minimum with Newton's method, may be more stable far from the minimum, and are computationally more efficient.

However, often they still require storage of a large matrix of coefficients, except when the problem is very sparse. Moreover, update time is often quite expensive (order n^2 , where n is the dimension of the system). These methods are only suitable for small or intermediate-size problems, or for large but very sparse problems.

Conjugate Gradient Method

These methods are techniques of modified gradient descent chosen so that each direction is conjugate (orthogonal) to the prior direction. The direction of each successive step d_{k+1} is chosen so that

$$d_{k+1} = -\lambda_w f(w_{k+1}) + \beta_k d_k \quad (2.14)$$

with

$$\beta_k = \frac{\|\lambda_w f(w_{k+1})\|^2}{\|\lambda_w f(w_k)\|^2} \quad (2.15)$$

These methods have the advantage that they only increase the amount of storage by a factor of two over the method of gradient descent, but the theoretical converge rate is faster than linear. In fact, the rate of convergence is usually quadratic for each n iterations of the method where n is number of weights. Their disadvantage is that convergence can be slow if the condition number (the ratio of the largest to smallest eigenvalue of the Hessian Matrix) is at the local optimum. Preconditioning offers a way to improve these results. In any event, the conjugate gradient method is typically far superior to gradient descent in a neighborhood of the local minimum.

2.4.3 New Methods of Solution

The new methods of integrating differential equations focus on extensions involving ease of use and automatic changing between solution methods (for example see [11]). There have been no major algorithmic breakthroughs recently which markedly increase the rate of computation, so the focus here is instead on methods of optimization and methods for multiprocessors.

Preconditioning

Recall that the rate of convergence of the conjugate gradient method speeds up considerably if the ratio of the largest to the smallest eigenvector is close to one. The idea of preconditioning is to rescale the directions of descent in an appropriate manner, so as to substantially reduce the condition number of the matrix. The methodology of preconditioning for selecting the appropriate choice of the preconditioning

matrix is under active research. A number of choices are the diagonal elements of an approximate Hessian matrix. (See [6] for further references.)

2.4.4 New Methods for Multiprocessors

With the advent of multiprocessing computer architectures the focus of solution methods for systems of differential equations has changed. Although initially developed for uniprocessor systems, the multigrid solution methods (see [2]) are well suited for concurrent hardware. While standard relaxation techniques work on a standard mesh size, the multigrid technique uses a variety of mesh sizes (scaled by factors of two) for relaxation. The application of this technique to concurrent processors appears straightforward. The number of processors represents the most coarse mesh size and each processor will process the region associated with its portion of the coarse mesh for each of the finer meshes. Variations of this scheme, such as using stochastic access to the processors, are also being investigated [13].

2.5 BIOLOGICAL COMPUTATIONAL REQUIREMENTS

Simulations are required for understanding biological as well as artificial neural networks. Two sets of biological computations are generally found: the simulation of biological systems, and the processing of experimental data.

2.5.1 Simulation of Biological Systems

Bower at CIT and others have been developing custom simulation packages for analyzing specific topological and chemical structures found through neuroanatomical examinations. In particular, Bower has developed a software package to execute on a Sun Microsystems workstation or a Hypercube parallel processor to investigate the dynamics of the pyriform cortex. The addition of phase information to the synaptic transmission is a divergence from the standard artificial neural network simulation approach.

2.5.2 Experimental Measurements

The ultimate goal of many neural network modelers is to develop systems which function similarly to the brain. For this and other reasons, it is of interest to attempt to understand how the brain is processing information. Similarly, there is a large body of knowledge from the field of behavioral psychology on how living systems respond to stimuli. What neural network modelers want to know is the design which ties individual cell characteristics to behavior – i.e., the information processing methods of large arrays of biological neurons.

This body of knowledge remains to be formulated, largely because of technical experimental reasons. First there is the problem of measuring the states of large numbers – say, 1,000 – of neurons in a behavioral situation. Figure 2-2 indicates two possible methods – micro wire and dagger probes. The idea behind both methods is to insert fine electrical conductors into tissue and to relay the data about pulse structures out to a computer for analysis. The second experimental problem is to get the data out to the analysis

station on a reasonable number of wires – e.g., by multiplexing micro and milli-volt signals. The dagger approach offers the potential for fixing electronic circuitry on a silicon dagger to help this data transmission problem.

Even assuming that the information can be gathered by a number of probes inserted into an animal, there remains the problem of relating the data to the animal's behavior. This is in itself a substantial problem, probably involving a very large amount of correlation operations on the databases. The Neural Network Study Simulation/Emulation Tools and Techniques Panel considers this analysis problem to be in a separate domain, largely because of the need for vast amounts of digital computational power.

For an order-of-magnitude estimate of the level of computational power required for this analysis, consider the case of 1,000 active measuring sites. Each site will produce up to 50 action potential spikes per second. It is estimated by neurological experimenters that each spike will require 10 bytes to be properly characterized digitally. Thus the output rate from the measurement section of the multicellular recording/analysis system will be at 500 kilobytes per second (kbytes/sec) (see Figure 2-3).

There is another issue, too: how much data do the experimenters need to analyze to determine behavioral characteristics relative to the electrical pulses? The order-of-magnitude time estimate is an hour – or 3,600 seconds. This means that the analysis portion of the system must be able to correlate up to two gigabytes of data.

These estimates indicate that the neuronal measurements can be placed in the lower righthand region of the speed/capacity diagram referred to above: a large capacity but not unreasonable rates, provided that off-line analysis is accounted for. The equating of a connection with a spike profile is taken as reasonable since the analytical operations will be of the correlation type, at least at the beginning. It is interesting to consider the use of artificial neural network models to assist in the analysis, since the computational nature of both the artificial and the biological neuron is signal processing.

The final conclusion is, however, that the determination of the relationship between biological neuronal activity and an organism's behavior is a problem on the same order as the simulation of large, slow artificial neural networks.

2.6 SIMULATION OF IMPLEMENTATION DEVICE CHARACTERISTICS

One of the advantages of neural networks is their eventual implementation in highly parallel hardware for realtime performance. Due to the large number of devices in such hardware, it would be necessary to *simulate* performance prior to actual fabrication. The effects of nonuniformity in the devices with respect to network performance is of especial interest. Three areas that can be affected by such nonuniformity are:

- The neuron nonlinear transfer function (shift in curve or slope change),
- The gain in the synaptic weights, and
- The learning function.

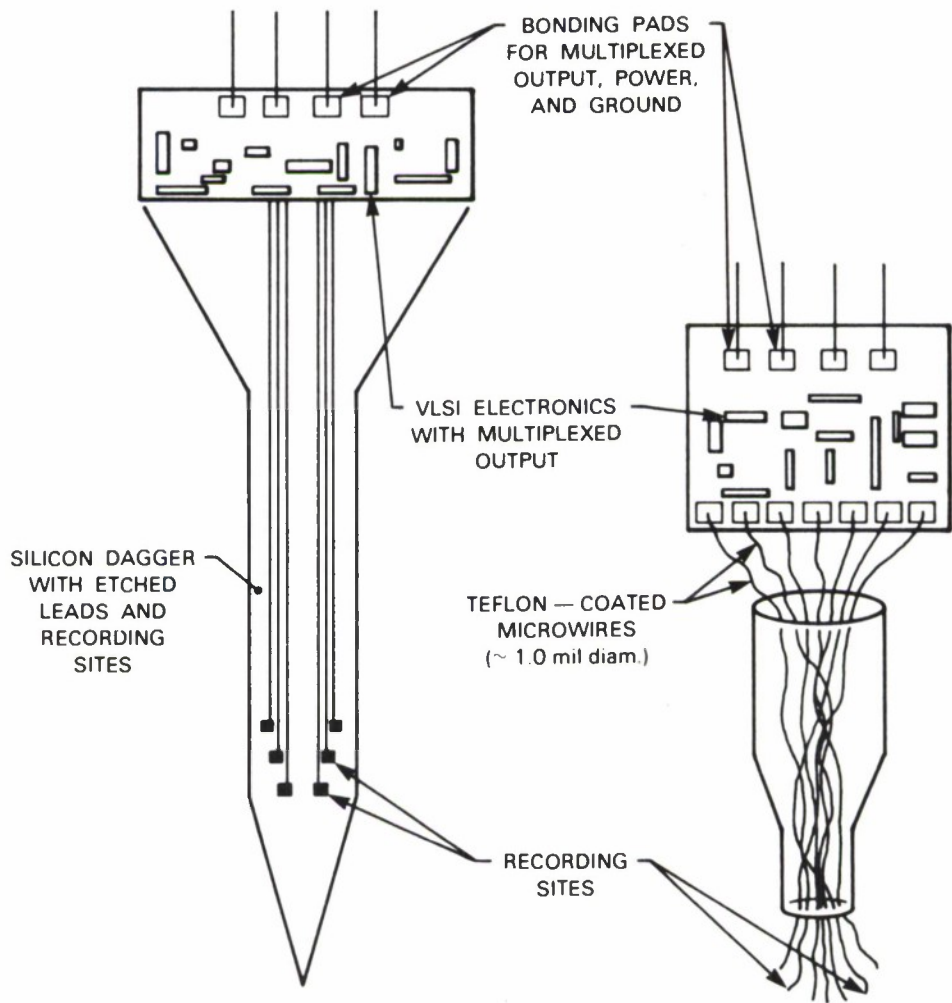


Figure 2-2. *Multi-electrode Recording Arrays Used for Neuroanatomical Measurements.*

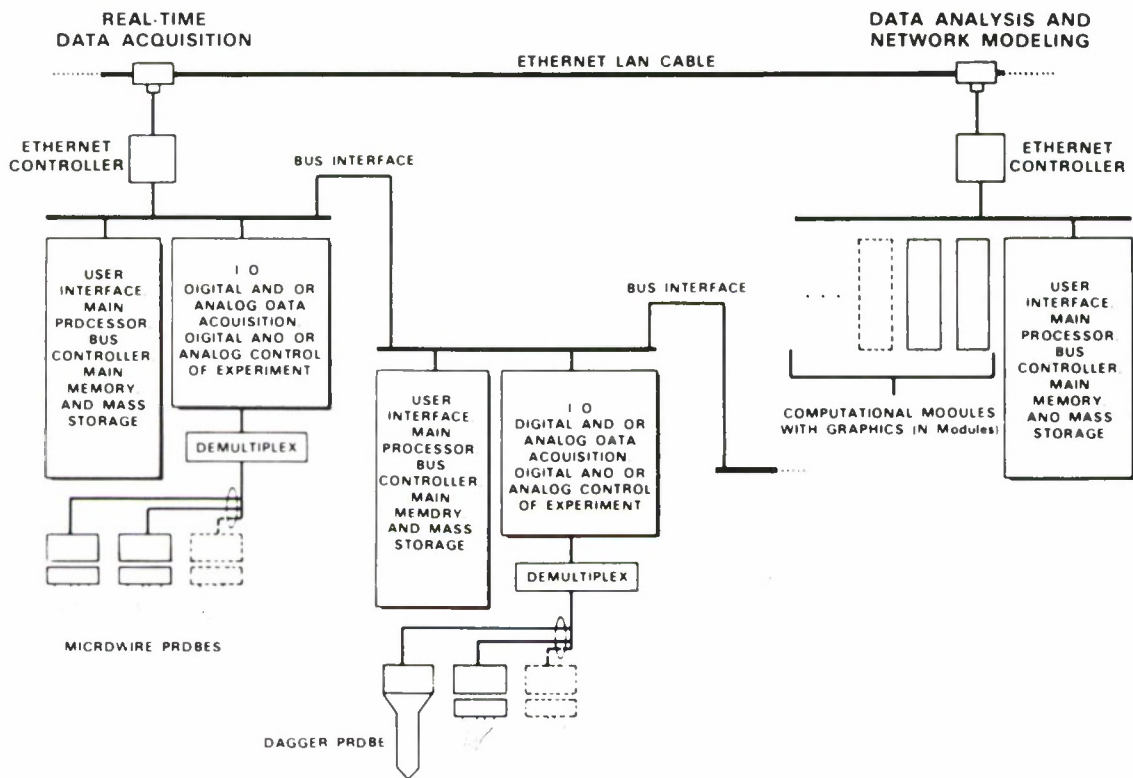


Figure 2-3. A Proposed Data Acquisition/Data Analysis System.

This nonuniformity can be fixed or stochastic. Simulators with the capability to adjust the device characteristics in a deterministic way are necessary to understand the precision necessary in device fabrication.

2.7 SUMMARY

There are a range of computations that must be performed when simulating neural networks. Currently, the bulk of simulations are performed by either iterating difference equations or maps. The intensive processing step for these systems is predominantly multiply-accumulate operations.

The next largest computation structure used in simulations is the solution techniques for large systems of differential equations. Although these structures also require large number of multiply-accumulate operations, they also require additional computations and storage related to the technique and dynamics of the network.

There are a variety of well-established algorithms to solve differential equations. New algorithms are being developed to quicken the solution time and to take advantage of multiprocessor systems.

Optimization networks can be simulated in an efficient manner by borrowing techniques from the literature. Those techniques are usually not tractable for large systems but the conjugate gradient technique is viable for sparse networks.

Although not currently in vogue, biological network simulations and networks requiring asynchronous processing and temporal synaptic transmission qualities are worthy of interest. Simulation hardware and software may need to address the issues associated with these qualities, or at least some significant subset. Simulators may need to be separated by their abilities and inabilities to compute particular systems.

3. EXISTING HARDWARE AND SOFTWARE

3.1 INTRODUCTION

The purpose of this section is to review the current state of the art in neural network implementation software and hardware. The data are based on interviews with selected researchers and developers and on a questionnaire sent to commercial vendors of neural network software.

There is unanimous agreement that *computational speed* will ultimately become a stumbling block in the application of neural networks. However, there are many different scenarios in which neural networks can be implemented and each of these has their own speed requirements. Further, speed requirements are determined by the algorithms being performed. For instance, in trained systems the learning procedures are often time-consuming. However, training can usually be performed off-line before the system is sent to the field.

The environment in which a neural network simulation is placed is a major determinant of its speed requirements. The most stringent requirements are expected to occur in a military system application where the neural network's input data are derived from a realtime sensor. Military systems often have to operate in a restricted volume and over wide temperature ranges.

However, there are few, if any, systems of this sort in existence at this time. The systems surveyed by the Simulation/Emulation Tools and Techniques Panel have all been laboratory research tools. These kinds of experiments have considerably reduced speed requirements and face no major constraints on their size. Speed is more a matter of convenience than of functionality. As a result, existing high-speed computer systems are adequate for today's research purposes.

Two distinct types of neural network investigations exist. One is the simulation of biological neural networks and the other is the use of simplified neuron-like structures for applications research. Of the two, the biological simulations seem to be more computationally difficult. Actual neurons contain several different physical processes and, as a result, are very complex. Application studies concentrate on the benefits of parallel distributed processing and use models predicated, for the most part, on the addition of weighted inputs and soft-limiting (or sigmoid) computation.

3.2 HARDWARE CLASSES OF NEURAL NETWORK SIMULATIONS

As in most computing systems, neural networks are simulated in either batch or interactive modes. There appear to be five distinct types of hardware systems used: two for batch processing, and three for interactive or workstation environments. A breakdown of these processor types and modes is shown in Figure 3-1. The two batch processing methods are the superspeed computer and the massively parallel processor. These are popular with academic institutions which appear to be able to obtain access to these types of computing resources. The workstation systems are the attached processor, the bus-oriented system and micro or minicomputer systems. Each of these systems is discussed below. Table 3-1 shows a comparison matrix of the speeds obtained with each system type and an indication of their size and

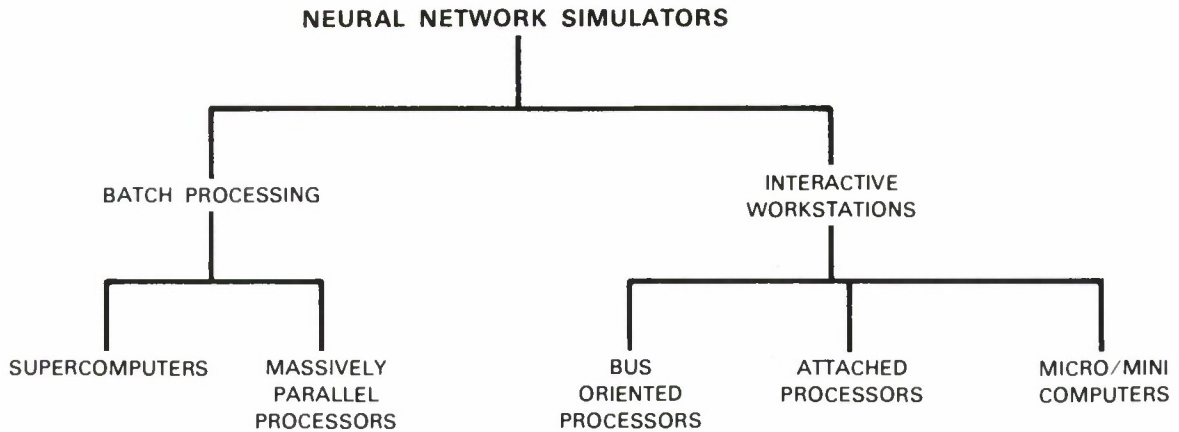


Figure 3-1. Types of Processors Used in Neural Network Simulations.

cost. Additional information on eight *state-of-the-art* parallel hardware engines is supplied in Appendix A. The architectures vary from SIMD (single-instruction/multiple-datastream) to MIMD (multiple-instruction/multiple-datastream), but all take advantage of some sort of parallelism.

Before proceeding too enthusiastically with the use of the speed numbers quoted in Tables 3-1 and 3-2, several explanatory and cautionary comments should be made. In neural network simulations, as in most other computer environments, it is very difficult to develop a metric that fully portrays functional speed. The term ‘interconnections/second’ is common in the discussion of neural networks. This refers to the number of multiply-and-add operations that can be performed in a second and is, of course, a quite important measure. However, the quoted interconnections/sec may or may not include the time necessary to obtain the variables that are to be multiplied and added. It may also ignore other functions that might be required by some algorithms, such as the comparison of values or the computation of a sigmoid function. More data than a simple “interconnections/sec” measurement are needed to appraise computational speed.

Another difficulty in comparing computational speeds is that there are few neural network simulation production systems. It is likely that considerable code optimization is possible for most systems.

In summary, speed differences of orders of magnitude are probably significant; small differences should be used with great caution.

3.2.1 Supercomputer

The supercomputer category is meant to include very high-speed commercial scientific computers. An example of these is Cray Research Inc.’s X-MP. Architecturally, these machines are based on very high-speed semiconductor processes (ECL or GaAs), multi-port, heavily-interleaved memories, vectorized arithmetic units, and relatively long operation pipelines. Multiple processor operation may take place, but the number of parallel processors is not large (less than 20, usually less than 10). Speed is obtained through fast logic and clever processor design rather than a proliferation of processing units.

	<i>Processor</i>	<i>interconnects</i> <i>sec</i>	<i>size</i>	<i>Cost</i>
Super Computers	Cray X-MP	50M/sec	room size	> \$1M
Massively Parallel	Connection Mach.	13M/sec	room size	> \$1M
	Hypercube		room size	> \$1M
	Butterfly	8M/sec	room size	> \$1M
	WARP (10)	10M/sec	room size	\$300K
	Parallon 16/2X	20M/sec	room size	\$40K
Bus-Oriented Mach.	TRW MK III	500K/sec	desktop	\$75K
	TRW MK V (16)	10M/sec	desktop	\$100K
	MX-1/16	120M/sec	desktop	\$300K
Attached Processors	SAIC SIGMA-1	5 - 8M/sec	desktop	\$15K
	TI Odyessy	5M/sec	desktop	\$15K
	AI Net	45 K/sec	desktop	\$3K
Micro/Mini Computer Workstations	IBM PC AT	160K/sec	desktop	\$8K
	SUN 3	250K/sec	desksize	\$20K
	Apple MacIntosh	5K/sec	desktop	\$8K
	Symbolics	35K/sec	desksize	\$100K
	DEC VAX	2M/sec	room size	\$400K

Table 3-1.

Selected Neural Network Hardware Simulators

	HARDWARE	WORD LENGTH	STORAGE (K Intercnts)	SPEED (K Int/Sec)	COST (\$)	SPEED COST
WORKSTATIONS						
<i>Micro/Mini Computers</i>	PC/AT	16	100	25	5K	5.0
	SUN 3	32	250	250	20K	10.0
	VAX	32	100	100	300K	0.3
	SYMBOLICS	32	32,000	35	100K	0.4
<i>Attached Processors</i>	ANZA	8-32	500	45	10K	5.0
	Δ -1	32	1,000	10,000	15K	300.0
	TRANSPUTER*	16	2,000	3,000	4K	750.0
<i>Bus-Oriented Machines</i>	MARK III, IV**	16	1,000	500	75K	7.0
	ODYSSEY***	16	256	20,000	15K	1000.0
	MX/1-16*	16	50,000	120,000	300K	400.0
MASSIVELY PARALLEL	CM-2 (64K)	32	64,000	13,000	2,000K	6.0
	WARP (10)	32	320	10,000	300K	30.0
	BUTTERFLY (64)	32	60,000	8,000	500K	16.0
SUPER-COMPUTERS	CRAY XMP (1)	64	2,000	50,000	4,000K	15.0
* Projected						
** Host Required						

Table 3-2.

Neural Network Simulation Engine Comparison

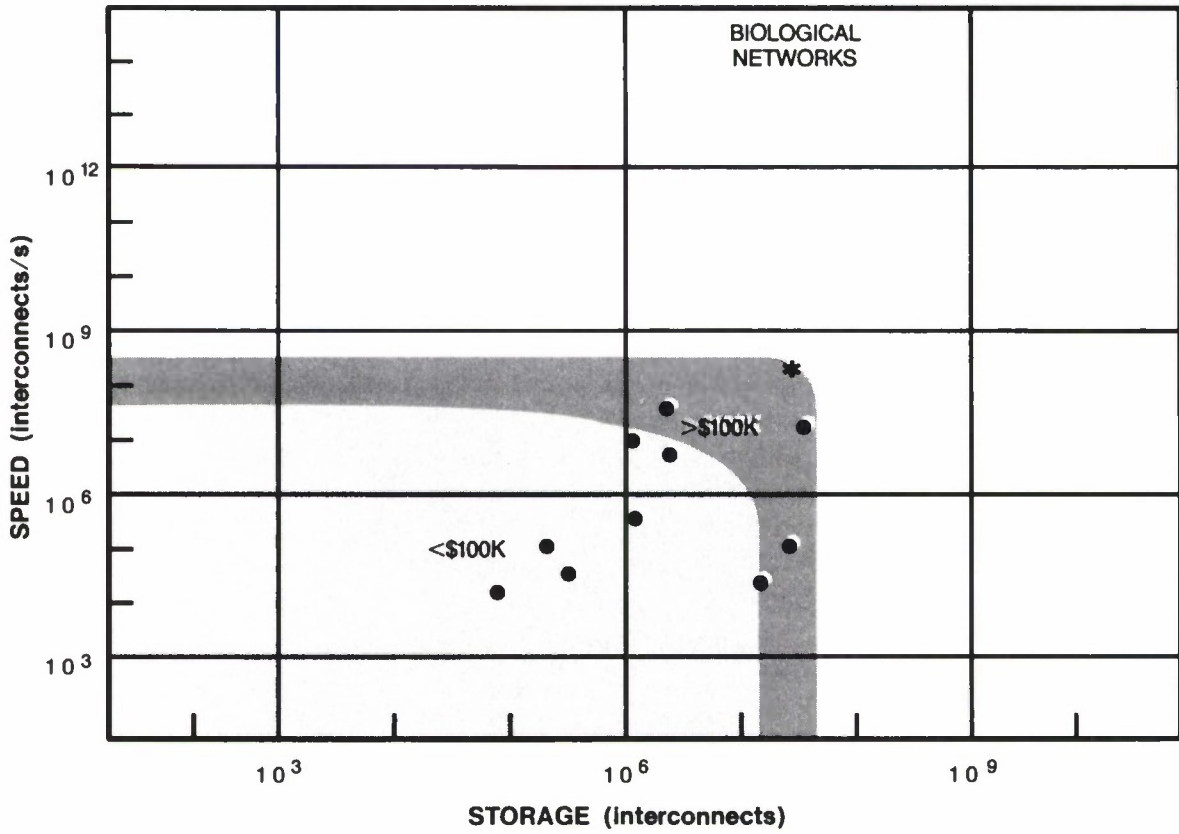


Figure 3-2. Graphical Representation of Neural Network Simulation Capabilities.

The table shows that the supercomputer category is the fastest operating implementation. Its principal difficulties are its size and expense. In addition, respondents to this Panel's questionnaire complained that software was not as easy to use as that available for the workstation environments and that the hardware was a bit temperamental.

Most of the work reported with these machines was custom programmed in a higher-order language (HOL) (e.g., FORTRAN).

3.2.2 Massively Parallel Processors

This is a bit of a muddled category, as it is meant to apply to several new processor types that achieve high speed through the use of a very large number of parallel operating processors. We include here Thinking Machines Corp.'s Connection Machine, the Hypercube, and Bolt Beranek & Newman's Butterfly machine, though in architecture these machines are radically different. They all achieve high speeds, however both the Connection Machine and the Hypercube have restrictive communication paths between processing nodes. As the nature of neural networks is to emphasize a high degree of connectivity, the Hypercube and the Connection Machine can sustain dramatic decreases in throughput because of communication delays. Arguably, it may be possible to eliminate some of these through careful program design. Nonetheless, it would appear that communications are a particularly important aspect of using large parallel structures in neural network emulations.

While the Panel does not do not have throughput estimates for the Butterfly, it would appear that its common memory structure makes it more amenable to neural network applications. On a simple qualitative basis, neural network users are happy with the Butterfly's throughput.

Machines in this class are custom programmed in an HOL, though several software tool sets written in C have been successfully rehosted to this class of processor. In general, applications are debugged in a microcomputer or minicomputer workstation environment before they are run on massively parallel machines.

3.2.3 Bus-Oriented Processors

This is a class of processors that achieves high throughput through the use of multiple microprocessors connected on a commercial bus. The number of processors in this category can vary from two to 20. This is different than the previous category of massively-parallel processors where hundreds to thousands of separate processors are operating in concert. TRW's Mark III is an example of this type of system. It uses a number of parallel Motorola 68020 microprocessors with a customized memory interface to a standard VME bus. Coupled with custom software, the machine is quite successful. Its high throughput is, again, an example of how memory structure (communications between processors), in addition to raw computational throughput, is an important facet of implementing neural networks. While the Mark III system can operate by itself, it is usually operated in conjunction with a Digital Equipment Corp. VAX computer that handles interfacing and loading of the neural network processor.

A limitation in this class of neural network simulation is the speed of the commercial bus. Eventually this will put a limit on the functional throughput, no matter how many processors are added to the system. The point where this limit will be reached is subject to considerable speculation.

This class of implementation is quite efficient in terms of its cost/performance ratio.

3.2.4 Attached Processors

This class of machine consists of neural network engines coupled with a conventional microprocessor or minicomputer. Two examples are the SAIC Sigma I and Texas Instruments' Odyssey. Both of these are new designs and are predicated on high-speed multiply-and-accumulate operations with a specialized memory interface. The attached processors are relatively small and inexpensive yet offer excellent throughput rates.

By attaching to more or less traditional computer systems (TI's Odyssey is really a LISP-based processor), the attached processors can take advantage of extensive graphics, database, and support software.

The SAIC processor uses a BIT Technology emitter coupled with a logic-based multiply accumulator chip and a custom-designed dual memory system. The TI processor uses single-chip digital signal processors (which provide dual internal memory structures). These LSI devices were developed for applications other than neural networks, but with proper system design (especially with regard to memory interfaces) they fit extremely well into neural network applications.

Attached processors appear to offer the best price/performance ratio, provided software exists. In addition, their utilization of existing new VLSI devices indicates that neural network digital implementations do not present radically new architecture problems.

3.2.5 Mini/Micro/Lisp Processors

The most commonly used category of neural network implementation environments is the desktop workstation. Examples of these are the IBM PC AT, the Apple Macintosh, the Sun Microsystems workstation and the Symbolics LISP-based workstation. The Simulation/Emulation Tools and Techniques Panel speculate that these machines are used so frequently because they are so common. Moreover, they offer a vast amount of graphics and support software that can be applied directly to neural networks. The difficulty with stand-alone micro and minicomputers is that they are quite slow when compared with the other implementations. Few of the Panel's respondents felt that the small-scale mini/micro standalone workstations would be adequate for anything other than research and development purposes. Most specialized neural network software tools have been developed for the mini/micro computer environment.

3.2.6 Other Processing Machines and Architectures

There are machines and system architectures that have yet to be used for neural network simulations but it is worth mentioning their merits. The Research Parallel Processor Project (RP3) machine developed by

IBM is one such machine. The RP3 can support up to 512 processors that can be connected in a variety of topologies. Such a system can evaluate different parallel simulator architectures to optimize performance.

IBM is also investigating the question of the level of parallelism necessary for implementation of a program. Although not within the purview of this Panel, the idea of comparing the parallel nature of "classical" algorithms and neural networks is appealing.

The Parallon 16-2 multiprocessor from Human Devices, the Pipe pipeline system from Aspex, and the MX-1/16 multiprocessor developed at Lincoln Laboratory are three untested but high-throughput machines. The Parallon uses a hierarchical communication structure with processors within the same board communicating over a parallel bus. The MX-1/16 uses a crossbar communication structure with a microprocessor and a digital signal processor (DSP) at each node. The Pipe uses video rate hardware to provide a high-speed image processing environment. All three machines offer substantial computation power – from 20 M $\frac{\text{interconnects}}{\text{second}}$ for the Parallon 16-2 to over 100 M $\frac{\text{interconnects}}{\text{second}}$ for the MX-1.

One additional point to be made with respect to the MX-1 and the Pipe machine is with respect to software integration. The MX-1 has been developed to operate in a LISP environment providing high computation throughput and symbolic manipulations. The Pipe uses a graphical description environment to describe how the discrete components for each of the processing stages are to be pipelined together.

3.3 SOFTWARE TOOLS FOR NEURAL NETWORK MODELING

There is an almost universal agreement that neural network simulations require better software tools. However, there is considerable activity in the development of these tools, and it appears that some emerging programs will be very good. Existing software development tools in neural networks seem to fall into three basic categories. These are:

1. Implementations of common neural network algorithms
2. General simulation systems applied to neural networks
3. Tool sets for developing neural network algorithms

The first type appears to be both the simplest and the least useful. Its concentration on existing models makes it very easy to use and a valuable learning tool. However, researchers believe that these classes of systems are of very little value in the development of new algorithms or the implementation of practical systems. Most of these programs operate on either IBM or Macintosh personal computers.

General simulation programs can be of either the differential equation solver type or circuit simulation programs such as SPICE. These packages are quite powerful and can be applied to a large number of problems. However, they are general-purpose tools and are considered by some to be awkward when applied to neural networks. Further, these classes of software tools are considered to be too slow by some researchers.

To be fair, it should be pointed out that the people who were most vocal in criticizing existing system simulation tools were the same people who were developing alternative neural network tool sets. This leads to a question: are custom tool sets being built because standard simulation packages are inadequate, or are standard simulation packages being criticized to justify the building of custom tool sets?

In any case, most neural network support software development is in the area of custom neural network packages. One or two such programs are being undertaken by every academic institution that interviewed by the Panel. In addition, all of the commercial companies developing neural network workstations are also providing custom software packages.

At the least, software tool sets provide a language for describing new neural networks and a graphics interface for portraying both the structure and the performance of neural networks. The difficulties are that the complex, multi-variable nature of neural networks is very difficult to present. Energy surfaces exist in n dimension space, and interconnection matrices are difficult to describe in a logically consistent and graphical manner.

One area of debate concerns the value of providing a graphical method of drawing in nodal interconnections. One school holds this to be a natural way of expressing neural network structures, while another (larger) school feels that this method of defining networks is worthless for networks of practical size.

Several researchers expressed the need for a standard simulation language to simplify comparisons of different types of networks. Not surprisingly, these people proposed their own tool sets for the standard and were not particularly knowledgeable about other developments. It is probably premature to attempt to develop a standard neural network language.

3.4 SURVEY OF COMMERCIAL SOFTWARE PACKAGES

As part of the Simulation/Emulation Panel's efforts, a questionnaire was sent to all suppliers of commercial neural network software. Fourteen responses were received for both commercial and internal software packages. A complete list of the results is given in Appendix B. The primary host for the simulators was the IBM PC/XT/AT family of personal computers.

A surprising finding was the use of LISP as the base language for many of the simulators; the LISP environment is used because of the ease with which software systems can be developed. *This finding suggests an opportunity to incorporate network algorithms with expert systems in a straightforward manner.*

Another finding was the specificity of the algorithms implemented on the simulators. Most packages are either for proprietary algorithms or a selected set of algorithms. The topology of the networks are flexible but the algorithms are fixed. The ability to construct new algorithms is not universal among software simulators.

3.5 TECHNOLOGY ISSUES IN NEURAL NETWORK IMPLEMENTATIONS

A popular belief is that digital neural network applications present requirements for a whole new class of digital VLSI components. Perhaps this will be the case when neural networks find widespread real system

applications. However, for the present, neural network implementations are very similar to conventional digital signal processing applications.

In implementation terms, neural network simulations have the following structural components.

- A majority of computations are multiplication followed by addition.
- Roughly two memory addresses are required per multiply-accumulate.
- Memory addressing is regular and predictable, as opposed to random or data-dependent.

These properties are very similar to the computational requirements of such digital signal processing functions as FFT calculation, recursive and non-recursive digital filters, and one- and two-dimensional correlation functions.

The computation of sigmoid functions appears to be unique to neural networks, but is actually similar to *sin* and *cos* interpolation functions also found in digital signal processing. Thresholding and finding the largest value of a set of signals are also common both in neural networks and in digital signal processing.

Digital signal processors are already a major component in military systems. Furthermore, commercial industry as well as the VHSIC program have invested considerable resources in the development of VLSI devices designed for digital signal processing applications. The most applicable classes of these to neural networks appear to be:

Single-chip digital signal processors which are available from: TRW, Texas Instruments, Motorola, Zoran, Analog Devices, Plessey, and others.

“Word slice components,” chip families that normally provide an arithmetic multiply-accumulate chip, a memory address generator, and a controller. They are available from Analog Devices, Weitek, Honeywell, BIT Technology, Advanced Memory Devices, Texas Instruments, and others. Word slice components were the primary target for Phase I of the VHSIC program.

These are the kinds of components used in the attached processor class of neural network implementations. It is not surprising that this class demonstrated the best performance/cost ratio. It had the advantage of using the extensive technology base already developed for digital signal processing.

Independent of the fate of neural network implementations, digital signal processing will continue to play a major role in both military and commercial systems. Neural network implementations will essentially enjoy a “free ride” on this technology bandwagon.

Another requirement of neural networks is high-speed memory. This is perhaps not as severe a requirement as it might first appear, given that the regular and predictable addressing structure of neural networks allows the achievement of high memory speeds through memory interleaving and the utilization of cache structures. Nevertheless, existing technology trends are pushing memory speeds as fast as possible. Both computers and digital signal processors rely on high-speed, high-density memories. ECL and GaAs memories promise speeds of one to five nanoseconds, while CMOS static memories of medium density are available with 15 nanoseconds access times.

The point is that the components for neural network implementations already exist and are being improved as fast as the state of the art will allow. The challenge in neural network simulations lies in developing the architectures to effectively use these components. These architectures, in turn, will depend quite heavily on the particular algorithms and applications being considered.

Another technology that is often suggested for neural network implementations is the use of massively-parallel computing structures. These systems rely on thousands of simple processors connected together and operating in a single-instruction/single-datastream manner. The advertising says that there are thousands of relatively slow neurons in biological systems, so thousands of slow processors should be the best way to simulate them.

There is a major fallacy in this reasoning. Electrical connections and their control are quite expensive, while biological connections are cheap. The massively-parallel structures are all limited in the way their constituents communicate. This limits these machines' ability to operate as conventional computers, and it is even more limiting in the implementation of neural networks. Clever programming can often compensate for the deficiencies of massively-parallel structures, but cleverness is both rare and expensive. Furthermore, it can't be automated (yet).

It would appear that the easiest way to digitally implement neural networks is with a relatively small number of high-speed specialized processors operating with a high-speed memory structure. This is, in essence, the approach taken with the supercomputer and the attached processor class of neural network implementations. These two examples represent different cost and size environments but essentially the same architectural philosophy.

3.5.1 General Processing Issues

To envision the general processing considerations for a digital simulation of a neural network, consider a model which specifies complete connection between input and output vectors, each with N components – e.g., auto-association with a distributed representation. Assume that the CPU/memory system is arranged with C memory locations primarily associated with each CPU, where C is also the memory address capacity of the CPU. Assume, also, that each of the C locations stores a word of W bits (see [10]).

In a given local memory, one wishes to store the neural network weights corresponding to the vector components to be calculated by the CPU associated with that memory. For an auto-associative system, the number of weights can be as large as N^2 . Thus, the capacity of the memory can hold two relationships to vector size:

Case	Computational Architecture
$C > N^2$	1 CPU
$C < N^2$	$M = \lceil N^2 \rceil / C$ CPU's

Table 3-3.

Capacity of Network

Either one CPU can address sufficient memory to completely store the matrix associated with the sensor output vector, or M CPUs are required along with M memories, each with capacity C. Note that the required number of CPUs scales with the square of the vector. This is a worst-case estimate because of the assumption that the neural network algorithm calls for complete connectivity. For sparse, banded connectivity, a considerable reduction in M is possible.

Each CPU/memory module will be assumed to be described by the parameters listed in Table 3-4 below. It is instructive to calculate the time required for a module to compute the matrix-vector process associated

W	CPU/Memory Word Length
T	CPU Clock Time for One Word Execution
C	CPU External Memory Capacity [$N^1/2$]
M	Number of CPU'S used to Calculate 1 Vector
L	Number of Instructions for a Multiply Accumulate with Local Memory Access Off Chip
R_b	Data Rate for BUS System between Modules

Table 3-4.

Module Parameters

with its C weights and $C^{\frac{1}{2}}$ vector components: it is simply the number of multiply-accumulate operations times the number of machine cycles for each multiply-accumulate with memory access times the machine cycle time.

$$\text{Execution Time} = L * N^2 * T = L * C * T.$$

Because of the well-structured computations associated with a matrix-vector operation, the number L can be quite small. In DSPs, for instance, $L = 3$ with pipeline access of external memory and pipeline accumulation. For MOS technology, $T = 200$ nSec is typical of high-volume production in 1988. A reasonable memory access for a MOS DSP is 64k words, making $N = 200$ a reasonable estimate for the size of the matrix to be stored/executed in a single module. Provided that a large number of modules can be afforded, $N \gg 200$ can be executed in the same $3 * [200^2] * 200$ nSec = 24 mSec.

3.5.2 Communications

The concept of modular communication assumes that the sensor data can be communicated to the modules fast enough so as to not slow down the modules significantly. The modular computational rate R_m and the system computation rate R_s are given by:

$$R_m = \frac{W}{L * C * T}$$

$$R_s = \frac{M * W}{L * C * T}$$

Assume that the communication rate would need to be 100 times the computation time for a module in order to not “significantly” slow the system (1% of computation time lost for communication). This implies a data transmission of $M * W * N$ bits in $L * C * T$ time. The system bus rate R_b must be 100 times greater than the ratio of these two numbers:

$$\begin{aligned}
 R_s &= 100 * \left[\frac{M * W * N}{L * C * T} \right] \\
 &= 100 * \frac{M * W}{L * N * T} \\
 &= 13 \text{ Mbits/Sec} \\
 M &= 1.
 \end{aligned}$$

This estimate implies that current bus systems can handle the $M=1$ case (e.g., VMEbus) up to $M=16$ (e.g., Nubus) cases, or, equivalently, up to the $N = 800$ component range without starving the modules for data. What about the more exotic digital simulators – e.g., the systolic system where $M=N$ and each memory stores a row or column of the matrix? Now the data rates move toward the gigabit/sec range – clearly beyond the bus systems now in use. This points up an important problem which the simple local memory with dedicated processor architecture does not solve: communication load between networks becomes severe when the number of processors per network gets as large as the number of components. The implication is that the data can no longer be brought in serially when M gets to be the same order of magnitude as N . This problem gets totally out of control when one contemplates analog hardware for which the computation of the whole matrix-vector product can take place in times on the order of a microsecond. The communication issue from the sensor to a single-layer neural network and from layer to layer in multi-layer neural networks is a crucial issue for future system architecture research.

3.5.3 Learning vs. Execution

Neural network models call for learning as well as execution operations, and it is really the learning procedure which is of most interest in this Study. To envision the learning simulation, again consider the auto-associative case, learning H vectors with N components with an Hebb learning rule (outer product); also return to the local memory/dedicated CPU hardware architecture. Learning takes an input vector and creates matrix elements. The important feature of Hebb-like learning rules is that the two matrix element indices and matrix element values are uniquely determined by the input vector index and component value:

$$\text{change} [\text{matrix element}_{ij}] = \text{vector}_i * \text{vector}_j$$

Note that to compute the i,j th matrix element, one only needs the i and j th elements of the input vector. Thus one can compartmentalize the learning procedure just as one compartmentalized the execution procedure. Consider the case of $N=800$ with 16 modules and $c = C^{\frac{1}{2}} = 200$. The modules would be loaded with

different quarters of the input vector in accordance with the prescription below:

$0 < i < c$	$c < i < 2c$	$2c < i < 3c$	$3c < i < 4c$
$0 < j < c$	$0 < j < c$	$0 < j < c$	$0 < j < c$
$0 < i < c$	$c < i < 2c$	$2c < i < 3c$	$3c < i < 4c$
$c < j < 2c$	$c < j < 2c$	$c < j < 2c$	$c < i < 2c$
$0 < i < c$	$c < i < 2c$	$2c < i < 3c$	$3c < i < 4c$
$2c < j < 3c$	$2c < j < 3c$	$2c < j < 3c$	$2c < j < 3c$
$0 < i < c$	$c < i < 2c$	$2c < i < 3c$	$3c < i < 4c$
$3c < i < 4c$	$3c < i < 4c$	$3c < i < 4c$	$3c < i < 4c$

The upper left module can compute matrix elements with indices from 0,0 to c,c. The lower left module can compute matrix elements with indices from 0,3c to c,4c. The point is that local memory will again suffice, and that the local memory organization is exactly that which is necessary for the execution phase.

The operations involved in learning are multiplication and summation to an already existing memory value in the simplest Hebb case. In more complex cases, such as Widrow-Hoff error-correction learning, one wants to compare the vector to be learned with the operation of the previously learned matrix. It is proposed that since the fact that the memory organization permits the matrix-vector multiplication, this operation be performed in the execution mode of the system. The resultant vector is stored in the CPU local memory along with the input vector. The outer product procedure is then performed on the error signal. This extra processing will increase L from 3 to perhaps 15, but at no extra cost in memory or CPU silicon real estate. Thus it is estimated that the learning of one vector will take on the order of two-to-five times longer than execution of one vector, independent of N and independent of initial cost.

3.6 SUMMARY

The simulation facilities currently available are limited to software running on general-purpose computers or attached processors for micro- and mini-computers. The workstation environment, which includes hardware generally under \$100,000, has the capacity of 10 M multiply-accumulate operations/second, although the majority of the processing power is under 500 K multiply-accumulate operations/second. The recent advent of special-purpose pipeline- and multi-processors accelerators have been the primary reason for this surge in capacity.

Supercomputers and massively parallel machines, such as the Connection Machine and the Cray XMP, have the capacity to process up to 80 M multiply-accumulate operations/second. The cost of such processing power is well over \$100,000 and usually over \$1 million. The high price tag of these larger machines restricts their availability to large industry and government laboratories.

The growth in processing power will be gradual over the next three years and will be motivated by current market forces. The Simulation/Emulation Tools and Techniques Panel expects no new technology to create major increases in computation capacity. The development of current technology will provide future enhancements. Limitations in accessing large blocks of memory will continue due to the constraints on communication bandwidth (i.e. bus speeds).

4. APPLICATION COMPUTATIONAL REQUIREMENTS

This chapter will attempt to define some of the computational requirements for four typical examples of DoD-related problems: signal processing, robotic arm movement, speech, and pattern recognition/vision. Using currently investigated algorithms with data rates from functional sensors, an *order of magnitude* estimate of the hardware requirements is ascertained. The Simulation/Emulation Tools and Techniques Panel stresses that these requirements are articulated only to place the current simulators in a framework of possible DoD needs.

This chapter will end with a comparison between the DoD examples and currently available hardware using storage and multiply-accumulate (interconnects per second) benchmarks. Additional comparisons are made with the 11 functional applications presented in detail by the Neural Network Study's System Applications Panel in Part IV of this Report.

4.1 SIGNAL PROCESSING

A possible application for a neural network is the *radar pulse identification* problem. Signal traces are measured every 50 μ Sec. A trace is divided into 128 bins. One solution technique could be a three-layer backpropagation network with 128 hidden units and 128 output units. Full connectivity will be assumed. Therefore, there are 384 neurons and 32K interconnections (see Figure 4-1).

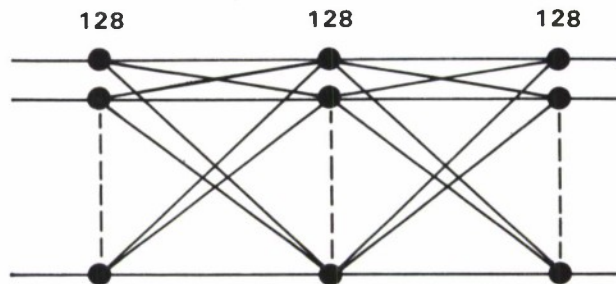


Figure 4-1. Signal Processing Application Example.

The computational requirements for the network shown in Figure 4-1 depend upon the data rate and the required number of presentations of a pattern during learning. Therefore, two sets of requirements are necessary: one for execution, the other for training. In execution mode, the network must process the input trace within 50 μ Sec. This leads to a computation rate of

$$\frac{2 * 128 * 128(\text{interconnects})}{50 * 10^{-6}(\text{sec})} = 625 M \frac{\text{interconnects}}{\text{sec}}$$

Many variations of the preceding calculations can be performed for different data rates. The relaxation of the realtime performance for developing and testing of the network also changes the computation rate values. Table 4-1 contains a list of computational requirements for various levels of development.

Training a multi-level network requires repeated presentations of the training set. For this example, a typical training set is 250 traces, which can arrive as often as every 5 mSec. The typical number of presentations required to train a multi-level backpropagation network is equal to the number of traces in the training set. Therefore, the network must process 250 traces 250 times each in 5 mSec. The computation rate for such a system is

$$250(\text{traces}) * 250(\text{presentations}) * \frac{2 * 128 * 128(\text{int})}{5.0 * 10^{-3}(\text{sec})} = 380G \frac{\text{int}}{\text{sec}}$$

The training of the network places a computational load on the hardware that is over 600 times that of the feedforward execution of the network! A list of computational requirements for various training set sizes for development, testing, and realtime performance of the system can be found in Table 4-1. The

Stage	Training Set	Iterations	Time	Requirement $\frac{\text{iterations}}{\text{second}}$
Development	200	1	1 hour	3.5K
	200	200	1 hour	700K
Testing	1000	1	1 hour	18.4K
	1000	1000	1 hour	18M
Execution in Field	1	1	50 μ Sec	625M
Learning in Field	250	250	5 mSec	380G

Table 4-1.

Computational Requirements for Signal Processing Application

requirements for developing and testing a network of the class just described have been met at the current stage of available hardware used for network simulation given in Chapter 3. The hardware for the realtime execution and learning stages of the system are not currently available.

4.2 ROBOTICS

Six-degree arm movement under varying loads is a unique problem that must be solved in robotics. There are examples by Albus, Kuperstein, and Reeke in the neural network literature that pose solutions to that problem [1,8,12]. The networks under study can be described as a three-layer network as shown in Figure 4-2. The network contains 312 neurons and 3,600 interconnects. Assuming a response time that may vary between 1 second and 10 milliseconds (depending upon the dynamics of the robotic arm), the computational requirements will be between 3.5K $\frac{\text{interconnects}}{\text{second}}$ and 350K $\frac{\text{interconnects}}{\text{second}}$. Most, if not all, of the currently available simulators can meet these requirements.

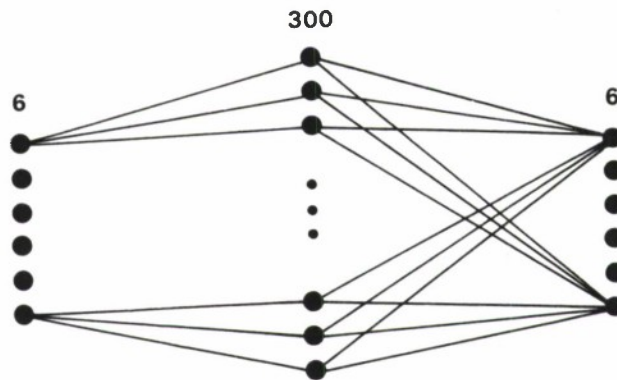


Figure 4-2. Robotic Arm Movement Example.

4.3 SPEECH

Isolated Word Recognition (ISW) is an ongoing research effort throughout the United States. One neural network approach to the problem is shown in Figure 4-3. An audio signal is first preprocessed to produce an 25-element input vector. This is currently done in real-time (every 10 milliseconds) with conventional hardware. Each word in the vocabulary consists of a network of 10 neurons. Each of the 25 inputs is connected to each neuron in every word. There is also a local connectivity in the 10-neuron network that represents a word. Therefore, the connectivity of the network is

$$25(\text{inputs}) * 10 \frac{\text{interconnects}}{\text{word}} * N(\text{words}) = 280 N(\text{interconnects}).$$

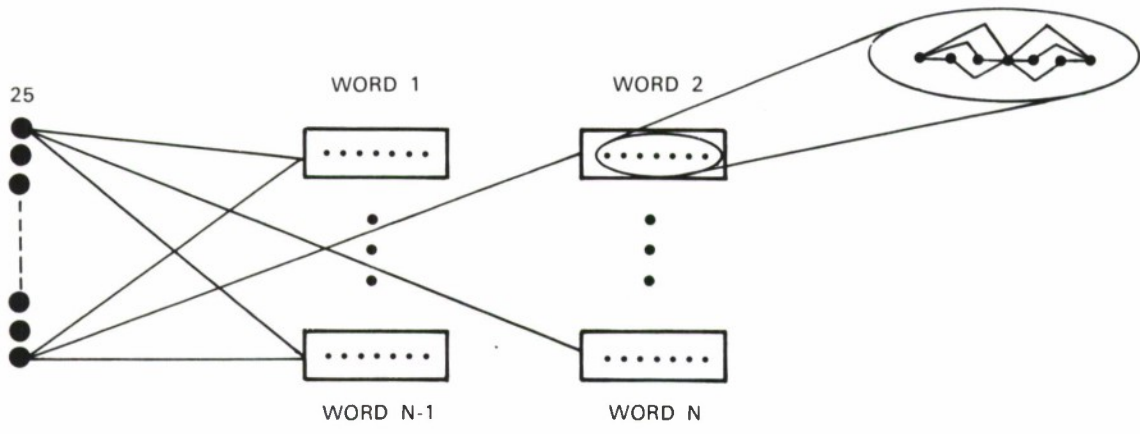
The connectivity is a linear function of the number of words in the vocabulary. The training or testing sets for the ISW are usually a set of 40 exemplars each for 10 speakers under 10 conditions. This requires that there be 4,000 input vectors for each word in the vocabulary. The previous requirement for the number of presentations to be equal to the number of input vectors is not required for the this network. The learning rules used require approximately five presentations.

The speech input for a 10,000 word vocabulary represents a training set of 40 million vectors. Since each vector arrives every 10 milliseconds, that represents a training set spanning 111 hours of realtime speech! Realtime performance may not be sufficient for the development and testing stage of the network.

A list of the computational requirements for different sized vocabularies and performance time (realtime and two-hour, respectively) is given in Table 4-2.

4.4 PATTERN RECOGNITION – VISION

Perhaps the most interesting of applications, and among the most useful, is vision. A large amount of defense resources is expended on sensors producing imagery using single or multiple wavelengths. Since an autonomous vision systems is a worthwhile goal, this section will concentrate on a theoretical model of image preprocessing with pattern classification.



10 SPEAKERS, 40 EXEMPLARS, 10 CONDITIONS = 4 K INPUTS/WORD
 REAL TIME PERFORMANCE (10 ms) = 25(10 × N) + 30N = 280N

Figure 4-3. Isolated Word Recognition Example.

102649-3

	Vocabulary Size		
	100	1,000	10,000
Interconnects	28K	280K	3M
Int/sec (RT)	3M	30M	300M
Int/sec (2 Hours)	1.4M	140M	14G

Table 4-2.

Computational Requirements for Speech Application

The proposed vision system consists of a segmenter, two image preprocessors, a feature extractor, and an associative memory. This system, shown in Figure 4-4, is for two-dimensional rotation- and scale-invariant pattern recognition. Table 4-3 lists the components of the system and their respective storage and computation requirements.

<i>Section</i>	<i>Neurons</i>	<i>Interconnects</i>	<i>Cycles</i>
Input	NM		
Segmenter	NM	4.9 NM	10
Sequencer	256^2	N/A	1
	(NM)	N/A	1
MRF	256^2	$9 * 256^2$	100
	(NM)	9NM	100
BCS 1	$4 * 12 * 256^2$	$20 * 4 * 12 * 256^2$	10
	$4 * 12 * NM$	$20 * 4 * 12 * NM$	10
BCS 2	$4 * 12 * 256^2$	$50 * 4 * 12 * 256^2$	10
	$4 * 12 * NM$	$50 * 4 * 12 * NM$	10
BCS 3	$4 * 12 * 256^2$	$100 * 4 * 12 * 256^2$	10
	$4 * 12 * NM$	$100 * 4 * 12 * NM$	10
Combiner	256^2	$25 * 3 * 12 * 256^2$	1
	NM	$25 * 3 * 12 * NM$	1
Log/Polar FFT	128^2	$512 * 128^2$	1
	128^2	$2 * N * 128^2$	1
Feature Map	512	$512 * 128^2$	1
	512	$512 * 128^2$	1
Associate Memory	10	$512 * 10$	1
	10	$512 * 10$	1

Table 4-3.

Computation and Storage Requirements for Vision Example

The total number of neurons, interconnects and interconnect calculations per input becomes

$$\begin{aligned} \text{Neurons} &= NM + 147 * 256^2 + 128^2 + 10 \\ &= 148 * NM + 128^2 + 10 \end{aligned}$$

$$\begin{aligned} \text{Interconnects} &= 4.9 * NM + 9069 * 256^2 + 1024 * 128^2 + 5120 \\ &= 4.9 * NM + 9069 * NM + (2N + 512) * 128^2 + 5120 \end{aligned}$$

$$\begin{aligned} \text{Computations} &= 4.9 * NM + 83,400 * 256^2 + 1024 * 128^2 + 5120 \\ &= 4.9 * NM + 83,400 * NM + (2N + 512) * 128^2 + 5120 \end{aligned}$$

where N and M are, respectively, the number of rows and columns in the input image. The values for the particular sections are included in order to reflect the approximate number of neurons and interconnects

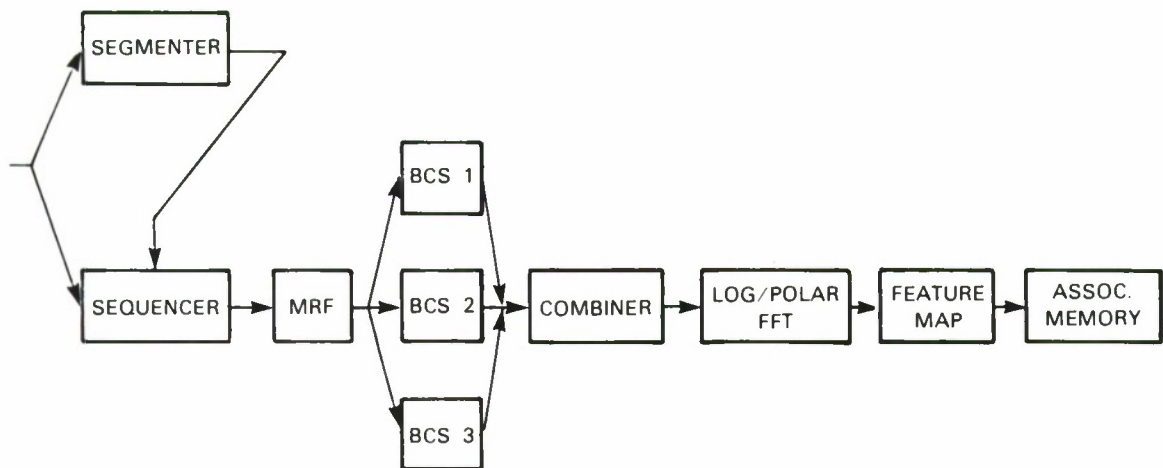


Figure 4-4. Vision Example Schematic.

necessary to complete the task. The values are shown only to assess an order-of-magnitude computational power requirement. The upper row of values for each section, both in the table and in the totals noted above, represents the computational power required if the image can be segmented down to a 256-by-256 pixel region of interest. The lower row contains the power requirements when segmentation cannot be done and the entire image must be processed. These two cases, then, are used to bound the computational requirements.

Two examples of imagery that may have use in the DoD community are from **Forward Looking InfraRed (FLIR)** and **InfraRed Search and Track (IRST)** sensors. The following is a list of pertinent information on data rates that have been assumed for the sake of demonstration.

- **FLIR**

1. 2° by 2° FOV
2. 10 μrad resolution
3. 350 pixel by 350 pixel image
4. 30 frames per second

- **IRST**

1. 4° by 360° FOV
2. 10 μrad resolution
3. 700 pixel by 63,000 pixel image
4. 1 frame per second

Using these figures with the equations listed above produces a lower limit of 5 Giga (billion) $\frac{\text{interconnects}}{\text{second}}$ and an upper limit of 3 Tera (trillion) $\frac{\text{interconnects}}{\text{second}}$. Table 4-4 lists the number of neurons, interconnects,

	FLIR	IRST
Neurons	9.3 M	51.2 M
	17.3 M	6224.4 M
Interconnects	583.4 M	788.9 M
	1079.0 M	372.7 G
Computation Rate	5229.0 M	5434.6 M
	9762.7 M	3425.5 G

Table 4-4.

Simulation Requirements for Vision System Example

and computation rates required using the above figures. These figures are for realtime processing and may be relaxed for research and development purposes.

4.5 APPLICATION EXAMPLES AND CURRENT HARDWARE

The comparison of future simulation requirements with current capabilities is shown in Figure 4-5. It is readily seen that the robotic arm example can be implemented with current hardware. The signal processing and speech applications can be minimally implemented today; advanced simulation engines are needed to completely explore those applications.

It is not surprising that vision applications are beyond the capabilities of available simulators. Data rates for vision sensors are usually much larger than most other sensors and require more processing steps. Simulators with an increase in speed and storage capacity in excess of two orders of magnitude greater than current systems are required for such applications.

4.6 COMPUTATIONAL REQUIREMENTS OF CURRENT APPLICATIONS

The System Applications Panel has focused on 11 applications of neural networks presently in various stages of development. This section will list the computational requirements for 10 of them. Every application is implemented using simulators. In fact, all of the simulations are on general-purpose computers.

The 10 applications are:

- Widrow's and Tolat's Broom Balancer
- GTE's Process Monitor
- Intel's Isolated Word Speech Recognizer
- Sejnowski's and Gorman's Sonar Target Classifier
- Nestor's Mortgage Underwriting and Risk Assessment System

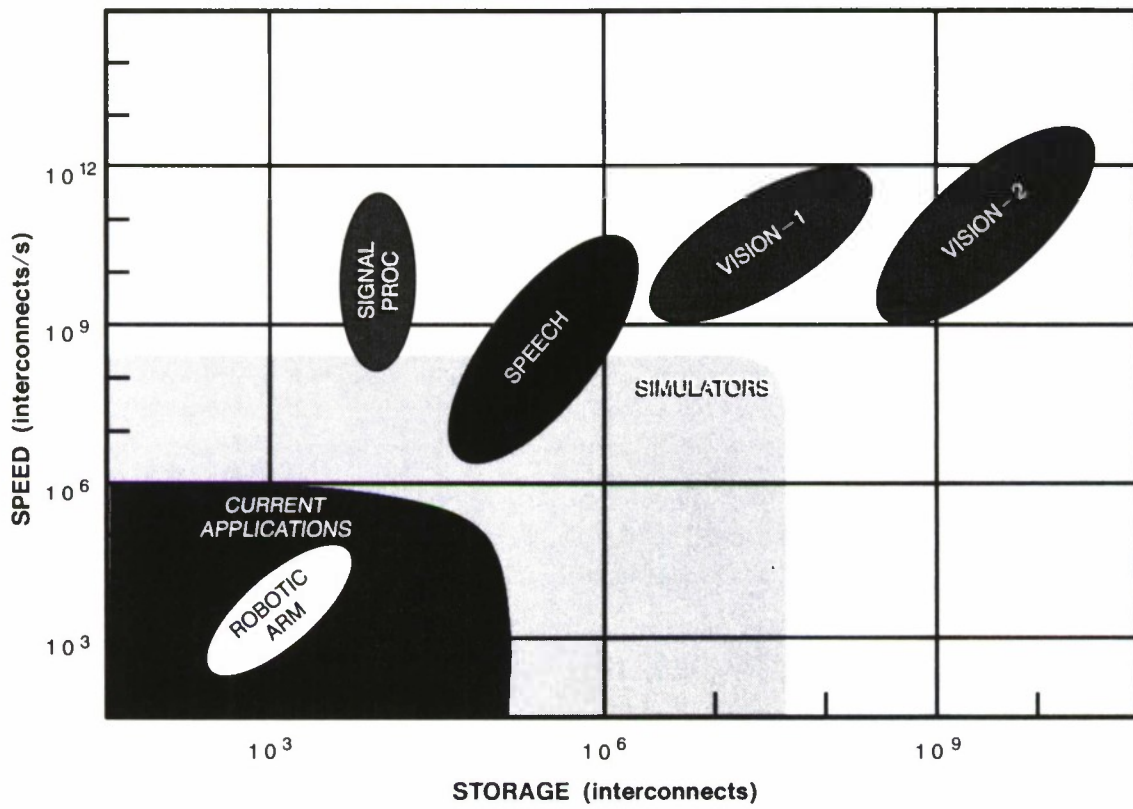


Figure 4-5. Hardware Capabilities and Application Example Requirements.

103258-1A

- Martin Marietta's Fork Lift Robot
- Hughes' Autonomous Target Recognizer
- Lincoln Laboratory's Neocognitron Simulation
- Kuczewski's Multi-Target Tracking System
- SAIC's Model Retina.

The requirements for each of the above applications was computed using the size and update rate required by the developer. Table 4-5 lists the parameters used for the calculation and the simulation requirements. The requirements are plotted in Figure 4-6 against current simulators.

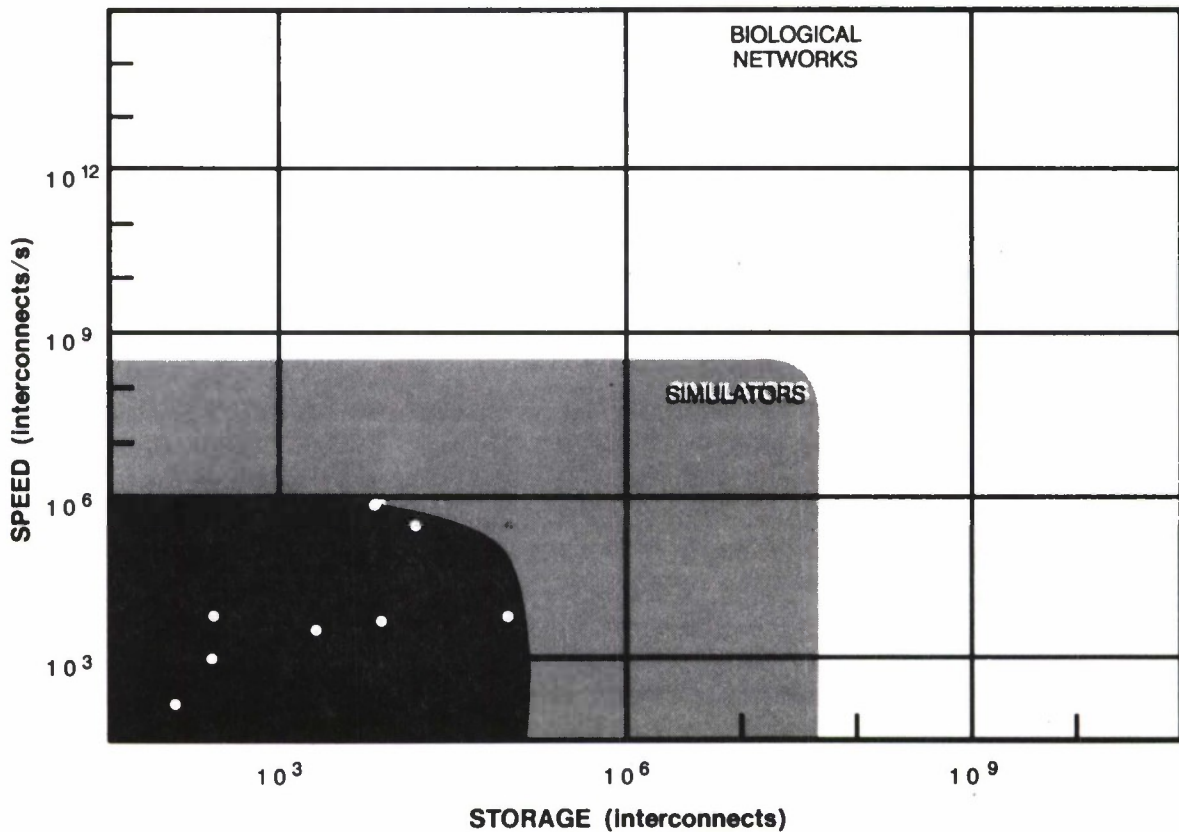


Figure 4-6. Simulation Requirements of Current Applications.

Although the current requirements are modest, less than 300K $\frac{\text{interconnects}}{\text{second}}$, the eventual implementation could be more stressed. For example, the Neocognitron, Multitarget Tracking, and Model Retina are all using update rates of less than or equal to one per second. These rates are not indicative of fielded systems but are limited by the speed of the simulator. These rates would be increased by an order of magnitude or more for fielded systems. The ATR application will require processing rates far in excess of the other

<i>Application</i>	<i>Input Size</i>	<i>Update Rate (sec⁻¹)</i>	<i>Interconnects</i>	<i>$\frac{\text{interconnects}}{\text{second}}$</i>
Broom Balancer	55	10.00	110	1K
GTE Processor		.03	200	<1K
Speech Recognizer	16	.50	6400	12K
Sonar	60	4.00	1488	6K
Mortgage Underwriting		.03	4M	14K
Fork Lift Robot		5.00	100	<1K
ATR	?	?	?	?
Neocognitron	16K	<.01	200M	100K
Multitarget Tracking	144	1.00	300K	300K
Model Retina	64K	1.00	192K	192K

Table 4-5.

Computational Requirements for Selected Applications

applications. That application is currently under study, which explains the absence of definitive numbers for that application.

The bulk of the current work has been limited to systems with modest simulation requirements. There are two possible explanations for this trend: the scaling of networks to large applications is not well-understood and/or high performance simulation tools are not available.

4.7 SUMMARY

Eventual computational needs can be estimated by sizing current and future applications. Current applications are restricted to the use of currently available hardware and therefore are modest in size and speed requirements. Future applications have been placed into four generic areas: robotics, signal processing, speech, and vision. The vision application was separated further into high data rate and low data rate systems. Computation needs ranged from low storage and low speed, to low storage and high speed, to high storage and high speed.

Current hardware cannot meet the requirements of existing neural network algorithms used to address speech and vision problems. Although the simulation requirements for signal processing are within the capacity of current hardware, realtime requirements for high-speed signals cannot be satisfied. Thus there is a need for greater computational capability if the hard application problems outlined in this chapter are to be tackled.

5. CONSIDERATIONS FOR FUTURE SIMULATIONS

5.1 PROJECTIONS OF STANDARD HARDWARE

One theme which has pervaded the Neural Network Study is that neural network algorithms can be considered to be most like the signal processing algorithms which are more familiar to traditional computer designers. Thus it should come as no surprise that most of the neural network simulators work in a traditional signal processor mode, using standard CPUs or application-specific digital signal processors. The key to many of these designs is the use of many CPUs with local random access memory.

If one looks at Figure 4-5, one can see that it is a straightforward process to project the growth of the simulator region on the speed-versus-storage-capacity coordinates over time by using projections of digital signal processors and random access memory. Both of these semiconductor parts have shown an exponential growth in functionality over the past few decades, and technical indications are that another order of magnitude in speed and capacity can be expected by the end of this century. This suggests that traditional digital simulators will be able to handle all but the very fastest signal processing jobs and 25% or more of speech applications as well. Only vision applications appear to be beyond the capability of the evolution of current technology during this century.

5.2 DESIGN CRITERIA

5.2.1 Processor Speed Versus Memory Access Time

In the above discussion of technological issues, it is assumed that memory access times will at least keep up with the maximum speed of the CPU. This implies, of course, that care be taken in the design cycle to insure this. One matter of concern is that the smaller device geometries and larger areas of RAM chips means larger resistances in the bus lines of the RAM. This will ultimately limit the access time of data due to charging times of the bus lines.

5.2.2 Processor Speed Versus Memory Access Capacity

The speed-versus-storage chart indicates that some applications are memory-intensive where lower rates are acceptable, while other applications require ultra-fast speeds with limited memory requirements. There are two methods for achieving speed – use more processors/memory systems or faster processor/memory systems – e.g., GaAs. Care must be taken, of course, that the input bus can support the serial data rate. The secret to intensive memory access is sufficient processor address capacity, within the bounds of the local memory philosophy. Fortunately, the need for larger address space is well understood by CPU designers.

5.2.3 Cycle Time Versus Number of Neurons with Respect to Interconnections/Sec

This technical issues discussion also assumes that the input data rate is equal to or greater than the processing capability of any single-processor memory system. This is generally the case for physical neurons which have millisecond processing times. With semiconductor neural network simulations, one generally exploits the faster cycle times to compute many nodes in one processor. The problems of interest to this Study are sufficiently large in terms of node number, or speed, requirements that there does not appear to be need to worry about under-utilizing a single-processor memory system.

5.3 VISION-CAPABLE SIMULATOR DEVELOPMENT

The technology for producing high-speed, vision-capable simulators exists. The use of charged-coupled device (CCD) technology can produce a single chip multiply/accumulate figure into the 10^9 to 10^{10} range. The number of interconnections that can be addressed on-chip severely limit the size of the networks. Off-chip memory can be used with a penalty.

One example of such a technology is a proposed device consisting of 1,024 CCD multiplying digital-to-analog converters (MDACs) with a 10 Mhz clock. This device, shown in Figure 5-1, can produce 10 G multiply/accumulate operations every second. The input to the chip is an analog stream of neuron state values of 8-bit accuracy. The digital weights are stored on-chip, also with 8-bit accuracy, with up to 20 weights per neuron state. Therefore the chip, as configured in Figure 5-1, computes the value of one neuron (using the upper eight bits of the 16-bit result) with 1,024 interconnects at each cycle and can store enough weights for up to 20 neurons. The total capacity of the chip is 20K interconnects. Other configurations can be used with the limitation being 20K interconnects.

The use of off-chip memory can increase the capacity of the *system* with a penalty in the throughput. Obtaining the 8-bit digital weights using 32 parallel lines at a current limit of 100 Mhz requires

$$\frac{8(\text{bits}) * 1024(\text{interconnects})}{32(\text{lines}) * 100(\text{Mhz})} = 2.56 \mu\text{sec},$$

or the weights can be updated at a rate of 400 Khz. The computations proceed at a rate of 10 Mhz and the weights can be read at a rate of 400 Khz. The overall performance decreases by a factor of 25 when using off-chip memory. Still, the overall simulation capacity is 400 M int/sec!

5.4 ACCESS TO DATABASES

The Study heard from several researchers who felt the need for DARPA to serve as a source of information regarding neural networks. Need was found for algorithms for use by naive users as well as for serious databases for testing and teaching big DoD applications.

101079-3

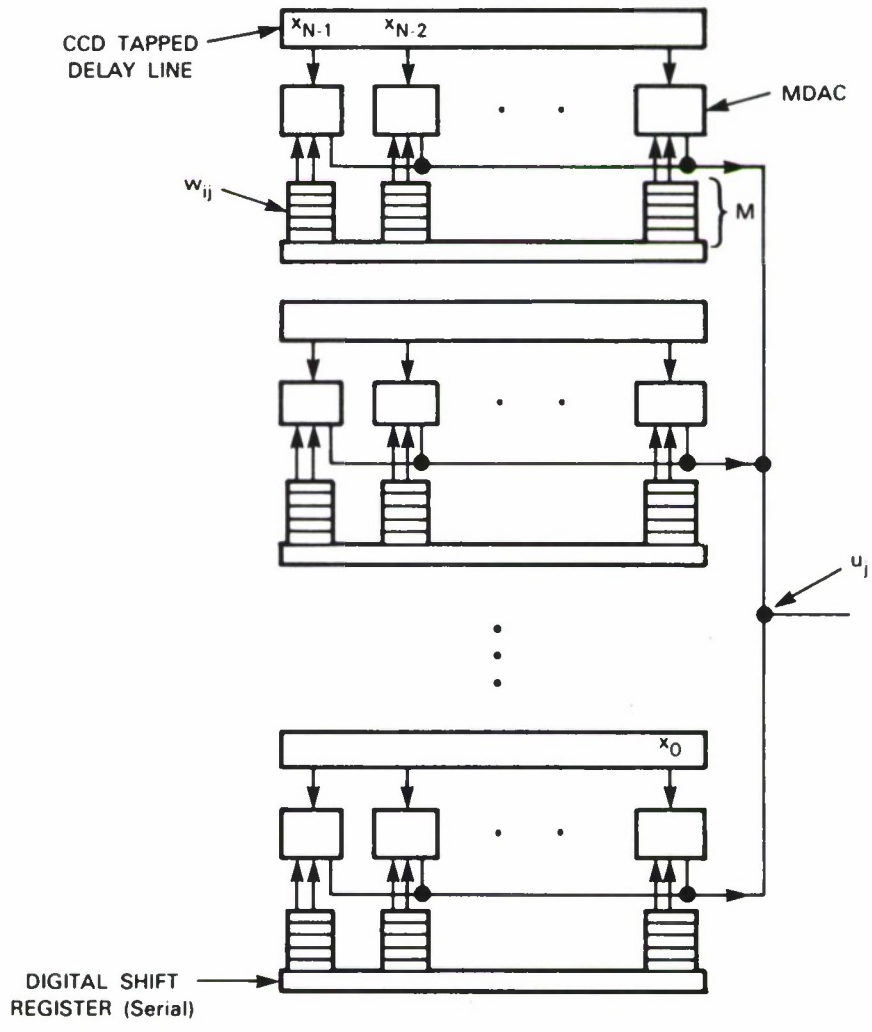


Figure 5-1. CCD Multiply/Accumulate Accelerator.

5.4.1 Algorithms (Standard or NN)

Software is traditionally considered to be proprietary and/or secret. Thus it would be impossible for DARPA to be a source of state-of-the-art software to the general community. There is the simpler case of artificial neural network novices wanting to try popular networks on a problem they are expert in – e.g., neurobiologists. There is also the desire for well-documented software to be used to judge one hardware system versus others. Naturally, such testing need not be proprietary. The desire here is for DARPA to serve as a software library, perhaps in the fashion of an electronic bulletin board, as DARPA has analogously done with the MOSIS foundry.

5.4.2 Data (Vision, Speech, Signal Processing, etc.)

The need for easily accessible sensor databases appears to be much more important than the need for a software library. DARPA will often find scientists reporting greatly improved software or hardware being tested on a particular vendor database. The only way to fairly determine the veracity of such claims is to have them demonstrated on a common database reflecting real DoD problems (visible images, IR images, laser images, radar tables, etc.) Another need for large databases stems from the requirement of large learning systems for very large training sets. This is often cited as one of the requirements of the backpropagation method.

Many research facilities, especially universities, do not have access to such databases. DARPA could serve an important need by providing such a service.

6. CONCLUSIONS CONCERNING NEURAL NETWORK SIMULATION/EMULATION TOOLS AND TECHNIQUES

6.1 SUMMARY

This report focused on the requirements of neural network simulations from the standpoint of algorithm development and near-term application implementations. Throughout the report, the Simulation/Emulation Tools and Techniques Panel has pointed out that simulations are not one-sided. Two measures of merit were selected for comparing simulators, but never was it assumed that these measures cover all possible situations. Hardware development has been brisk recently but by no means complete. There remain many uncertainties in defining the requirements for neural network simulations.

Still, there *are* a number of certainties this Panel can point to.

Neural network simulations are comprised primarily of multiply-accumulate or interconnect operations. For systems of difference, differential, or algebraic equations, the dominant operation is in the interconnect calculations. The ratio between interconnect and non-interconnect operations can be many orders of magnitude. However, the solution techniques for systems not defined by difference equations can be critical with respect to their numerical worthiness. The current level of research into the manner in which to simulate such systems is high – but generally unused within the neural network community. This Part of the Neural Network Study Report outlined a few of the techniques available.

The current surge in development of hardware and software simulators is encouraging. Taking advantage of advances in digital signal processing hardware has led to board-level systems to simulate difference equation networks. Processor speeds up to five to ten million $\frac{\text{interconnects}}{\text{second}}$ are not uncommon for micro-computer plug-in boards. The use of supercomputer and parallel processor hardware has bridged the 100 million $\frac{\text{interconnect}}{\text{second}}$ level – but at a significant cost. Such systems are not easily accessible by most researchers.

Software development will continue to keep pace with the hardware development with a finite lag time. Commercial software available today allows for small network development at low costs. The commercial marketplace and individual needs will drive further development in both the commercial or private arenas.

One of the reasons for the current advancements in neural networks has been the availability of affordable, easy-to-use computing facilities. Current applications have exploited these facilities. When the capacity of these facilities is compared to possible applications in signal processing, robotics, speech, and vision the facilities are deficient. Those problems are hard. The data rates are enormous and on-line learning pushes the effective data rates orders-of-magnitude higher. *Giga-connection* (1 billion $\frac{\text{interconnects}}{\text{second}}$) simulators with tens of millions of interconnections are needed.

The next generation of simulators will come from one of three directions:

- Incremental speed increase in devices,
- Increase processor parallelism, or
- New processor technology.

Each of these directions will increase the overall throughput of the simulator. Two important restrictions can limit the development of ultra- fast computing engines: the available local memory, and capacity for local learning. As the number of neurons and interconnects increases with regard to the size of the application, the amount of memory required to store the interconnect values increases. If that memory cannot be stored locally with every processor, then the processor must access memory external to itself – and that slows the overall speed of the simulator. Conversely, if the processor cannot update the weights on-chip, then it must access a processor and/or memory off-chip – and that will also slow the processor.

6.2 CONCLUSIONS

Simulations will play an important role in the development of neural network algorithms and applications. In fact, most short-term applications will be implemented as a simulation. This leads the Simulation/Emulation Tools and Techniques Panel to suggest that *the development of simulation engines specific to future application needs should be sponsored*. This can be accomplished through the understanding of the limitations of current simulators.

Neural network algorithms are presented in numerous mathematical forms. One of the popular forms is as a set of coupled differential equations. Current hardware accelerators used for neural network simulations do not easily allow (if at all) the use of such equations. Thus, *it is recommended by the Panel that the development of hardware and software simulators using the differential equation descriptions be encouraged*. A natural starting point would be to exploit current parallel processor solution techniques.

Simulations are primarily used to understand the dynamics of a particular network and for modest implementations. As the need for high-speed, low- cost, low-power, and small-size implementations increases, so will the need to understand the characteristics of the devices used in these advanced implementations. It will be through simulations that the characteristics of implementations are studied and understood. Future simulation requirements must, therefore, account for such simulations of implementations. This type of simulation is more demanding than those used chiefly to understand network dynamics or for small-scale implementations because the latter type of simulations must encompass the dynamics of both the network *and* the devices.

The limitations of the hardware with respect to their possible applications must be overcome if research in those areas (signal processing, speech, and vision) are to be explored. Hardware accelerators that address the current bottlenecks in the size of the networks (memory) and training time (learning) need to be developed. *Two orders of magnitude of increased capacity (both in speed and storage) is necessary to allow the development of large-scale applications; it is advised that such hardware and software developments be pursued*.

The availability of high-end simulation tools to the neural network research community is critical for the development of near-term applications. Many researchers throughout the study noted that the lack of inexpensive and easily accessible simulation facilities inhibit their research. Current hardware accelerators have brought a significant increase in computational power to researchers. However, *if the development of neural network capability in signal processing, speech, and vision applications become a focal point of DoD interest, then simulation facilities beyond hardware accelerators are necessary for researchers; so*

the Simulation/Emulation Tools and Techniques Panel recommends that access to high-end computational facilities be available for large network (e.g., vision) research and development.

REFERENCES

1. J. Albus, "A theory of cerebellar function," *Mathematical Bioscience*, vol. 10, pp. 25–61, 1971.
2. W. L. Briggs, "A multigrid tutorial," *Society for Industrial and Applied Mathematics*, 1987. Philadelphia, PA.
3. G. A. Carpenter and S. G. Grossberg, "A neural theory of circadian rhythms: the gated pacemaker," *Biological Cybernetics*, vol. 48, pp. 35–49, 1983.
4. C. W. Gear, *Numerical Initial Value Problems in Ordinary Differential Equations*. Englewood Cliffs, NJ: Prentice Hall, 1971.
5. S. G. D. Geman, "Stochastic relaxation, gibbs distribution, and the bayesian restoration of images," *IEEE Transactions in Pattern Analysis and Machine Intelligence*, vol. 6, pp. 721–741, 1985.
6. P. E. Gill, W. Murray, and M. H. Wright, *Practical Optimization*. New York: Academic Press, 1981.
7. G. Hall and J. W. Watt, *Modern Numerical Methods for Ordinary Differential Equations*. Oxford: Clarendon Press, 1976.
8. M. Kuperstein, "Adaptive visual-motor coordination in multijoint robots using a parallel architecture," in *IEEE International Conference on Automatic Robotics*, (Raleigh, NC), pp. 1595–1602, March 1987.
9. J. M. Ortega and W. C. Rheinboldt, *The Iterative Solution of Nonlinear Equations in Several Variables*. New York: Academic Press, 1970.
10. P. Penz and R. Wiggins, *AIP Conference Proceedings*. 1986. vol. 151, (J. Denker, ed.).
11. L. R. Pezold, "Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations," *Siam Journal of Scientific and Statistical Computing*, vol. 4, pp. 136–148, 1983.
12. G. N. Reeke and G. M. Edelman, "Selective neural networks and their implications for recognition automata," *International Journal of Supercomputer Applications*, vol. 1, pp. 44–69, 1987.
13. T. Chan, UCLA. Private Communications.

APPENDIX A PARALLEL PROCESSING HARDWARE

A.1 BUTTERFLY

The Butterfly Parallel Processing Computer is a product of Bolt Beranek and Newman, Inc. The basic configuration of this MIMD computer starts with four processors; it can be expanded in single-processor increments to 256 processors which yield up to 250 MIPS. It has a butterfly switch for interconnecting all processors described below.

Each processor and its memory are located on a single board called a Processor Node (PN), which is the basic computing element of the Butterfly computer. Each PN uses a standard Motorola MC68020 microprocessor, capable of executing one million instructions per second. The microprocessor is augmented by the MC68881 floating point computational unit. Each PN also contains a microcoded control processor that provides inter-processor communication, synchronization, and support for parallel processing.

Each node may have memory size from one Mbyte to four Mbytes, with a maximum system-wide shared memory of one Gbyte. It can independently execute its own sequence of instructions, referencing data as specified by the instructions. Though memory is local to the PNs, each processor can access remotely any memory in the system using the Butterfly switch.

The Butterfly switch implements inter-processor communication using techniques similar to packet switching. Its topology is similar to that of the Fast Fourier Transform Butterfly and it implements a subset of the hypercube network. The switch has a latency $\log_2 N$ and the bandwidth through each processor-to-processor path in the switch is 32 Mbits/second.

The Butterfly I/O system is distributed among the PNs. Any node can have an I/O board that supports data transfer at a maximum rate of two Mbytes/second.

Butterfly application software development is done on a DEC VAX or Sun Microsystems workstation running Berkeley 4.2 UNIX. The system supports C and FORTRAN 77. A multiprocessing Common LISP system that will be compatible with the Symbolic 3600 series of LISP workstations is under development.

A.2 CAPP

The Content Addressable Parallel Processor (CAPP) is an SIMD computer built at the University of Massachusetts. Its goal is to explore the applications of content addressability and parallelism. It also serves as an interface between the host and secondary data storage devices. The central control unit is pipelined and instructions can be prefetched. It contains a ROM with commonly needed micro-coded instructions and a small program memory for storing user programs.

The parallel processor contains 262,144 cells arranged as a 512-by-512 array. Each cell consists of 32 bits of static memory, an ALU, and four one-bit static tags. One of the tags controls whether the cell is active.

Cells are connected to their nearest four neighbors. Because the pin count on the chip is limited, an 8:1 multiplexing scheme was used for communication between chips. Cells on the edge are processed in three ways, first as dead edges connecting no neighbor; secondly, wrapped around; or, third, wrapping around but offset by one cell to make a linear array.

A.3 CONNECTION MACHINE

Thinking Machines Corp. builds the Connection Machine computer, which has 65,536 processors provides a raw computing power of 1,000 MIPS. The computer can be configured to a much larger number of logical processors by creating a two-dimensional array of virtual processors on each physical processors.

The computer can be divided into four equal subparts. Each works at a separate front end, and each has a separate instruction stream executing in a quarter of the system's processors. Thus it becomes a MSIMD computer.

Each processor is a bit-serial processor that opens onto two data values specified by each instruction. It has four Kbytes of memory (32 Mbytes for the computer. The memory is divided into a data area and a start area, and it is bit-addressable. Sixteen physical processors are grouped on a chip.

The CM has two ways of communicating between processors: first, it is wired to its neighbors to the north, east, west, and south by the NEWS network; second, a "Boolean n-cube" network, the Connection Machine Router, connects each of the 65,536 physical processors to 16 other physical processors whose binary addresses are different in just one of the 16 bits. The Router allows a full message to be sent from any processors to 16 other physical processors whose binary addresses are different in just one of the 16 bits. The Router allows a full message to be sent from any processor to any other. The sender processor simply needs to know the address of the destination processor.

Data are exchanged between memory and the front end in three ways. "Read-slice" reads a single bit from the memory of each of series of consecutive processors. "Read-processor" moves a single field between the front end and a single processor. "Read-array" moves fields between the front end and set of contiguous processors.

A microcontroller expands macro-instruction from the front end into nano-instructions, which are broadcast to all virtual processors. Processors have the option of "sitting out" some instructions, depending on the one-bit Context Flag in each processor.

The Connection Machine provides an assembly-level language REL-2. It also supports parallel versions of C and LISP.

A.4 GAPP

The Geometric Array Parallel Processor, GAPP, was originally developed by the Martin Marietta Corp. and marketed commercially by NCR Corp.'s Microelectronics Division. It is a SIMD processor with 6-by-12 processing elements (PEs).

Each of the PEs contains an 1-bit ALU, 128 bits of RAM, and four registers. It operates with a 10 MHz clock and takes 25 clock cycles to add two 8-bit numbers. With all the PEs running, the processor can deliver 921 million additions per second. Seventy-two PEs are arranged in a 6-by-12 array in the current version of the chip. The chip is fabricated with a 3- μ m double layer metal CMOS process, and it is currently housed in a ceramic 84-lead pin-grid array.

Connecting each PE to its neighbors on the north, south, east, and west are bidirectional communication lines. In addition, a separate I/O communication bus allows data to be input from the south end of the array and output to the north without interfering with computations within the ALU.

The implementation of a control store lets the processor receive a set of instructions from the host and stores them, freeing the host for other tasks. The control store operates in conjunction with a sequence, which watches for and maintains the correct sequence as the processor performs its instructions.

The processor is programmed with a sequence of instructions that, when compiled by an assembler, directs the appropriate control signals to every cell in the array. Up to five commands (four for each of the four registers and one for the RAM) can be executed simultaneously on every instruction cycle. A software library of macro-cells forms the basis of a high-level command set for the processor.

A.5 iPSC

Intel Scientific Computers offers the iPSC as a multiprocessor system that can operate concurrently with as many as 128 independent processing units connected as a hypercube network.

Each node of the cube is a board-level microcomputer with high-speed versions of the Intel 80286/80287 microcomputer chip sets. It has its own memory 512 Kbytes, expandable to 4.5 Mbytes. Each node also contains eight bidirectional communication channels managed by dedicated communication co-processors. Seven of these channels are physically linked to other nodes and serve as dedicated channels. The eighth channel is a global Ethernet channel that provides direct access to and from the Cube manager for program loading, data I/O, and diagnosis. It has a 10-Mbit/second channel for internode serial communication.

The cube manager provides a high-level interface. It serves as the local host for the cube and provides the communication/control software and the system diagnostic facility. It runs the XENIX operating system, a version of UNIX, with LISP, FORTRAN, C, and Assembly language.

The iPSC-VX is a vector concurrent system built upon the basic architecture of iPSC. It couples a high-performance vector processor to each iPSC processing node, thus yielding a peak performance of 1,280 MFLOPS (on 32-bit data, or 424 MFLOPS on 64-bit data) on a 64-node iPSC-VX/64 version. Each node consists of 1.5 Mbytes, giving the whole system 96 Mbytes of memory.

A.6 MASSIVELY PARALLEL PROCESSOR

The Massively Parallel Processor (MPP) is a bit-serial SIMD parallel processing computer built for the NASA Goddard Space Flight Center by Goodyear Aerospace Corp. It has 16,384 processing elements (PEs).

The major blocks in the MPP are the array unit, array control unit, staging memory, program and data management unit, and an interface to the host.

Logically, the array unit contains 16,384 PEs arranged in a 128 row-by-128 column square array. Physically, the array unit contains an extra 128 row-by-4 column rectangle of PEs for redundancy. When a faulty PE is discovered, the processor bypasses all the PEs in that column (or row), and the topology is not disturbed.

Each PE is a bit-serial processor which uses a full adder and a shift register for arithmetic. Each PE has a RAM storing 1,024 bits. The address lines of all PEs are tied together so that memories are accessed by a bit-plane accessed by each PE. The PE memory can be expanded to 65,536 bits per PE or to 128 Mbytes total. The basic cycle time of the PE is 100ns. With all PEs operating in parallel, it delivers 3,000 MOPS for integer addition and 400 MFLOPS for floating point addition.

Each PE communicates with its four nearest neighbors. The edge connection is programmable. Between the top and bottom edges of the array unit, one can either connect them together to make the array look like a cylinder or separate them to make that array look like a plane. Similarly, the left and right edges can be independently connected together, or separated. When the left and right edges are connected together, one can either connect corresponding rows together or slide the connection by one row so the left PE of row i communicates with the right PE row $i+1$. Thus rows are connected together in a spiral fashion like a long linear string.

The array control unit has three subunits – the PE control unit (PCU) controls processing in the array unit, the I/O control unit manages flow of I/O data through the array unit, and the main control unit (MCU) runs application programs, performs necessary scalar processing, and controls the other two subunits. The division of responsibility allows array processing, scalar processing, and I/O to proceed simultaneously.

The staging memory buffers and reorders the bit-serial format of the array unit to the word format of the outside world. Data can be input at a rate of 160 Mbytes/second. An I/O rate of 320 Mbytes/second can be achieved when input and output proceed simultaneously.

The program and data management unit controls the overall flow of program and data in and out of the MPP. It also handles program development and diagnostics. It is implemented on a DEC PDP-11 minicomputer operating under DEC's RSX-11M realtime multiprogramming system.

A.7 RP3

Research Parallel Processor Project (RP3) was initiated at IBM. It is designed as a parallel MIMD computer for investigating both hardware and software aspects of parallel computation. It can be configured as both shared memory and local memory message passing paradigms, as well as mixtures of the two.

RP3 is designed to have up to 512 processor/memory elements (PME) with an interconnection network. A full system is expected to provide up to 1.3 GIPS, 800 MFLOPS, 1-2 Gbytes of main storage, 192-Mbyte/second I/O rate, and 13-Gbyte/second interprocessor communication capability.

Each PME contains a 32-bit processor, 4 Mbytes of memory, a 32-Kbyte cache, a floating-point unit, and an interface to the I/O and Support Processor (ISP). The RP3 processor is a proprietary design based

on the philosophy that all instructions should be completed in a single cycle. But unlike other RISC architectures, it interlocks internally in hardware. The PME provides a memory mapping function as part of address translation and allows memory to be dynamically partitioned between global and local memory.

The RP3 interconnection network is composed of two networks. One provides low latency, and the other has the ability to combine messages directed to the same memory location. The low latency network is similar to an Omega network but provides dual source-sink paths.

A.8 WARP

Warp is a MIMD one-dimensional systolic array computer being designed and built at Carnegie Mellon University. It consists of 10 identical cells in a linear array and delivers a peak performance of 100 MFLOPS.

Each cell contains two floating point processors, one ALU, and one multiplier. The processors are pipelined, and each can deliver up to 5 MFLOPS.

An operand register file is dedicated to each arithmetic processing units to ensure that data can be supplied at the rate they are consumed. Within each cell, a crossbar is used to support high intra-cell bandwidth.

Every cell contains 4K of 152-bit word micro-store and 4K of 32-bit RAM as well as other registers to provide sufficient control. Each cell can be programmed individually to execute a different computation.

Data flow through the array on two data paths while addresses and control signals travel on the address path. Each input data path has a queue to buffer input data.

As interface unit handles I/O between array and the host and performs data conversion. Addresses for data and control signals are generated but the interface unit and are propagated from cell to cell.

Warp is designed to interface with a VAX 11/780 through an interface computer which provides 1 Mbyte of memory and a 24-Mbyte/second bandwidth. The host is responsible for carrying out high-level application routines and supplying data to the WARP.

APPENDIX B
SIMULATION QUESTIONNAIRE RESULTS

SIMULATOR NAME: AI-NET
COMPANY: AI WARE Inc.
HOST MACHINES: IBM PC/AT
LANGUAGE BASED ON: "C"
ALGORITHMS IMPLEMENTED: BEP
INTERNAL/COMMERCIAL: Commercial
SPECIFIC/GENERAL: General
USER FRIENDLY:
PRICE: \$1500
NOTES: Accelerator Board for \$3000

SIMULATOR NAME: BehavHeuristics Adaptive Network Knowledge Engineering
COMPANY: BehavHerTool
HOST MACHINES: Sun, Apollo, Tektronics Workstations; MacIntosh II
LANGUAGE BASED ON:
ALGORITHMS IMPLEMENTED:
INTERNAL/COMMERCIAL: Commercial
SPECIFIC/GENERAL: General
USER FRIENDLY:
PRICE:
NOTES: To be on CM and Symbolic 36xx

SIMULATOR NAME: Netwurz
COMPANY: DAIR Computer Systems
HOST MACHINES: IBM PC/XT/AT, 2
LANGUAGE BASED ON: PL/D
ALGORITHMS IMPLEMENTED:
INTERNAL/COMMERCIAL: Commercial
SPECIFIC/GENERAL: General
USER FRIENDLY:
PRICE: \$80
NOTES:

SIMULATOR NAME: Hughes International Simulator
COMPANY: Hughes Aircraft Company
HOST MACHINES: IBM PC/AT
LANGUAGE BASED ON:
ALGORITHMS IMPLEMENTED:
INTERNAL/COMMERCIAL: Internal
SPECIFIC/GENERAL: Specific
USER FRIENDLY:
PRICE:
NOTES:

SIMULATOR NAME: ?
COMPANY: Loral Systems Group
HOST MACHINES: Loral Aspro
LANGUAGE BASED ON:
ALGORITHMS IMPLEMENTED: BEP
INTERNAL/COMMERCIAL: Internal
SPECIFIC/GENERAL: Specific
USER FRIENDLY:
PRICE:
NOTES:

SIMULATOR NAME: Autocode
COMPANY: Los Alamos National Laboratory
HOST MACHINES: Cray
LANGUAGE BASED ON: FORTRAN
ALGORITHMS IMPLEMENTED: BEP
INTERNAL/COMMERCIAL: Internal
SPECIFIC/GENERAL: Specific
USER FRIENDLY:
PRICE:
NOTES:

SIMULATOR NAME: Syspro
COMPANY: Martingale Research
HOST MACHINES: IBM PC/XT/AT, MassComp MC5xxx
LANGUAGE BASED ON: Fortran
ALGORITHMS IMPLEMENTED:
INTERNAL/COMMERCIAL: Commercial
SPECIFIC/GENERAL: General
USER FRIENDLY:
PRICE: \$1,000
NOTES: Neural Language

SIMULATOR NAME: Nestor Development System
COMPANY: Nestor, Inc.
HOST MACHINES: IBM PC/AT; Apollo and Sun Workstations
LANGUAGE BASED ON:
ALGORITHMS IMPLEMENTED: RCE
INTERNAL/COMMERCIAL: Commercial
SPECIFIC/GENERAL: Specific
USER FRIENDLY:
PRICE: \$20,000
NOTES:

SIMULATOR NAME: Plato/Aristotle
COMPANY: Neuraltech
HOST MACHINES: General Purpose Computers 286/386 to Cray
LANGUAGE BASED ON: "C"
ALGORITHMS IMPLEMENTED:
INTERNAL/COMMERCIAL: Commercial
SPECIFIC/GENERAL: General
USER FRIENDLY:
PRICE: \$2,000 (Lease)
NOTES: Neural Language

SIMULATOR NAME: Neural Works Professional
COMPANY: NeuralWare
HOST MACHINES: IBM PC/XT/AT
LANGUAGE BASED ON: "C"
ALGORITHMS IMPLEMENTED: Hopfield, BSB, Adeline, Feature Maps, BEP Perceptron
INTERNAL/COMMERCIAL: Commercial
SPECIFIC/GENERAL: General
USER FRIENDLY:
PRICE: \$495
NOTES:

SIMULATOR NAME: GINNI
COMPANY: SAIC
HOST MACHINES: Symbolics 36xx
LANGUAGE BASED ON: LISP
ALGORITHMS IMPLEMENTED:
INTERNAL/COMMERCIAL: Internal
SPECIFIC/GENERAL: General
USER FRIENDLY:
PRICE:
NOTES: Development Environment

SIMULATOR NAME: Neural Network Workstation
COMPANY: Texas Instruments
HOST MACHINES: Explorer/Odyssey
LANGUAGE BASED ON: LISP
ALGORITHMS IMPLEMENTED: Hopfield, BSB, BEP, ART II
INTERNAL/COMMERCIAL: Commercial
SPECIFIC/GENERAL: General
USER FRIENDLY:
PRICE: Free with Explorer
NOTES:

SIMULATOR NAME: Motion Detection
COMPANY: Xerox Palo Alto Research Center
HOST MACHINES: VAX; IBM PC; InterLisp-D Machines; MacIntosh
LANGUAGE BASED ON: LISP
ALGORITHMS IMPLEMENTED: Custom
INTERNAL/COMMERCIAL: Internal
SPECIFIC/GENERAL: Specific
USER FRIENDLY:
PRICE:
NOTES:

SIMULATOR NAME: PRISM
COMPANY: Xerox Palo Alto Research Center
HOST MACHINES: VAX, IBM PC; InterLisp-D Machines
LANGUAGE BASED ON: LISP
ALGORITHMS IMPLEMENTED: Custom
INTERNAL/COMMERCIAL: Internal
SPECIFIC/GENERAL: Specific
USER FRIENDLY:
PRICE:
NOTES:

Part VI

**ADVANCED IMPLEMENTATION
TECHNOLOGY**

Edited by Jay Sage

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS	iv
LIST OF TABLES	v
VI ADVANCED IMPLEMENTATION TECHNOLOGY	
1. OVERVIEW	1
1.1 Contents of Report	2
1.2 The Panel Charter	2
1.3 Connection to Other Neural Network Research	3
1.4 Implementation Using Conventional Hardware	3
1.5 Advanced Technology Neural Networks for Simulation	5
2. GENERAL AND PHILOSOPHICAL ISSUES	7
2.1 The Neural Network Definition	7
2.2 Connectivity Constraints in Implementations	9
2.3 The Analog-versus-Digital Issue	16
2.4 The Role of Learning	19
2.5 The Role of Optics	20
3. SURVEY OF IMPLEMENTATION EFFORTS	23
3.1 Taxonomy of Network Implementations	23
3.2 Summary of Implementations	26
4. CONCLUSIONS CONCERNING ADVANCED IMPLEMENTATION TECHNOLOGY	31
4.1 Adaptive Synapses	31
4.2 Massively Parallel Chip Interconnection	31
4.3 Optical Devices	32

4.4 Implementation-oriented Theory	32
APPENDIX A – DESCRIPTIONS OF IMPLEMENTATION PROJECTS	35
A.1 Electronic Implementations	35
A.2 Optical Implementations	52

LIST OF ILLUSTRATIONS

Figure No.		Page
2-1	Nearest-neighbor Connectivity	10
2-2	Intermediate Connectivity	11
2-3	Full Connectivity	12
2-4	Special-ordered Connectivity	13
2-5	Three-dimensional Integrated Circuits	14
2-6	Optical Interconnection Between Chips	15
2-7	Optical Neural Network	21
3-1	Graphical Performance Summary	27

LIST OF TABLES

Table No.		Page
3-1	Tables of Chip Characteristics (1)	28
3-2	Tables of Chip Characteristics (2)	28
3-3	Tables of Chip Characteristics (3)	29
3-4	Tables of Chip Characteristics (4)	29
3-5	Tables of Chip Characteristics (5)	30

1. OVERVIEW

Just as neural networks offer a radically different approach to information processing, so they offer radically different opportunities for advanced implementation techniques. In many comparisons between neural networks and alternative solutions to problems, the mistake is made of overlooking technology issues in the comparison. For example, neural networks are generally studied using computer simulations, and the comparisons to alternative algorithmic solutions are often made in terms of the computation times required by each approach. Such comparisons are not relevant, however, when advanced implementations can give special computational advantages to neural networks. Thus implementation technology is a critical part of the neural network research picture.

Neural networks can be implemented using conventional digital VLSI. However, the special characteristics of neural networks appear to be a natural match to other implementation approaches that may offer unique advantages. One of these approaches makes use of analog techniques that perform physical rather than symbolic computations. These techniques allow computations with massive amounts of data to be performed extremely rapidly and with very little hardware. For example, Kirchoff's Law can be used to compute the sum of hundreds of values represented by electrical currents by allowing those currents to flow onto a common wire. The entire summation is performed in a single operation using a device – a conducting line – that takes up less chip area than the power supply line of a digital processor. Moreover, synapses can store analog weights in a single device, while digital weights require one device per bit. This may further increase the number of synapses and neurons that can be implemented in a single component.

On the other hand, analog neural network circuitry does not afford the accuracy of symbolic circuitry. Adaptive learning in neural networks may provide a feedback mechanism that can, in effect, calibrate the analog circuitry and result in high effective accuracy. Relatively little research has addressed these issues so far, and it is not yet clear whether the potential advantages of analog processing techniques will be borne out in practice and whether those advantages will be great enough to outweigh the advantage offered by digital technology, which has reached a high state of development and which will continue to be strongly driven by other applications.

Two technologies appear to offer particular promise for advanced network implementations: electronics and optics. Electronics, even analog electronics, is relatively well-developed and can readily provide flexible, well-controlled nonlinear characteristics. On the other hand, electronic circuits, being confined to a plane, can support only a relatively small number of neurons and synapses in a single component ($< 10^6$). They are also constrained in the types of connectivity that can be achieved. The two extremes of fully-connected networks and nearest-neighbor-connected networks can be implemented efficiently in integrated circuits; intermediate connectivity patterns, however, pose a problem.

Optical and electro-optical devices are considerably less well developed than electronic devices but offer potentially much larger numbers of neurons and synapses per component (perhaps as high as 10^{10}). Optics can support more flexible connectivity patterns and can provide long-distance interconnects without capacitive or inductive crosstalk and loading. Optics is particularly well-suited to space-invariant connectivity, where the connection weight between any pair of neurons depends only on their relative geometric positions. Many vision and image processing network architectures require precisely this kind of connectivity.

A number of experimental efforts have been undertaken to explore advanced implementation approaches to neural networks. Few of these efforts have been well-funded; most have been pursued as a sideline or at a very low level of effort. Since a significant part of the practical advantage potentially offered by neural networks depends on their successful implementation using advanced technologies, technology development should be an essential element in any neural network research program. In addition, theoretical studies that take into account implementation issues – such as device imperfections and variations – need to be encouraged.

1.1 CONTENTS OF REPORT

This Part of the Neural Network Study's Technical Report – that concerned with *Advanced Implementation Technology* – comprises four chapters. The first sets forth the charter of this panel and indicates why implementation is an important subject within the field of neural network research. The second chapter presents a discussion of general and philosophical issues relating to advanced implementations of neural networks in technologies available at this time and in the foreseeable future. The third chapter is a summary of current implementation efforts, including devices that have been demonstrated, devices that are under development now, and some devices that have been proposed based on existing technology. The final chapter summarizes the findings of the Neural Network Study's *Advanced Implementation Technology Panel* and presents its recommendations for future research.

1.2 THE PANEL CHARTER

As the name of this Panel indicates, its charter has been to examine the application of advanced implementation technology to the fabrication of artificial neural networks. This definition has several important consequences.

First, implementation of neural networks is, by definition, an engineering issue. Many questions, although of great interest in general, are not the concern of this panel. Among these are questions of biological fidelity and even whether the neural network provides an adequate solution to any particular problem. Nevertheless, implementation researchers, like others in the neural network field, are generally motivated by and interested in biological nervous systems, and there will likely be important cross-fertilization between implementation research and biological research. Carver Mead, in particular, views his implementation research as “artificial biology” and feels that important clues to how biological systems work will arise from the effort to produce artificial neural systems.

The Panel's title – *Advanced Implementation Technology* – has been taken to exclude consideration of implementations of neural networks in either software or conventional or general-purpose integrated circuits. Those areas have been considered to fall within the domain of the *Simulation/Emulation Tools and Techniques Panel* (Panel 3). This Panel has focused its attention on devices where the implementation takes advantage in some way of the natural topology of neural networks. This does not mean that only novel device technologies or only fully parallel implementations have been considered. Conventional digital VLSI may offer the best implementation vehicle, and systolic and other partially serial architectures

can be applied effectively to neural network implementation, as some of the work surveyed in Chapter 3 and in Appendix A indicates.

The work of this Panel is predicated on the assumption that neural networks will prove to offer solutions to important problems and that one will want to implement those solutions in a form that can exploit the advantages offered by neural networks. Among these are very high throughput (realtime operation) as a result of massive parallelism; small size; and low power consumption.

1.3 CONNECTION TO OTHER NEURAL NETWORK RESEARCH

Implementation research has generally been regarded as being concerned only with end-use applications of neural networks after theoretical research has demonstrated that they offer a practical solution to a problem. This is too narrow a view.

As a result of this view, almost all research into model development has considered mathematical equations and computer simulations of perfect networks. All neurons and synapses in the network are assumed to be described by the same equations. Fluctuations from neuron to neuron and synapse to synapse (variation) and within a given neuron or synapse (noise) have been almost entirely neglected.

Both real biological networks and networks built with man-made technology exhibit variation and noise. That neural networks will not be significantly affected by these characteristics is often asserted as a matter of faith. Robustness of neural networks in the face of component variation and noise may be a critical factor for implementation technology. It is, therefore, important for theoretical research to begin to take into account these characteristics. Network models that can perform a desired function but lack this robustness may not be able to meet the requirements of a practical application.

From an implementation point of view, one of the aims of neural networks is to achieve accurate, low noise information processing with components that, individually, may be inaccurate and noisy. This, interestingly enough, is the same goal that digital processing systems achieve. Digital processors achieve the goal at the device level by (1) using transistors as bistable devices and (2) representing the data in symbolic digital form. Neural networks aim to achieve the goal at a system level as a result of adaptation and learning.

1.4 IMPLEMENTATION USING CONVENTIONAL HARDWARE

Although the Advanced Implementation Technology Panel is concerned with advanced implementation, this discussion will gain perspective if some characteristics of neural networks implemented using conventional digital integrated circuit technology, which generally offers shorter development cycles, is briefly considered.

In a digital implementation of a neural network, there are three functions which dominate the processing:

- Generating sums of weighted inputs;
- Updating the weights during adaptation; and

- Passing data (from inputs and intermediate layer outputs) to the neural elements.

In addition to these functions, the system may need to implement nonlinear transformations of the sums, determine adaptation parameters, etc.

A digital realization may be either sequential or parallel in nature, or may have a mixture of sequential and parallel elements. For example, if a standard microprocessor is used as the basis of an implementation, the program it executes represents a sequential operation. If a single processor is not sufficient to achieve a desired performance goal, a system may have several such sequential processors operating in parallel.

Sequential realizations have the advantage that costly components, such as multipliers, are time-shared among many weights. However this sharing slows down the system and may affect reliability. One of the expected advantages of neural networks is reduced sensitivity to individual component failures. However, if one building block is shared among many weights or many neurons, the failure of that building block could be catastrophic in that the system may not have sufficient reserves left to be able to correct by adaptation.

A fully parallel realization may be possible, using gate-arrays or custom chips, if the need for multiplication and high-resolution addition can be eliminated. If the neural network uses binary variables as signals between neural elements, the multiply/accumulate operations are reduced to single-bit add/subtract decisions. Hard-threshold neural elements are therefore to be preferred over those using sigmoid functions.

As one measure of the performance of a neural network implementation, consider the number of weighting operations performed per second. A general-purpose microprocessor will typically be capable of from 10^5 to 10^6 weight-operations per second. However, with the use of non-binary variables, this performance may be reduced by as much as an order of magnitude because of the time required for multiplications.

Digital signal processing microprocessors (DSPs) will offer higher performance, particularly when non-binary variables are used. These devices usually contain parallel multipliers and may allow performance in excess of 10^6 weight-operations per second, even with non-binary variables. DSP's are also more self-contained than most general-purpose microprocessors in that they usually contain program ROM or EPROM and a certain amount of data memory. When building parallel systems, this self-contained nature allows the packing of more processors into a given volume.

To achieve performance beyond that available with microprocessors, the designer may consider special-purpose hardware. If built with off-the-shelf components, such hardware may offer higher speeds, but at the cost of additional packages of logic. Because the most important operations are addition/subtraction and/or multiplication, common timing and control might be used to drive a large number of simple processors consisting primarily of memory and multiply/accumulate chips. Each such module should achieve performance beyond 10^7 weight-operations per second.

The highest performance can be expected from semi-custom or fully custom integrated circuits. A fully parallel system might be made using an array of flip-flops for each weight. Associated with each weight is also a multi-bit adder which is used during adaptation to add (or subtract) an increment to (or from) the

weight. During the summation process, multiplexers rearrange some of the adder connections to make a fully parallel summation unit.

Using an assumption that (1) each weight bit requires some 20 transistors, (2) the average full-adder bit requires some 40 transistors, and (3) multiplexing requires some 15 transistors per bit, the total becomes approximately 75 transistors per bit. Assuming 10-bit weights, a figure of about 750 transistors per weight is arrived at. Today's technology may permit several hundred thousand transistors per chip, so one could expect to realize perhaps 200 or more weights per chip. If a propagation time in the order of 20 nanoseconds could be achieved, the performance would approach 10^{10} weight-operations per second. Thus it would appear that better use of today's technology could achieve at least three orders of magnitude improvement in performance per unit volume of hardware over conventional simulation on microprocessors. This performance figure is borne out by the projections for the AT&T all-digital Hamming network described in Appendix A.1.

1.5 ADVANCED TECHNOLOGY NEURAL NETWORKS FOR SIMULATION

The ultimate purpose of advanced-technology implementations of neural networks is to serve high-performance end-use applications. However, as an intermediate step, they might also be of use as coprocessors in advanced neural network simulators.

There are some serious limitations to this application. Most advanced-technology implementations directly exploit the architecture and topology of a neural network model. Thus, they will generally be highly model-dependent and will not offer the flexibility desired in general-purpose simulation tools.

Secondly, simulations must normally allow precise specification of the parameters of the model under study and must include only the effects of features explicitly included in the model. Effects introduced by features of the hardware used to perform the simulation are not desirable. This is precisely the attraction of digital computers: with the exception of effects due to limited numerical precision, the computation follows perfectly the specifications expressed in the program. With advanced-technology neural network implementations, noise and imperfections in the device itself will contribute in an inseparable way to the result.

For example, one of the applications of simulation tools should be to study the effect on neural network performance of noise and component imperfections. In such a study, one must have precise control over the types and magnitudes of the noise and imperfections. Simulators based on actual neural network devices would experience noise and imperfections that could not be controlled.

Thus advanced simulators will in many cases want to make use of special purpose integrated circuit devices based on conventional digital technology. However, as studies advance — as surely they soon will — to the point where they deal with very large networks, digital simulation of the entire network will become nearly impossible because of the number of computations required. In such cases, advanced neural network devices will be essential for investigating the gross behaviors of particular network models in order to develop intuition about the performance of the network or to determine, for example, the ranges of parameters that result in interesting or useful behavior. Time-consuming digital simulations can then focus on just the problems of interest.

2. GENERAL AND PHILOSOPHICAL ISSUES

In the course of this Study, considerable thought has been given by all of the panels to several fundamental questions, and several of the panels have attempted to answer those questions from their own particular perspectives. The key questions are:

- What are neural networks?
- What tasks do they attempt to perform?
- What is distinctive or unique about the ways in which they attempt to perform these tasks?

This chapter looks at the answers to these questions as they apply to advanced implementation technology for neural networks. The discussion will lead to three specific important technical issues relating to implementation:

- The connectivity limitations inherent in the various approaches;
- The appropriate roles for analog, digital, and mixed circuitry; and
- The role that learning can play.

2.1 THE NEURAL NETWORK DEFINITION

The working definition of a neural network adopted for this Study was presented in *Part I, Chapter 2, Figure 2-3* of this Report (the definition is presented also in *Part I, Appendix B: A Neural Network Glossary of Terms* and discussed in detail in *Part II: Adaptive Knowledge Processing, Chapters 1 and 2*). The elements of that definition can be viewed specifically from an implementation viewpoint, allowing identification of the essential components required in a network implementation, the basic characteristics of those elements, and the features of neural networks that lead to special forms of implementation.

2.1.1 Connection to Biology

This discussion will begin with the last clause of the definition, the one relating to inspiration from biological networks. Implementors, like others involved in neural network research, are often inspired by the biological nervous system and the processing feats it can accomplish. Both network model development (addressed by the Adaptive Knowledge Processing Panel – see *Part II* of this Report) and simulations (addressed by the Simulation/Emulation Tools and Techniques Panel – see *Part V* of this Report) contribute in an important way to the study of biology as well as to the development of engineering applications. This is generally not true of advanced implementation technology, where the overwhelming interest is in

building practical information processing systems that meet requirements for realtime operation at low power and in a compact package.

There are some exceptions. For example, some of the work of Carver Mead and his students at the California Institute of Technology is attempting to elucidate the sensory preprocessing performed in biological systems by studying engineering systems modeled after them.

2.1.2 Many Simple Processors

The first clause of the definition indicates that a neural network performs its computations using a large number of simple processors. This is quite different from what is done in conventional computers, which depend on small numbers of extremely powerful, highly sophisticated, complex processing units.

In recent years, there has been a lively discussion concerning the appropriate level of complexity in the central processing units of digital computers, from microcomputers to mainframes. A new school of thought has been arguing for computers with central processing units that execute a smaller than usual set of fundamental, simple operations. These computers are identified by the acronym RISC, for "reduced instruction set computer."

A second trend has been to get around the speed limit imposed by serial operation by building computers with multiple processors, the number ranging from several (as in the Cray machines) to tens of thousands (as in the Connection Machine).

Neural networks represent the limit of thinking in this direction. The processors are so simple that as many as several hundred thousand have been implemented on a single integrated circuit. With these processors functioning totally in parallel, individual neural network circuits or optical subsystems can achieve processing rates between a billion and a trillion operations per second. Of course, these operations, because of their simplicity, do not generally accomplish what an operation on a digital computer does, and so these figures can be misleading.

2.1.3 Processing Elements and Operations

There are two types of processing elements in a neural network. By analogy with components in biological networks, these are often called neurons and synapses. A synapse performs an operation on its single input. The operation is typically a multiplicative weighting; in some cases, it may be a difference calculation or other operation. A neuron receives inputs from many synapses, combines them in some way, and then performs a generally nonlinear operation on the result.

Linear summation naturally lends itself to direct physical implementation. Currents directed into a conductor, charges transferred onto a capacitor, and optical photons directed onto a detector are all added without requiring explicit information processing. Very high accuracy is achieved with virtually no effort and in a very compact device structure.

The Neural Network Study's Adaptive Knowledge Processing Panel (Panel 2) has found that models employing more sophisticated interactions between the individual synaptic signals to a neuron (such as

temporal coincidence) should be investigated because they may lead to significantly more powerful network models. If this turns out to be the case, new thinking on the part of implementors will be required to determine what kinds of processing besides addition can be performed efficiently.

2.2 CONNECTIVITY CONSTRAINTS IN IMPLEMENTATION

Biological neural networks are constructed in a three-dimensional world from microscopic components that seem to be capable of nearly unrestricted interconnection. This is, alas, not true of man-made network implementations, either electronic or optical.

2.2.1 Connectivity Within a Chip

Single integrated circuits using present-day technology are planar objects with a limited number of layers available for interconnection. This places severe constraints on the types of neural network models that can be implemented efficiently in silicon.

Figure 2-1 shows a two-dimensional array of neurons with near-neighbor-only connections. This network architecture is common for sensor preprocessing applications, especially in vision, where the neurons are the optical detectors and the interconnections allow preprocessing operations to be performed in the focal plane. Although the direct connections may be nearest-neighbor-only, longer range interaction between the neurons can occur as a result of multiple processing cycles.

The nearest-neighbor structure makes very efficient use of silicon area. First of all, the total number of processing elements (neurons plus synapses) is a small integer multiple of the number of neurons and scales with the area of the chip. Secondly, no crossovers are required for the wires that connect the neurons to their synapses, and thus little chip real estate is wasted on crossover connections between conducting layers.

As connections are required to more distant neighbors, the number of crossover connections grows extremely rapidly. Figure 2-2 shows the situation with only second nearest neighbors added, and it is already almost too complex to draw. If non-multiplexed interconnection lines are used, most of the chip area is soon consumed in wiring and crossovers, and a relatively small fraction of the area is devoted to useful information processing. If multiplexed lines are used, control becomes far more complex and fully parallel operation is lost.

At the other extreme of full interconnection between all neurons in an input set and all neurons in an output set, an efficient use of silicon area can again be achieved. As shown in Figure 2-3, this has been achieved by changing to a different chip architecture, with the neurons arranged along the edges of the chip. Although the number of neurons now only scales as the linear dimension of the chip, the total number of processing elements is now dominated by the number of synapses, and their number scales as chip area. As with nearest-neighbor interconnection, essentially no crossovers are required. The axons (neuron outputs) and dendrites (neuron inputs) run as perpendicular bus lines in separate conductor layers in the circuit. One should note that the efficiency of the fully-connected network depends critically on

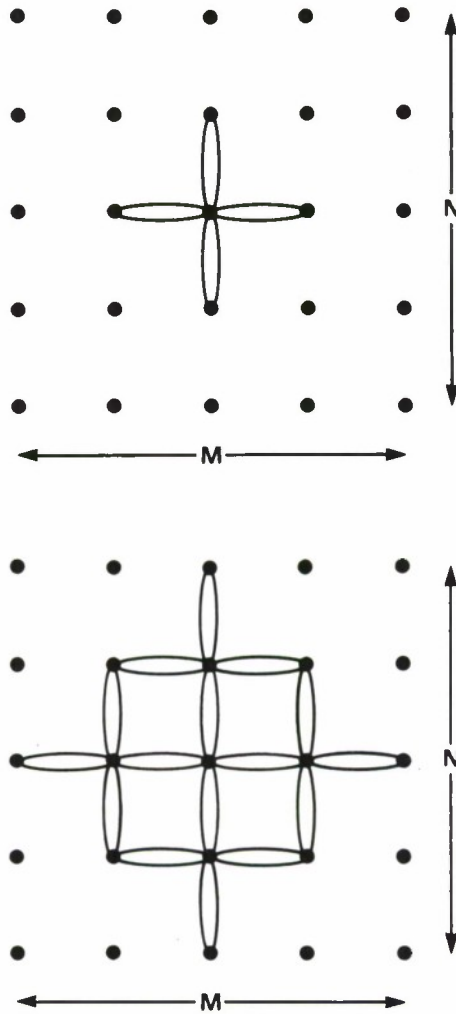


Figure 2-1. Nearest-neighbor Connectivity. This figure shows the topological constraints imposed by nearest-neighbor interconnection within a two-dimensional array of neurons. The upper part of the figure shows the connectivity for a single neuron, while the lower part of the figure shows the overlapping connectivity of a group of neurons.

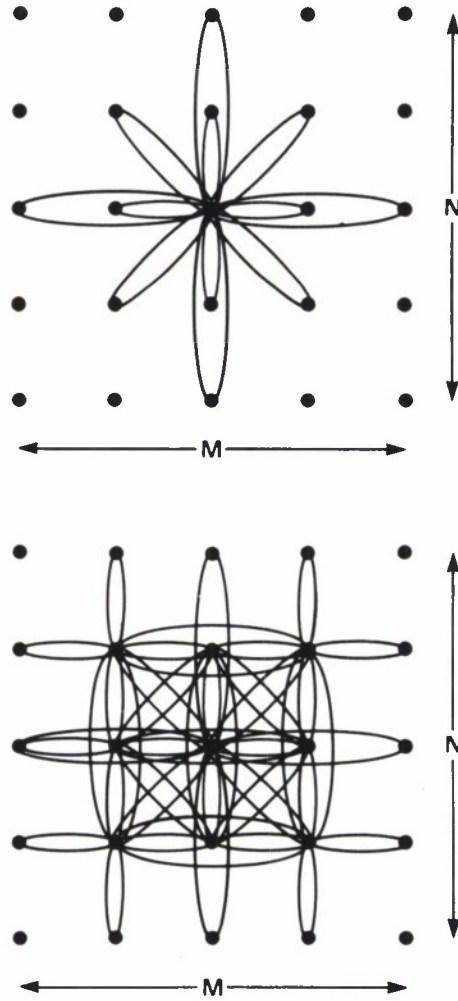


Figure 2-2. *Intermediate Connectivity.* This figure shows the topological constraints imposed when connectivity is extended to second nearest neighbors. The upper part of the figure shows the connectivity for a single neuron to illustrate the number of wires required; the lower part of the figure shows the connectivity for a group of neurons to illustrate the large number of wire crossovers required.

linear addition being used to combine the individual synaptic outputs by depositing current or charge on a bus line.

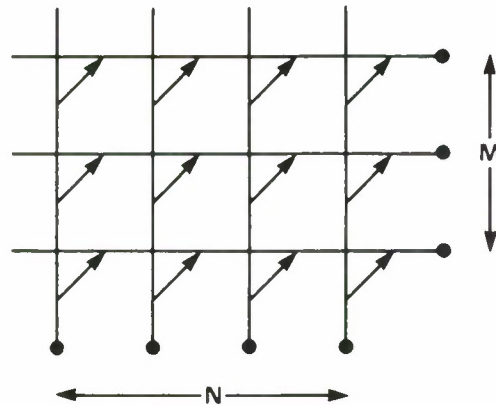


Figure 2-3. Full Connectivity. This figure shows the topological constraints imposed by full interconnection between two linear arrays of neurons. The synapses are now packed densely in the area of the chip with very little interconnect wiring required.

While intermediate levels of connectivity do not, in general, map well into integrated circuit implementation, there are some special cases where efficient designs can be achieved. One of these is the case of N input neurons, each of which connects to its M nearest neighbors. A schematic layout for this case is shown in Figure 2-4. An efficient layout is possible here because of the uniform geometrical relationship in the connectivity pattern.

2.2.2 Connectivity Between Chips

Connectivity constraints within a single network on an integrated circuit have been discussed so far. In hierarchical networks, the outputs from one network must be transferred to the input of the next network. For subnetworks with linearly-arranged neurons, it might be possible on a wafer-scale integrated circuit to connect directly from one network to the next by placing the subnetworks adjacent to one another so that the outputs from one network line up with the inputs to the next.

In other circumstances, such as when signals have to leave one chip and connect to another chip, a new set of problems arises. For very large numbers of neurons or when the neurons are distributed in two dimensions, it is impractical to bring separate leads for each neuron off a chip. First of all, it would be difficult to design a package with the required number of leads. Secondly, bringing signals off-chip through a package would require output driver amplifiers for each line, and this would consume a great deal of chip real estate and power.

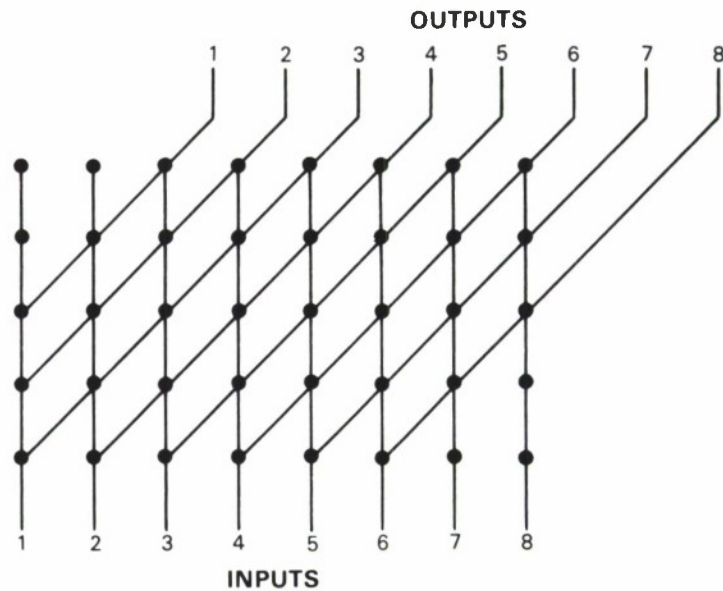


Figure 2-4. Special-ordered Connectivity. This figure shows how a network with a particular form of sparse interconnection can be implemented efficiently by taking advantage of the linear topology of the connectivity. In general, however, sparse connectivity can not be implemented efficiently in fully parallel form in a planar device.

Massively parallel chip interconnection will be required to solve this problem. Three approaches can be considered:

- Bump bonding,
- Three-dimensional integrated circuits, and
- Optical interconnection.

Bump bonding has been under development for use with infrared focal planes for coupling area-image-sensing chips to amplifier/readout chips. It could be adapted to the interconnection of neural network chips.

Three-dimensional integrated circuits are circuits with more than one layer of active semiconductor and a means for connecting the layers. Functionally, it is equivalent to a sandwich of individual integrated circuits fabricated on very thin substrates (approximately $1\ \mu\text{m}$ thick) with vias to allow electrical connections between adjacent layers. Little work on circuits of this type is under way in the United States. The Japanese, on the other hand, have been pursuing it intensively, with projects at nearly all the major laboratories in the country. The stated objective of their work has been to integrate, as illustrated in Figure 2-5, the image sensing and the digital image processing that is normally performed by a computer after the image is read out. The applicability of three-dimensional integrated circuits to hierarchical neural networks is obvious.

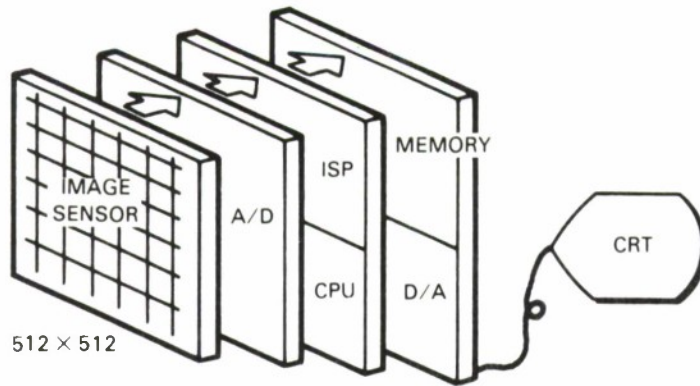


Figure 2-5. Three-dimensional Integrated Circuits. This figure illustrates an integrated circuit with more than one active silicon layer. An application to conventional computational image processing is shown in this Japanese work, but the applicability to hierarchical neural networks is obvious.

A third approach is to use optics as a passive medium to interconnect electronic chips. The arrangement shown in Figure 2-6 is meant only to illustrate some technologies that could play a role in this kind of interconnection. The lower chip incorporates the integrated deformable mirror spatial light modulator (SLM) technology under development by Texas Instruments. Light from an external source (to minimize power dissipation on the chip) is focused onto the microscopic mirrors by a precision array of microlenses, fabricated using technology developed for DARPA by Lincoln Laboratory. The reflected light beams are focused by another microlens array onto a second chip, where detectors convert the signal back into electronic form.

2.2.3 Connectivity in Optical Neural Networks

There are two generic approaches to using optics to support the connectivity of a neural network. One is to store synaptic weights in a planar transmissive (or reflective) device, such as a spatial light modulator (SLM), and to use lenses and fixed holograms to provide interconnection topology. The maximum number of independent synapses which can be provided is N^2 , where N is the linear dimension of the SLM array divided by the optical wavelength. Thus N neurons can be fully interconnected with N neurons. The shortcoming of this approach is that this connectivity is no greater than that which can already be implemented within an electronic chip. However, the optics can provide long-distance interconnects without capacitive and inductive crosstalk and loading. It can be used to supplement nearest-neighbor electronic interconnects on a chip of opto-electronic neurons. Also, many neural network problems do not require full connectivity between planes of N^2 neurons. Many feature extraction tasks performed by the early visual system, for example, rely on intermediate or even nearest-neighbor connectivity.

Lenses and planar holograms impose another constraint. The interconnect pattern they provide is space-invariant. If a planar hologram connects a neuron to three neurons, it connects another neuron in a shifted position to three neurons in correspondingly shifted positions with the same relative connection weights.

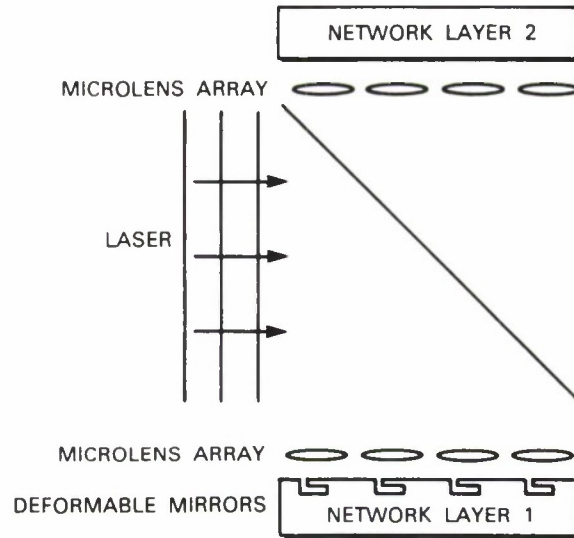


Figure 2-6. *Optical Interconnection Between Chips. Light can be used passively (non-computationally) to provide massively parallel interconnection between two integrated circuits. This figure shows an arrangement for illustrative purposes.*

For certain neural network problems, however, this type of constraint may be exactly what is required. Again, an example would be a feature extraction task of early vision, such as searching the visual field for horizontal lines.

The second generic approach is to use volume holograms stored in photorefractive crystals. This approach appears to offer connectivity between two area arrays of $N \times N = N^2$ neurons for a total connectivity of N^4 . However, the number of independent connections (i.e., gratings) that can be stored in an ideal medium is limited by diffraction to N^3 (i.e., the volume of the hologram divided by the cube of the wavelength). In actuality, each of these N^3 gratings provides roughly N connections of identical weight lying on the surface of a cone (the so-called Bragg cone degeneracy). This gives the appearance of N^4 connections, but only one connection from each grating is useful. Nevertheless, this approach does in principle provide richer connectivity: $N^{3/2}$ neurons can be fully interconnected with $N^{3/2}$ neurons. Furthermore, a volume hologram provides for space-variant interconnects. Because of the Bragg conditions, each angle of incidence can be made to address a separate pattern of connections.

The above estimate of N^3 connections assumes gratings that span the entire range of solid angles. However, a practical optical system has a limited aperture, and the fraction of the N^3 gratings that can actually be used is proportional to the square of the aperture (one factor for input and another for output). This penalty is in the 10^2 to 10^3 range.

Furthermore, the photorefractive effect does not support idealized superposition of sequentially written gratings. Each stored grating is a spatial pattern of electric fields and ionized impurity concentrations which affects the exposure process associated with the writing of subsequent gratings. Conversely, the writing of each new grating fades the ones previously stored. The limits of photorefractive materials in this

respect have not been demonstrated. Indeed, it may be easy to write a hologram using a single exposure which associates one 200×200 -element image with another and to claim that this represents over 10^9 connections. In the context of neural networks, where learning and adaptation require multiple sequential exposures, such claims obscure the real issues.

2.3 THE ANALOG-VERSUS-DIGITAL ISSUE

In all areas of information processing, there have been debates as to whether analog processing or digital processing is the more advantageous approach. Traditionally, the digital approach has offered greater flexibility and arbitrarily high accuracy. The analog approach has offered speed, size, and power advantages in specialized applications.

2.3.1 Advantages of Digital Processing

In the case of data processing and general computation, the digital approach has nearly completely supplanted earlier analog approaches. Several key factors have led to its acceptance in this domain. Perhaps the most important has been the speed, ease, and reliability with which digital circuits can be designed by inexperienced engineers with only average ability. The tremendous effort put into the development of digital design tools is only one factor that has made this possible. Even if comparable efforts were devoted to the development of analog design tools, analog design would still probably require a higher level of skill and experience, partly because the circuit interfaces are much less rigidly defined.

A second factor that contributes to the popularity of digital circuits is the ease with which highly complex systems can be constructed from a relatively small set of moderately complex standard building blocks. This leads to high production volumes and decreasing cost. Digital circuits gained even greater popularity with the advent of programmable circuits, since they can perform an extremely wide variety of tasks under the control of software, which is far easier to redesign and reconfigure than is hardware.

Still another factor is the accuracy with which computations can be performed. Perfect arithmetic accuracy is, after all, essential in financial calculations. In other applications, the perfect predictability of the results is an advantage. Analog circuits always have both random, time-dependent noise and fixed device deviations, and both of these contribute behavior which is essentially unknown.

As digital circuits found wider and wider application, more and more resources were brought to bear on the development of the technology. As processing speed increased, tasks that only analog circuitry could previously handle, such as video signal processing, came within the realm of capability of digital processing.

2.3.2 Advantages of Analog Processing

Analog processing derives its main advantage when physical processes can be used to perform required computational functions. When this is done, extremely high-speed operation can be achieved with very simple devices. For example, natural propagation of electromagnetic or acoustic signals can be used to

provide time delays, and conservation of charge can be used to sum large numbers of currents or charge packets virtually instantaneously.

As a result of these advantages, analog circuitry still reigns at the high end of the bandwidth and processing-throughput spectrum. Single charge-coupled device signal processing ICs can perform programmable correlation operations on hundreds of signal samples at a sampling rate of 40 MHz; surface acoustic wave (SAW) devices carry out pulse compression on analog waveforms with bandwidths of hundreds of MHz; superconducting delay line circuits do the same for signals with bandwidths in the GHz range. Some of these tasks can be performed by digital circuitry, but the circuitry is far more complex, takes up much more space, and requires much more power; some of these tasks cannot be performed by digital circuits at all.

Analog circuitry holds undisputed claim to one other area: the interfaces between digital systems and the real, analog world. Fast, very high-accuracy analog-to-digital and digital-to-analog converters are often the critical items that make the application of intermediate digital processing possible (for example, digital audio). Preprocessing and postprocessing of the real-world analog signals requires precision amplifiers and filters and waveshaping circuits.

2.3.3 Analog versus Digital Neural Networks

The same analog-versus-digital question has naturally arisen with neural networks. There was little likelihood that this question could be settled during the course of this Study; the answer will surely have to emerge from the results of research and the best efforts of both schools of thought. Here the Advanced Implementation Technology Panel (1) reports the opinions of the experts it interviewed and (2) summarizes some of the issues that affect the tradeoff between digital and analog implementations of neural networks.

The general, though not unanimous, feeling in the neural network research community is that neural networks and analog circuitry (most likely in concert with digital circuitry) are naturally suited to each other. This is a feeling that is held even by a number of individuals, such as Carver Mead and Federico Faggin, with notable reputations in the area of digital design. The following discussion raises some of the issues involved in the analog-versus-digital question. In practice, more and more network implementations are using combinations of analog and digital processing to exploit the advantages of each.

Encoding Efficiency

One consideration is the amount of hardware needed to store a single synaptic weight. With analog operation, a weight can be stored in a single device or group of devices; with digital operation, a device or group of devices is required for each bit in the digital representation.

On the other hand, the size of the device required to store an analog value may be larger than that of the devices required to store binary values. This is particularly true when absolute accuracy is required. In general, doubling the accuracy of an analog device requires doubling the size of the device, while the digital representation needs to add only one more bit. Thus the advantage that analog storage offers in number of devices can rapidly be lost as accuracy requirements increase.

Mitigating this effect is the possibility that neural networks will not need devices with high absolute accuracy. If networks are trained individually (as opposed to having a pre-computed set of weights downloaded into them), then one is not so much concerned with accuracy in the devices as with the precision (the fineness with which levels can be set). A device might only have an accuracy of 20% but be adjustable in increments of 0.1%. As design rules shrink, however, one will reach the point where the noise level becomes too high to support fine settability. At this point, analog devices may not be able to take advantage of scaling to smaller devices. On the other hand, the averaging process that takes place over the many synapses that contribute to the activity of a neuron may make the system less sensitive to random noise. These questions are still open issues for research.

Processing Efficiency

As noted earlier, analog processing that makes use of a physical phenomena to perform computations can offer tremendous advantages in processing speed and hardware complexity.

Conservation of charge and Kirchoff's Law were cited earlier as very simple and fast ways to carry out the addition of a large number of signals in the form of current or charge. Other examples are:

- Addition of signals in the form of light beams,
- Natural relaxation of signals on resistor-capacitor networks or by carrier diffusion, and
- Spreading of optical beams by imaging or diffraction.

Another very significant area where physical phenomena can accomplish operations that are very time-consuming to perform symbolically is stochastic processing, as in Boltzmann machines. Here the natural noise in an optical or electronic device is amplified and used for the computation.

Processing Speed

Analog processing achieves a speed advantage only as a result of either more effective processing (as noted in the previous paragraph) or more parallel processing. At the level of a single operation, digital processing is faster because one has to wait only long enough to be sure that signals have crossed the logic threshold, whereas in analog processing one must wait several time constants so that the signal can settle to an accurate analog level.

2.3.4 Behavior at the Limits of Integration

Much of the progress in integrated circuit technology has been made by making circuit features smaller. In general, reducing line width by a factor of two allows packing four times as many circuits in a given area, and the resultant circuits each operate twice as fast as their larger counterpart. Power dissipation per unit area is unchanged, yet computational power has increased by a factor of eight. In addition to

performance improvements, scaling down size has cost benefits as well. Generally, if more functions can be printed into a given area of silicon, the cost per unit function is reduced.

Of interest is whether there is a limit to the process of making features smaller. Photolithographic techniques do impose some practical limits, based on the wavelength of the light used. Methods now in development, such as electron beam and x-ray lithography, potentially offer much finer features than available with the optical methods used today.

Assuming that very fine features can be practically printed, there are circuit limitations that must be considered. The scaling results cited above assume that dielectric thicknesses and doping depths are scaled along with doping density. Thus significantly different processing techniques may be needed to make smaller transistors.

A fundamental limit on circuit size would appear to be associated with signal-to-noise ratios. As circuit size is reduced, it becomes necessary to reduce supply (and therefore signal) voltages to maintain limits on electric fields. At the same time, thermal noise effects become more important.

For digital circuits to work reliably, the signal-to-noise ratio (SNR) must be in the order of at least 20 dB. Below this level, the probability that a circuit will give erroneous output due to noise grows rapidly. Consider devices operating at a five-volt power supply and fabricated with a process technology offering a minimum feature size of $1\ \mu\text{m}$ and a gate oxide in the order of $400\ \text{Å}$. Thermal noise in such a device leads to a signal-to-noise ratio of about 70 dB. The SNR depends on the third power of the scaling parameter, and thus the maximum possible reduction in dimensions would be approximately a factor of 50 ($50^3 \approx 100,000 = 50\ \text{dB}$). Under these assumptions, the ultimate circuit would operate with a 0.1 volt power supply and experience 0.01 volts of noise.

If one were to attempt to use devices at this scale for analog circuits, no more than 10 statistically distinct analog levels could be expected from the device. This small number of levels might not permit adaptation because misadjustment during the adaptation process might be too great. Answering this question would require a careful theoretical analysis of the collective adaptive behavior of a large number of noisy devices. In general, one might expect that when large numbers of noisy signals are combined, the processing gain would counteract the effect of the noise in individual synapses. An error in one bit of a digital representation can lead to a very large error in the value, and processing gain would probably be much less effective in counteracting the error. This is still an open question.

2.4 THE ROLE OF LEARNING

With respect to learning, neural networks fall into a number of classes. Some networks, such as those that perform sensory preprocessing functions, are completely fixed. The design of the network determines the synaptic coupling coefficients. Examples are the retina and cochlea chips of Carver Mead's group at CalTech.

Other networks have synaptic weights that depend on the problem that is being addressed by the network, but the weights are pre-programmed into the network. In electronic networks, this pre-programming

may be accomplished by one of the photomasks used during fabrication or by one-time programming using laser, electron-beam, or fuse-link programming. Optical networks may be pre-programmed using photographic plates or holograms. These networks can only contain one set of weights.

Other networks can be programmed more than once. Most networks of this type can be programmed continuously from an outside source of data. Many hybrid analog/digital networks fall into this class. Synaptic weights are maintained in a digital computer and loaded onto the chip or into a spatial light modulator as needed to update the configuration of the network. Networks of this type can perform adaptive learning in response to changing environments.

The final class of networks is capable of *in situ* learning. The learning algorithm is incorporated directly into the network, and no external computer is needed to perform the learning function.

The members of the Advanced Implementation Technology Panel were in general agreement that the greatest potential of neural networks lies in learning and adaptivity. From an application point of view, this will allow networks to adapt based on experience and thus serve a broad range of applications.

A second advantage afforded by adaptivity is the possibility to compensate for device deficiencies. In applications that involve or can be made to involve training of individual network components, the training process may act as feedback to compensate for variations in the characteristics of individual devices in the neural network. This would allow much smaller and less accurate devices to be used and would result in larger network implementations.

2.5 THE ROLE OF OPTICS

The principal advantage of using optics for the implementation of neural networks is the fact that one can optically implement a three-dimensional system with relative ease. The most common method for synthesizing a three-dimensional optical neural network is depicted in Figure 2-7. The active devices, or neurons, are arranged in planes. Each neuron communicates to other neurons optically via the third dimension. Each neuron receives its inputs in the form of optical signals incident at each location, and the state of each neuron modulates a light beam that is either locally-generated or illuminates the site of each neuron.

One way to utilize the optical communication capability is in a hybrid implementation. In this case, each neural plane is essentially an electronic neural network chip which performs all the necessary nonlinear mappings and provides all the connectivity that can be accommodated on the chip with wires. Optics then provides vertical out-coupling of the information from the chip and allows chip-to-chip connections to avoid bottlenecks that would be created by pins on electronic packages. The optical connections could additionally be used to supplement the connectivity within a single chip. This approach is very attractive because it offers the best of both worlds: the flexibility of electronic design and the connectivity of optics.

The most promising technology for implementation of such opto-electronic neural networks is based on gallium arsenide (GaAs) and other III-V semiconductors, alloys, and superlattices. With the advent of molecular beam epitaxy and organometallic vapor phase epitaxy, a great variety of electronic and electro-optical devices can be fabricated on a monolithic substrate. These devices include high-speed analog and digital circuits, fast spatial light modulators, detectors, LEDs, lasers, and waveguide devices. No other

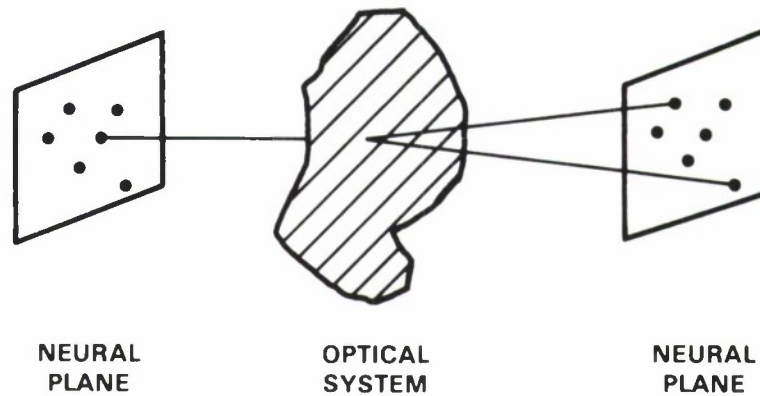


Figure 2-7. Optical Neural Network. The basic architecture of an optical neural network, showing how the third spatial dimension is utilized to advantage to achieve a very high density of neurons and synapses.

material system has this degree of versatility. While yield is still a thorny issue when fabricating large arrays, this technology is evolving rapidly because of the variety of applications that it supports.

Finally, there is a variety of hybrid devices that could be used, such as gallium arsenide on silicon or PLZT electro-optic modulators on silicon.

The arrangement shown in Figure 2-7 also allows the implementation of what might be called a fully-optical network, one in which all connections are made optically. Each neural plane can be fully populated (that is, to density $N^{3/2}$) with active devices (neurons), since there is no need to devote any of the real estate to the interconnections within the neural planes. Thus, the density of neurons per unit area on the active device, or "chip," that can be accommodated with this approach is higher. The light that originates from each position in a neural plane is redistributed to other neurons in the same or subsequent planes by holograms or other optical transparencies that are placed in the space between the neural planes. The information that is needed to specify the connections between the neurons can be recorded on either planar or volume optical storage media.

3. SURVEY OF IMPLEMENTATION EFFORTS

3.1 TAXONOMY OF NETWORK IMPLEMENTATIONS

Before describing current research efforts in network implementation, a set of classifications that will help put these implementation approaches into perspective is presented. These classifications can be applied either to the intrinsic characteristics of a technology or only to one particular application of that technology.

3.1.1 Synaptic Weight Representation

The synaptic (or interconnect) weights in a neural network can be classified by the range of values they can take on into the following classes:

- Binary (only two possible values),
- Discrete (a larger number of fixed values), or
- Analog (a continuous range of values).

The synapse circuits process input signals from external or internal sources, depending on stored “weight” values. The most common operation is multiplication, but differencing and other operations are possible. There are several possibilities for the range of weight values that can be represented. They may have binary values representing either of two possible paired memory states: (a) 0 (neutral) and 1 (excitatory) or (b) -1 (inhibitory) and +1 (excitatory). Another possibility is that the weights take on three values: -1 (inhibitory), 0 (neutral), and +1 (excitatory). Other multiple, discrete values might be supported, or the weights might take on continuous analog values over some range. The synaptic weight representation may be binary even when the technology used in the synapse is fully analog, as in the case of resistors that are either present or absent rather than having adjustable values.

3.1.2 Synaptic Programmability

The programmability of the synaptic weights can be classified as follows:

- Fixed,
- One-time programmable,
- Reprogrammable, or
- Self-adaptive.

The synaptic weights may be fixed or programmable. If they are programmable, there are several degrees of programmability. They may be programmable only once, like a PROM, either at the mask level or after fabrication; or they may be reprogrammable, like an EPROM or EEPROM. If they are reprogrammable, the programming may be externally controlled or may be internally controlled, as in biological systems, in response to the network state (self-adaptive).

3.1.3 Neural State Representation

Like the synapses, the neuron (processing element) may have output states that fall into the following classes:

- Binary,
- Discrete, or
- Continuous.

Neural states also can be represented in various forms. They might take on binary values (0/1 or -1/+1), discrete values with more than two states (ternary -1/0/+1 or multi-bit digital), or continuous analog values over some range. Two aspects of the circuit affect this representation. One is the kinds of input states that the synapses support. Some implementations, for example, use gated (on/off) synapses, which is appropriate for binary neural states. Other implementations use multiplicative inputs and can handle either multi-bit digital inputs or continuous analog inputs. The second circuit aspect that affects the representation is the kind of nonlinear detection that is performed after the weighted summation. If a bistable (comparator) circuit is used, only binary neural states can be represented.

3.1.4 Temporal Characteristics of Inputs

The timing of inputs to the network can fall into the following two classes:

- Sampled (inputs sampled at discrete times), or
- Continuous (network responds continuously to inputs).

External inputs to the network can be applied either continuously or in time-sampled fashion. Almost all implementations studied to date use discrete-time inputs.

3.1.5 Temporal Evolution of the Network

Regardless of the way external inputs are applied to the network, the neurons may respond in time in different ways. The following classes can be identified:

- Synchronous,

- Discrete,
- Asynchronous, or
- Continuous.

One major temporal characteristic of a network is whether it is synchronous or asynchronous. In a synchronous network, all neurons change state at the same discrete times in response to a global system clock. In asynchronous networks, each neuron responds to its inputs at its own pace. Asynchronous networks may be either continuous or discrete – that is, each neuron may either respond continuously in time or it may respond at discrete times on a self-clocked (non-global) basis.

3.1.6 Imbedding of Network Model

The neural network implementation can perform the network processing function in one of the two following ways:

- Direct, or
- Algorithmic.

An integrated-circuit-based neural network may implement the equations of the mathematical model of a network, or it may implement an algorithmic reduction of that model. For example, the Carpenter-Grossberg adaptive resonance theory (ART) model for an associative memory classifier involves a complex set of coupled differential equations for the neural states and synaptic weights. It would be very difficult to implement this model directly with an electronic circuit. However, there are approximate separations of the equations that allow the model to be reduced to an algorithmic form that can be implemented in the form of a parallel network. On the other hand, Carver Mead's artificial retina is a direct implementation of the first three layers of neural cells in a biological retina.

3.1.7 Technology Employed

Neural networks can be implemented in various technologies that can be classified as follows:

- Standard or special; or
- Electronic, optical, opto-electronic, or other.

The technology used to implement a neural network may be one of the standard IC technologies, such as digital or analog CMOS, or it may be a technology with some non-standard components. Examples of the latter are the Bell Laboratories resistor network, which requires special processing for the submicron resistors, and the Lincoln Laboratory MNOS/CCD-based network, which requires non-standard tunnel oxides for the analog MNOS storage elements.

Most of the electronic implementations are based entirely on electronic circuits. It is possible, however, to use a hybrid approach in which optical input (or possibly optical output) is used.

3.1.8 Breadth of Applicability

The breadth of applicability of a neural network implementation approach may be described as follows:

- Limited, or
- General.

A final category is one that is hard to judge – but worth thinking about when evaluating an implementation approach: whether the technology is limited or specifically-suited to only particular types of network models – such as Hopfield models – or whether it is capable of supporting a wide range of network models.

3.2 SUMMARY OF IMPLEMENTATIONS

Toward the end of the Study, researchers who have designed or developed advanced implementations of neural networks were invited to submit brief descriptions of their work. Many responded to the invitation; a few did not. Those descriptions that were received are included in full in Appendix A. In this section of the Report, those implementation efforts are summarized. The Panel notes, as an aside, that few of them have been well-funded; most have been pursued on an experimental basis at a very low level of support using internal discretionary funds.

The data in Tables 3-1 to 3-5 at the end of this section summarize some the characteristics of these implementations. Figure 3-1 maps the performance domain in terms of *total interconnects* (synapses) *implemented* and maps the processing rate in *interconnects-per-second*. On the speed axis, both single electronic chips and single optical subsystems have capabilities in the same range of 10^9 to 10^{12} interconnects per second. With advances in technology, both can do better — electronics slightly and optics perhaps substantially.

On the other axis – total interconnects implemented or supported – the domains of optics and electronics are quite different. Dynamic RAMs, which are just reaching the 16 Mbit level, set an upper limit to the weight storage density. Since synapses are slightly-to-considerably more complex than a RAM cell, numbers in the 10^5 to 10^6 range seem reasonable for the upper limit to the number of interconnect weights that can be stored on a single chip.

Optics, on the other hand, can make use of volume rather than area storage. A volume hologram 1 cm on a side contains more than 10^{12} cubes with a lateral dimension of λ , the wavelength of light. This sets a theoretical storage density limit that is not likely to be even approached, but a value of 10^{10} is not out of the question.

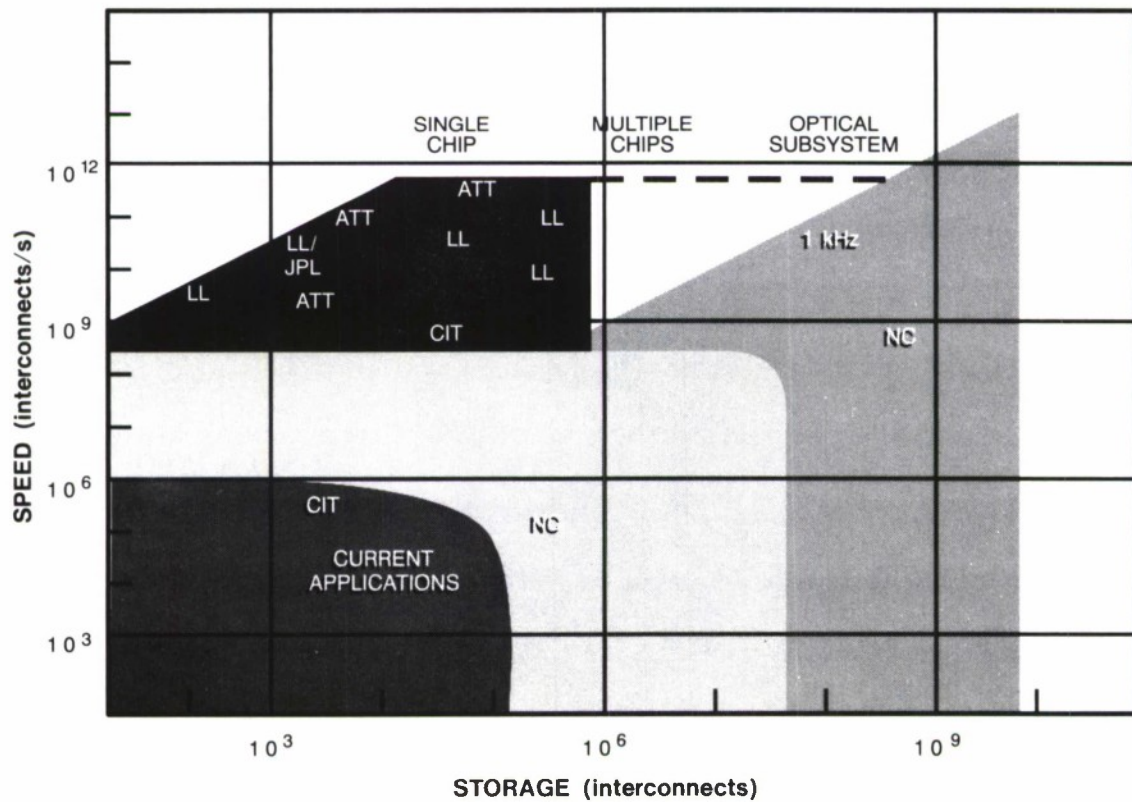


Figure 3-1. Graphical Performance Summary. This graph shows the performance domains demonstrated or proposed for single integrated circuit chips and single optical subsystems in terms of the total number of interconnection weights stored and the number of connections performed per second.

LABORATORY MODEL STATUS	ATT HAMMING BUILT	ATT FF BUILT	ATT HAMMING IN FAB
CONNECTIVITY PROGRAMMABILITY LEARNING PARALLELISM	FULL ONE TIME NONE FULL	FULL ELECTRICAL OFF CHIP FULL	FULL ELECTRICAL OFF CHIP PIPELINE
CIRCUIT TECHNOLOGY	RESISTOR SPECIAL	HYBRID A/D CMOS	DIGITAL CMOS
INPUT TIMING NEURON TIMING SYNAPSE STATES NEURON STATES	SAMPLED CONTINUOUS BINARY ANALOG	SAMPLED CONTINUOUS BINARY ANALOG	SAMPLED SAMPLED BINARY DIGITAL
SIZE (NEURONS) SYNAPSES SPEED INTERCONNECTS INT/SEC	256/256 66 K 1 MHz 66 K 66 G	46/96 4.4 K 10 MHz 4.4 K 44 G	128/50 50 100 MHz 6.4 K 5 G

Table 3-1.

Characteristics of Chip Implementations of Neural Networks

LABORATORY MODEL STATUS	CALTECH RETINA BUILT	CALTECH COCHLEA BUILT	CALTECH FF PROPOSED	JPL FF BUILT	JPL FF BUILT
CONNECTIVITY PROGRAMMABILITY LEARNING PARALLELISM	NN FIXED NONE FULL	NN FIXED NONE FULL	FULL ELEC/OPTIC OFF CHIP SEMI	FULL ELECTRICAL ONE TIME FULL	FULL ELECTRICAL OFF/ON CHIP FULL
CIRCUIT TECHNOLOGY	SUBTHRESH CMOS(MOSIS)	SUBTHRESH CMOS(MOSIS)	CCD CCD	RESISTOR CERMET/A-SI	RESISTOR EL-CHEM
INPUT TIMING NEURON TIMING SYNAPSE STATES NEURON STATES	CONT CONT ANALOG CONTG	CONT CONT ANALOG CONT	SAMPLED SAMPLED ANALOG BINARY	SAMPLED CONT BINARY ANALOG	SAMPLED CONT ANALOG ANALOG
SIZE (NEURONS) SYNAPSES SPEED INTERCONNECTS INT/SEC	48 × 48 2300 100 Hz 2300 230 K	10 kHz	256/256 66 K 1-10 MHz 66 K 1 G	40/40 1600 10 MHz 1600 16 G	16 10 MHz

Table 3-2.

Characteristics of Chip Implementations of Neural Networks

LABORATORY MODEL STATUS	MIT/LL GAUSSIAN BUILT (PROP)	MIT/LL HOPFIELD BUILT	MIT/LL HAMMING BUILT	MIT/LL KOHONEN BUILT	MIT/LL FF PROPOSED
CONNECTIVITY	NN	FULL	FULL	FULL	FULL
PROGRAMMABILITY	FIXED	ELECTRICAL	ELECTRICAL	ELECTRICAL	ELECTRICAL
LEARNING	NONE	ON CHIP	OFF CHIP	ON CHIP	OFF CHIP
PARALLELISM	SEMI (FULL)	FULL	FULL	SEMI	SEMI
CIRCUIT TECHNOLOGY	CCD CCD	CCD MNOS/CCD	HYBRID A/D WAFER	HYBRID A/D CMOS(MOSIS)	CCD CCD
INPUT TIMING	SAMPLED	SAMPLED	SAMPLED	SAMPLED	SAMPLED
NEURON TIMING	SAMPLED	SAMPLED	SAMPLED	SAMPLED	SAMPLED
SYNAPSE STATES		ANALOG	DIGITAL	DISCRETE	DISCRETE
NEURON STATES	ANALOG	BINARY	ANALOG	ANALOG	ANALOG
SIZE (NEURONS)	288 × 384	13/13	32/16	40/40	1000/20
SYNAPSES	110 K	169	128	1600	1000
SPEED	1-10 MHz	(10 MHz)	10 MHz	10 MHz	10 MHz
INTERCONNECTS	450 K	169	128	1600	20 K
INT/SEC	4 G (110 G) (I/O = 0.4 G)	1.7 G (160 G)	1.3 G	16 G	10 G

Table 3-3.

Characteristics of Chip Implementations of Neural Networks

LABORATORY MODEL STATUS	CHINA LAKE BACK PROP IN DESIGN	EDINBURGH FF BUILT	NRL FF IN FAB
CONNECTIVITY	FULL	FULL	FULL
PROGRAMMABILITY	ELECTRICAL	ELECTRICAL	ELECTRICAL
LEARNING	ON CHIP	OFF CHIP	OFF CHIP
PARALLELISM	FULL	(FULL)	FULL
CIRCUIT TECHNOLOGY	ANALOG FLOATING GATE	DIGITAL CMOS	ANALOG CMOS
INPUT TIMING	SAMPLED	SAMPLED	SAMPLED
NEURON TIMING	SYNCHRONOUS	SYNCHRONOUS	CONTINUOUS
SYNAPSE STATES	ANALOG	DIGITAL	ANALOG
NEURON STATES	ANALOG	PULSE	ANALOG
SIZE (NEURONS)			
SYNAPSES			
SPEED			
INTERCONNECTS			
INT/SEC			

Table 3-4.

Characteristics of Chip Implementations of Neural Networks

LABORATORY MODEL STATUS	ROCKWELL RETINA IN DESIGN	UCLA RETINA IN DESIGN	UCLA BACK PROP IN DESIGN
CONNECTIVITY	2ND NN	NN	FULL
PROGRAMMABILITY	NONE	NONE	ELECTRICAL
LEARNING	NONE	NONE	ON CHIP
PARALLELISM	FULL	FULL	FULL
CIRCUIT TECHNOLOGY	ANALOG CMOS	ANALOG CMOS	COOLED CAP CMOS
INPUT TIMING	SAMPLED	SAMPLED	SAMPLED
NEURON TIMING	CONTINUOUS	CONTINUOUS	SAMPLED
SYNAPSE STATES	ANALOG	ANALOG	ANALOG
NEURON STATES	ANALOG	ANALOG	ANALOG
SIZE (NEURONS)	32 × 32	8 × 8	48/10
SYNAPSES	12 K		480
SPEED	100 kHz		
INTERCONNECTS INT/SEC			

Table 3-5.

Characteristics of Chip Implementations of Neural Networks

4. CONCLUSIONS CONCERNING ADVANCED IMPLEMENTATION TECHNOLOGY

Digital electronic circuitry and precision analog electronic circuitry (such as that used for precision amplifiers and D/A and A/D converters) has been developed intensively over the past two decades. At the same time, there has been relatively little work on low-accuracy, high-density analog circuitry using very small electronic and optical devices.

4.1 ADAPTIVE SYNAPSES

Most of the implementation work to date on programmable networks has concentrated on those designed to be programmed by external sources. Little work has been done on networks that can perform their learning operations as an intrinsic function of the network. This capability will be increasingly important in two circumstances:

- When large amounts of training data must be processed at high speed; and
- When the circuit must adapt to its own imperfections.

4.2 MASSIVELY PARALLEL CHIP INTERCONNECTION

Quite a few experimental neural networks have been developed. Despite their small size in many cases, they achieve connection rates approaching 10^{11} interconnects per second. At this processing rate, input/output of data to and from the chip or optical system is often the limiting factor. Since it is widely recognized that few problems will be solved with a single neural network and that hierarchies of interconnected networks will be needed, massively parallel interconnections between networks will be required to take advantage of the processing power of the individual network subsystems.

Indium bump bonding is one approach to this problem. It has been under development for several years now, principally for connecting non-silicon infrared detector arrays to silicon integrated circuit preamplifiers and signal processors.

Another technology that would be highly advantageous to neural networks would be so-called three-dimensional integrated circuits. These are circuits containing more than one active layer of silicon. Complex integrated circuitry can be fabricated in each layer, and the layers can be densely interconnected. This technology is being pursued vigorously in Japan but is receiving relatively little attention in the United States. Figure 2-5 in section 2.2.2 illustrates such a circuit.

A third possibility is to use optics in a non-computing mode to provide dense interconnection between chips. Figure 2-6 in section 2.2.2 illustrates the concept. Massively parallel interchip connections have not been demonstrated using optics.

4.3 OPTICAL DEVICES

Although optical neural networks offer great promise for applications requiring a very large number of neurons and systems with complex sparse connections, progress and performance in optics are hampered by poorly-developed devices and materials.

Though the III-V semiconductors support the most mature and versatile technology, research is still needed to produce large uniform arrays of devices with reasonable yields. While this research is relatively expensive (involving such high-cost technologies as MBE), the payoffs are also high.

Research on volume holograms in photorefractive materials should focus attention on the characteristics of the materials themselves. One issue is the storage capacity and readout characteristics when many patterns are sequentially stored (even the Kukhtarev model predicts non-idealities that are inherent to the photorefractive effect). A second issue is that in some applications, a fixing mechanism is needed in order to stabilize the stored patterns, and this is an area of active research. A third issue relates to materials preparation, that is, the problem of how to grow photorefractive crystals with optimal characteristics. *Fe*-doped *LiNbO₃* is the most well-developed and well-characterized material in this respect.

In general, the realization of the benefits of optical and opto-electronic neural network implementations will require long-term funding of interdisciplinary research involving chemists, materials scientists, physicists, computer scientists, mathematicians, and engineers. Current funding realities in optical computing tend to reward short-term work on conceptual architectures based on idealized materials and devices. This sort of work pays too little heed to the impact of the non-ideal behavior of real materials and devices. Such non-idealities must be incorporated into the “error analysis” used to evaluate the performance of competing neural architectures and algorithms. Indeed, this type of analysis is intrinsic to the design of optical and opto-electronic neural architectures; it is not merely an engineering detail to be postponed until a separate “implementation phase.”

4.4 IMPLEMENTATION-ORIENTED THEORY

As noted elsewhere in this Report, almost all research on neural network models has concentrated on the performance of networks that exactly obey the equations defined for that model. In real networks there will be variations from component to component. Not all neurons will have the same nonlinear transfer relationship; gains and thresholds will vary. Not all synapses will perform their weighting operation or respond to learning signals in exactly the same way. Weighting functions will have nonlinearities, and the programming response gain will vary from synapse to synapse.

It is very important that neural network theory begin to address these factors. Robustness in the face of defects and imperfections is almost universally cited as a characteristic of neural networks, but almost no effort has been devoted to substantiating the claim. It will not be surprising if some models are found to offer little robustness, while others perform well in the face of certain types of imperfections. Unfortunately, as the Neural Network Study’s Simulation/Emulation Tools and Techniques Panel has discovered, very few neural network simulation tools have the capability to simulate these aspects of networks.

If a network is to take advantage of the extremely high performance capabilities of dense network integrated circuits and optical systems, such robustness will be essential. In addition to trying to find models that account for the behavior and performance of biological systems, network theorists should also try to find models that will perform well even when implemented with noisy, low-accuracy components.

APPENDIX A

DESCRIPTIONS OF IMPLEMENTATION PROJECTS

This appendix contains short descriptions of a number of neural network implementation projects. Toward the end of the Study, the Advanced Implementation Technology Panel sent letters to the neural network implementors whose work had come to its attention, inviting them to contribute one-page descriptions of each distinct implementation they had developed. Many responded to this request; a few did not. Those contributions that were received, with only slight editing, make up this Appendix.

A.1 ELECTRONIC IMPLEMENTATIONS

A.1.1 AT&T High Density Interconnection Matrices

A fabrication technology for high-density resistor matrices has been developed. The goal is to make resistors as small as possible with values high enough that a large number of neurons can be interconnected. Also, the technology must be VLSI-compatible in order to combine these matrices with macro-electronic circuits.

The matrices built consist of two sets of electrodes at right angles to each other, separated by a dielectric layer. At some of the crosspoints, resistive connections are stacked vertically between the wires. In this way, a resistor occupies only the area of the crosspoint of two wires and requires no extra on the chip.

The fabrication process for such a matrix proceeds as follows: First, tungsten wires forming the base electrodes are patterned. These wires are coated with a layer of polyamide (dielectric) and, at the places where a resistor will connect two crossing wires, holes are etched through the polyamide. Then, a layer of amorphous silicon is deposited and, on top of that, a layer of tungsten. As a final step, the top electrodes, which run at a right angle to the base electrodes, are patterned.

Matrices with 22×22 connections and widths of the tungsten wires of $2 \mu\text{m}$ have been fabricated. The resistor values for these matrices are few hundred $\text{k}\Omega$, and their standard deviation across a chip is less than 3%. Such matrices have been integrated in an associative memory circuit built with discrete amplifiers.

This type of resistive connection can be scaled down into the submicron range. With wires of $0.25 \mu\text{m}$ pitch, 12×12 matrices were fabricated. In this way, four resistors can fit into one square micron. This corresponds to a density of 4×10^8 connections per cm^2 and represents by far the highest connection density reported so far.

Primary References:

Hubbard, W. and D. Schwartz, J. Denker, H.P. Graf, R. Howard, L. Jackel, B. Straughn, and F. Tennant, "Electronic Neural Networks," in *AIP Conference Proceedings 151: Neural Networks for Computing*, J. Denker, ed., pp. 227-234, 1986.

Jackel, L.D. and R.E. Howard, H.P. Graf, and J.S. Denker, "Artificial Neural Networks for Computing," *J. Vac. Sci. Technology*, B4, p. 61, 1986.

A.1.2 AT&T Read-Only Network

In order to try out the fabrication for the high-density matrices on a real VLSI chip, a CMOS (complementary metal oxide semiconductor) test circuit was designed. It consists of a frame with 512 amplifiers plus the digital logic to load the data on and off the chip. The chip is fabricated in standard CMOS technology with 2.5 μm design rules.

The fabrication of these chips is done in a silicon foundry, except for the connections between the amplifiers. The connections are application-specific and are added in a final fabrication step in AT&T's laboratory. The amplifiers can be interconnected to form fully-interconnected networks with feedback or any type of feedforward network.

In one version, 256 of the amplifiers are connected to a matrix of aluminum and tantalum-silicide wires. These wires are placed in the silicon foundry and, at each crosspoint of two wires, space is provided where a resistor can be added. To customize the circuit, a film of amorphous silicon is evaporated onto the chip and the resistors are patterned with optical lithography or electron-beam lithography. So far, a few such chips have been finished and tested. The CMOS part of the circuit works properly – however, broken wires in the connection matrix prevented the full circuit from functioning. A revised version with a matrix connecting 192 amplifiers has recently been received from the silicon foundry.

Experiments have also been started to add resistor matrices with resistors stacked vertically between the crossing wires (see the description in section A.1.1). With line widths around 1 μm , all 512 amplifiers can be connected in this way.

The resistors consist of undoped amorphous silicon and their values are typically around 1 $\text{M}\Omega$. This is the proper range for networks with a few hundred neurons. The reproducibility of the resistor values lies within a few percent across a chip. So far, all the fabricated resistors had the same geometry, resulting in one value for the resistance. But the geometry of the resistors can easily be changed to give the connection strengths analog depth.

The circuit described represents a new approach to building large neural networks. By combining state-of-the-art CMOS technology with the high-density connection matrices described before, well over one thousand neurons can be packed on a single chip of a size less than 10 \times 10 mm.

Primary Reference:

Graf, H.P. and L.D. Jackel, R.E. Howard, B. Straughn, J.S. Denker, W. Hubbard, D.M. Tennant, and D. Schwartz, "VLSI Implementation of a Neural Network Memory with Several Hundreds of Neurons," in *AIP Conference Proceedings 151: Neural Networks for Computing*, J. Denker, ed., pp. 182-187, 1986.

A.1.3 AT&T 54-Neuron Programmable Network

This circuit consists of an array of 54 amplifiers fully interconnected through a matrix of programmable resistive interconnections. It is an experimental circuit designed to study the behavior of large analog networks. Its architecture is general enough that several different networks can be mapped into this circuit simply by programming the interconnections.

Each amplifier consists of two simple inverters connected in series; they work as high-gain, saturating amplifiers. A connection can be excitatory (+1), inhibitory (-1) or open (0), and its state is determined by the value of a two-bit weight stored in RAM cells nearby. Learning is done off-chip and the connection strengths are programmed by loading in all the weight values. The outputs of the amplifiers do not feed current directly into the input lines through the resistive connections. Instead, they control switches which, together with switches controlled by the weights, enable the current flow through resistors. In this way, the load an amplifier has to drive is minimal – only the capacitive load of the output wire. All the currents flowing into an input line of an amplifier are summed up and the voltage of a line adjusts to a value that the total current is zero. If the voltage of an input is above the switching threshold of the amplifier, its output goes high; otherwise, it stays low. This corresponds to an analog computation of the inner product between a weight vector and an input vector plus a thresholding of the result. Analog computation is used only within the connection matrix; all the other signals are digital.

The chip has been fabricated in standard digital CMOS technology with 2.5 μm design rules. It contains roughly 75,000 transistors in an area of 6.7×6.7 mm.

This circuit has been tested extensively working as pattern classifier and associative memory. In addition, it has also been used to parse sequences of vectors. As associative memory, it finds a best match among 10 vectors of 40 bits length in times between 50 ns and 60 ns. In pattern classification applications, it can store up to 54 vectors, each 54 bits long, and find close matches in less than 1 μs . The chip has been interfaced to a minicomputer and is used as a co-processor in character recognition experiments.

A new feature of this design is the interconnection element, which eliminates the requirement that the amplifiers drive all the currents for the analog computation. Instead, there are local current sources at each interconnection element that drive the currents. The chip is the first VLSI network this big with programmable interconnections, and the first VLSI neural network to be integrated in an application.

Primary References:

Graf, H.P. and P. deVegvar, "A CMOS Implementation of a Neural Network Model," in *Advanced Research in VLSI, Proceedings of the 1987 Stanford Conference*, P. Losleben, ed., p. 351-367, MIT Press, Cambridge, MA, 1987.

Graf, H.P. and P. deVegvar, "A CMOS Associative Memory Chip Based on Neural Networks," *Digest ISSCC*, L. Winner, ed., pp. 304-305, 1987.

A.1.4 AT&T Pattern Matching Network

This circuit searches a set of stored vectors for close matches to an input vector. It has been designed for pattern recognition applications and machine learning (e.g., clustering). The design is based on the experience gained with the 54-“neuron” chip.

The architecture of the circuit corresponds to a one-layered network with 96 binary input units and 46 output units. A matrix with 4,416 resistive elements connects the input units with the output units. Up to 46 vectors, each 96 bits long, can be stored in this matrix, where each vector occupies one row. The resistive connections are programmable with binary values and the weight value is stored in a static RAM cell near the connection. An input vector is compared in parallel with all the stored vectors, and the inner product between the input vector and all the stored vectors is computed. If a result exceeds a certain threshold, a flag connected to that vector turns on; otherwise, it remains off. The thresholds are programmable and can be set individually for each vector.

Analog current summing is used to calculate the inner products. Analog signals are used only within the interconnection matrix, and outputs are digital signals. A shift register transports the input vector along one side of the connection matrix. In this way, a new input vector can be ready for a computation at each clock cycle when, for example, an image is scanned for features.

The circuit is designed to go through a full computation cycle in less than 100 ns. This time includes resetting the network and preparing the input data for a run; the analog computation takes only around 20 ns. When running at full speed, the circuit evaluates around 500 million inner products of 96-bit vectors per second. The performance of the circuit is actually limited by the time required to read out the results and not by the time needed for the analog computation.

The chip has been fabricated in standard digital CMOS technology with 2.5 μm design rules. It contains over 70,000 transistors in an area of 6.7×6.7 mm. It is now being tested and all modules have been determined to work properly. Three versions of this chip have been designed to try out two different input structures and two different versions of comparators.

New in this design is the way the analog signal representing the result is sensed. By using a direct comparison of currents, it was possible to reduce the time needed for the sensing to around 10 ns and keep power consumption at moderate levels. The whole architecture of the circuit, which is optimized for the pattern matching operation, is also new.

Primary Reference:

Not published yet; designed by H.P. Graf.

A.1.5 AT&T Programmable Analog Synapse Matrix

Two chips which are generic test vehicles for studying VLSI adaptive learning have been submitted for fabrication. The chips are organized as analog matrix multipliers with voltage inputs and current outputs.

The analog lines are all brought off-chip, which makes it possible to combine several chips to form a larger network.

The weights are stored as the voltage difference between a pair of accumulation-mode MOS capacitors. Charge is moved between the capacitors by clocking a pair of long transistors in series with three pass transistors, mimicking a charge-coupled device (CCD). In the current conservative designs, at least six bits of analog depth are expected, limited by parasitic capacitances between the long transistors. A transconductance amplifier is used as a two-quadrant multiplier. Since each synapse outputs a single-ended current, the outputs can be summed by a single wire.

At each weight, a switching matrix controlled by a pair of RAM cells connects the charge injection transistors to the global control lines. The response to a global “reinforcement” signal can be either an increment, a decrement, or nothing – depending on the states of the RAM cells and the clocking of the global signals. By modifying the clocking sequence, a decay operation, in which selected weights are multiplied by a constant k ($k < 1$), can also be obtained. The decay operation effectively extends the dynamic range available for learning.

A $1.25\ \mu\text{m}$ CMOS version of the chip has 1,104 connections organized as a 46×24 matrix. Since the weights are stored in the form of charge on capacitors, special attention has to be paid to leakage currents. Measurements on test structures at room temperature gave 1% leakage in about 300 s with a $500\ \text{mm}^2$ capacitor as used on the chip. At 77 K in liquid nitrogen, the leakage is undetectable. A 10^4 increase in hold time from the -55 C drop obtainable from a thermo-electric cooler is estimated. The substantial area taken up by the capacitors and multipliers is easily used to isolate the access transistors from non-equilibrium processes associated with the digital circuitry. A similar but smaller chip is also being fabricated in $2.5\ \mu\text{m}$ CMOS.

The concept described represents a new approach to storing analog weights in a small area with a standard CMOS technology. In learning processes, the weights are integrated in a feedback loop and they are updated frequently. Therefore, a “semi-permanent” storage method for the weights is adequate. Moreover, cooling a circuit a few tens of degrees represents a much more economical solution to the leakage problem than developing a new fabrication process.

Primary Reference:

Schwartz, D.B. and R.E. Howard, “A Programmable Analog Neural Network Chip,” *Digest IEEE Custom Integrated Circuits Conference*, to be published May, 1988.

A.1.6 AT&T All-Digital Hamming Network

This is a fully-pipelined bit-serial classifier that outputs a list of the five best matches to an input vector found from a list of features stored on the chip. The chip was designed to compare analog and digital implementations of neural network algorithms and is intended for use in pattern recognition applications, such as image processing and speech recognition. High throughput is achieved by dedicating a separate processor for every stored vector and having them run in parallel. Unlike its analog counterparts, the

design enables chips to be connected together to extend the pipeline without loss of throughput whenever more stored vectors are required.

The design contains 50 processors per chip, chained together along a one-bit-wide data pipeline, a one-bit-wide match pipeline, and 12 one-bit-wide control pipelines. All calculations are bit-serial. Each processor has next to it a 128-bit ring register containing a stored vector. The Hamming distance – i.e., total number of bits difference – between the input vector and a stored vector is summed in an accumulator. At the time a distance computation is completed, the match list for the input vector arrives at the comparator. It is an ordered list of the five lowest Hamming distances found in the pipeline so far, together with the associated tag strings. If the match found by the current processor is better than one of the matches in the list, the current distance and tag are inserted. After the last processor in the pipeline, the list contains the best five distances overall, together with the tags of the stored vectors that generated them. The data stream and the list stream are loaded into 16-bit-wide registers for output.

The chip is designed to operate at an on-chip frequency of at least 100 MHz, and uses non-overlapping two-phase clocks. This high speed is possible because stage sizes have been kept to two gates or fewer and data paths are short. The chip will produce a list of the five best distances and tag strings every 1.3 ms with a latency of about 2.5 ms. A string of 1,000 chips containing 50,000 stored vectors will have a latency of only 2.5 ms. This chip will be fabricated in 1.25 μm CMOS technology and will contain roughly 300,000 transistors. It is presently in the final design stage.

Primary Reference:

Mackie, S. and J. S. Denker, "A Digital Implementation of a Best Match Classifier," *Digest IEEE Custom Integrated Circuits Conference*, to be published May, 1988.

A.1.7 Bellcore Stochastic Network

Joshua Alspector, Robert B. Allen, Victor Hu, and Srinagesh Satyanarayana of Bellcore have designed a test chip that can use a variety of learning algorithms that operate on a recurrent, symmetrically connected, neuromorphic, Hopfield-style network that, like the Boltzmann machine, settles to a global minimum in its Liapunov function in the presence of noise. These networks learn by modifying synaptic connection strengths on the basis of correlations seen locally by each synapse. The correlational synapse, when combined with a stochastic decision rule using physical noise, and weight saturation and decay, can create networks that learn in a supervised, unsupervised, or reinforcement manner, thus providing a basis for unifying these diverse learning paradigms.

The chip is currently in fabrication in 2 μm CMOS. The components of the test chip are a noise amplifier, a neuron amplifier, and a 300-transistor adaptive synapse, each of which is separately testable. These components are also integrated into a six-neuron and 15-synapse network. The synapse occupies an area of about $400 \times 600 \mu\text{m}$, and the neuron is about half that.

It will be operated as a supervised learning chip where digital training signals are applied at the input and output neurons. An analog waveform for the simulated annealing cycle will also be applied. There are no reserved neurons for input, hidden units, or output. Initial connectivity and weights are set by shifting in

five bits of information through the weight-controlling flip-flops. The weights can also be shifted out after learning for examination. The network is potentially fully connected. The annealing cycle will allow about 10,000 pattern presentations per second. This has a slow dependence on the number of neurons so that there is a speed-up of roughly 10,000 times the number of neurons squared over an equivalent simulation on a VAX 11/780. The speed-up is due to parallel analog computation for summing and multiplying weights and activations, and the use of physical processes for generating random noise.

Primary References:

Alspector, J. and R.B. Allen, "A Neuromorphic VLSI Learning System," in *Advanced Research in VLSI: Proceedings of the 1987 Stanford Conference*, P. Losleben, ed., pp. 313-349, MIT Press, Cambridge, MA, 1987.

Alspector, J. and R.B. Allen, V. Hu, and S. Satyanarayana, "Stochastic Learning Networks and their Electronic Implementation," to be published in *Proceedings of the conference on Neural Information Processing Systems – Natural and Synthetic*, D.Z. Anderson, ed., MIT Press, Cambridge, MA, 1987.

A.1.8 CalTech Artificial Retinas

The work of Carver Mead and his graduate students at the California Institute of Technology (CalTech) differs significantly from that of other neural network implementors in its goals and in the circuit techniques that it uses. It concentrates on the lowest level of neural processing: the preprocessing of sensory data, such as that in the retina. The architecture for this kind of processing involves fixed (non-adaptive) interconnections over very short range, typically nearest neighbors.

Mead's work also differs in the degree to which it attempts to mimic biology. It attempts to go beyond biological inspiration and to achieve what Mead refers to as "synthetic biology," where learning about the engineering issues faced by biological systems is an explicit goal. Following this constraint, the work stresses extremely low power consumption and fully analog circuitry operating in continuous-time.

The Mead silicon retina chips use conventional MOS transistor structures but unconventional circuit implementations. Light-sensitive parasitic bipolar transistors in the MOS structure provide a logarithmic light-intensity-to-current response, similar to that observed in biological vision, that achieves a constant contrast sensitivity over a very wide range of light intensities. The MOS transistors for the neural network operate in the rarely used subthreshold region, where gate voltages are low and drain currents extremely small. The side effect of large variance from device to device is viewed by Mead as a useful characteristic in so far as it forces the system design to deal with this issue and to achieve accurate system performance from components of low intrinsic accuracy.

Mead and his students have fabricated a number of simple retina chips with photosensors, resistive nearest-neighbor interconnections, and lateral inhibition. The chips have demonstrated image acquisition, motion detection, local contrast enhancement, and automatic gain control. The largest of the chips has implemented 48×48 photosensors in a hexagonal array.

In terms of the implementation taxonomy presented earlier, this approach offers synapses with fixed weights and continuous values, neurons with continuous values that evolve continuously in time, and external inputs that are applied continuously in time. The chip represents a direct embodiment of a neural network model of a retina. The technology is conventional CMOS as available through MOSIS, but the circuit design (operating conditions of the devices) is non-standard.

Primary Reference:

Sivilotti, M.A. and M.A. Mahowald and C.A. Mead, "Real Time Visual Computations Using Analog CMOS Processing Arrays," in *Advanced Research in VLSI, Proceedings of the 1987 Stanford Conference*, P. Losleben, ed., MIT Press, Cambridge, MA, 1987.

A.1.9 CalTech Optical-input Synapse Networks

Aharon Agranat, Charles Neugebauer, and Amnon Yariv at CalTech are building prototype neural networks based on a new generic architecture which they proposed.

The underlying principle is to store the synaptic interaction matrix in an optical spatial light modulator and to image it in parallel onto a detector array which is the input of the processing chip. Thus parallel processing in silicon is enabled, while the interconnectivity problem is avoided.

Three different versions are now being built:

1. A semi-parallel synchronous neural network with binary neurons based on CCD technology. The details of this version are given in the reference. The state of the art of CCD technology enables the building of 1,000-dimensional neural networks with complete update in 0.1–1.0 ms. A first chip with the various building blocks (integrators, etc.) required for this architecture has been fabricated and tested, and a second-generation chip is now being designed.
2. A parallel synchronous neural network with either analog or binary neurons using CID (charge injection device) technology. A first chip with the building blocks for this version has been designed and fabricated and is now being tested.
3. A parallel asynchronous neural network with binary or analog neurons, in which the synaptic interaction strength is transformed into light intensity which is continuously transformed into current at the detector array. A first chip of this version, with 32 neurons, is now being tested.

In conclusion, it should be pointed out that this approach includes two main advantages:

- It can be realized immediately with present-day silicon technology, and
- The information is loaded optically into the chip and, once loaded, can be operated upon or modified optically, thus opening the way to learning.

Primary Reference:

Agranat, A. and A. Yariv, "A New Architecture for a Microelectronic Implementation of Neural Network Models," in the *Proceedings of the IEEE First Annual International Conference on Neural Networks*, San Diego, June, 1987.

A.1.10 JPL Discrete Component Network

Researchers at the Jet Propulsion Laboratory (JPL) have developed an electronic programmable neurocomputer based on off-the-shelf discrete components and a semi-parallel analog-digital hybrid architecture which can significantly reduce the hardware complexity of an electronic neural network. The hybrid architecture utilizes high-density digital memories to store the large quantity of information in a synaptic network and exploits the high-speed capability of neurons in analog hardware for parallel information processing. The present prototype hybrid neurocomputer, with 32 neurons, features 1,024 programmable synapses with three levels of synaptic connection strength. Additional memories are currently being incorporated to increase the number of gray levels.

The system is distinguished by its configurability; it can simulate a neural network with either a feedback or feedforward architecture, in a continuous-or discrete-time mode. The system operation can also be suspended momentarily at the end of each iteration cycle to study the dynamic evolution of the network into a stable state. Since the complete physical interconnections between neurons are no longer required, the hybrid neurocomputer can easily be expanded in size to realistically simulate large neural networks. The prototype hybrid neurocomputer has demonstrated a many-fold speed improvement relative to software-based simulations of neural networks. For a 1,000-neuron system, this architecture promises a processing speed of over 100 million interconnects per second.

The analog-digital hybrid neurocomputer consists of three basic components: digital memory for the synaptic data, an array of binary-weighted resistors or multiplying digital-to-analog converters for quantized gray-scale synapses, and an array of threshold amplifiers (neurons) with input and output sample-and-hold units. During system operation in a feedback configuration, a neuron input is updated with a weighted sum of the present voltage outputs of the neurons. The synaptic data from the memory set the resistance values of the synaptic array to provide the appropriate weights for the selected neuron. The neuron inputs are updated sequentially to complete a single iteration cycle.

Primary Reference:

Moopenn, A., *et al.*, "A Neurocomputer Based on an Analog-Digital Hybrid Architecture," in *Proceedings of the IEEE International Conference on Neural Networks*, M. Caudill and C. Butler, eds., Vol. 3, p. 479, 1987.

A.1.11 JPL Cascadable VLSI Synaptic Chip

A programmable binary synaptic chip as a basic building block for electronic neural networks has been designed and fabricated at JPL. The chip, implemented in $3\text{-}\mu$ bulk CMOS technology, features complete programmability of the 32×32 resistive binary connections (synapses) in a feedback or feedforward architecture. For maximum versatility, flexibility, and expandability, the processing units (neurons) have been kept off-chip.

The unique feature of this chip is its cascadability and the use of long-channel MOSFETs (metallic oxide semiconductor field-effect transistors) for uniform, high-resistance synapses. It can be cascaded not only to form larger binary synaptic arrays with sizes and shapes of the user's choice, but it can also be "stacked" in parallel to form quantized gray synapses. As a research tool, a neural network system-based on these chips offers a significant speed advantage over software-based simulations, particularly in the feedback configuration. The convergence time of a chip-based feedback network (Hopfield model) is on the order of a microsecond.

The matrix chip contains a 32×32 array of synaptic cells with long-channel MOSFETs, each functioning as a two-state (ON/OFF) resistive connection. The long-channel MOSFET-based "resistors" provide the weak ON connection resistance ($\sim 300\text{ K}\Omega$) for low static power consumption, and the precision ($\pm 3\%$) necessary for proper threshold operation of the neurons. Programming or updating of the synapses is accomplished by either a shift-register memory in one design or addressable latches in another design. In an alternative approach, thin-film (germanium-copper alloy) resistors have been deposited in series with the FET switches in an additional special processing step to the normal CMOS fabrication run to obtain the desired uniform high-valued connection resistance. This approach provides linear and bidirectional resistive synapses, in contrast to the transistor-based "resistive" synapse.

Primary References:

Thakoor, A.P., *et al.*, "Electronic Hardware Implementations of Neural Networks," *Applied Optics*, Vol. 26, No. 23, p. 5085, 1987.

Moopenn, A., *et al.*, "Programmable Synaptic Chip for Electronic Neural Networks," to be published in *Proceedings of the IEEE conference Neural Information Processing System – Natural and Synthetic*, MIT Press, Cambridge, MA, 1987.

A.1.12 JPL One-time Programmable Synaptic Arrays

Electrically programmable, write-once, non-volatile thin film (40×40) prototype synaptic arrays based on memory switching in hydrogenated amorphous silicon have been demonstrated for the first time by JPL. At each node of the matrix is a normally-OFF microswitch of hydrogenated amorphous silicon in series with a current-limiting resistor. Resistivity-tailored cermet, or amorphous-germanium-metal-alloy thin films, provide the required high resistance ($> 10^6\Omega$) at a node in its ON state. By applying a short voltage pulse of sufficient amplitude, the microswitch is permanently switched ON and the resistance is essentially determined by the current-limiting resistor. Since $\alpha\text{-Si} : \text{H}$ requires only a very small amount

of switching energy (about one nanojoule per cubic-micron), it can easily be switched through a current limiting resistance of greater than one megohm.

Hopfield's neural network model with feedback has been implemented using the thin-film-based, programmable, non-volatile synaptic array and a set of discrete-component neurons (current-summing amplifiers followed by comparators). Although connecting several programmable binary synapses in parallel provides some quantized gray scale in such arrays, the true potential of such synapses is in its possible ultra-high density, since the size of a connection is essentially that of the thin film wire intersection.

A feedforward, line-by-line-addressed, parallel-read-out, programmable (once), non-volatile high-density memory with limited associative nature and fault tolerance (associative reflex memory, or ARM) with no moving parts has been proposed and is currently under development at JPL especially for space and defense applications.

Primary References:

Thakoor, A.P., *et al.*, "Binary Synaptic Connections Based on Memory Switching in α -Si : H," in *AIP Conference Proceedings 151: Neural Networks for Computing*, J. Denker, ed., p. 426, 1986.

Daud, T., *et al.*, "Neural Network Based Feed-Forward High Density Associative Memory," *IEEE/IEDM Technical Digest*, p. 107, Washington, DC, Dec., 1987.

Daud, T., *et al.*, "Feed-forward, High Density, Programmable Read-Only Neural Network Based Memory System," *SPIE Proceedings of the O-E/LASE '88 Symposium*, Los Angeles, Jan., 1988.

A.1.13 JPL Thin-film Solid-state "Memistor"

The first solid-state, non-volatile, modifiable, analog memory resistor (memistor) has been developed by JPL. A gate-controlled, reversible injection of H⁺ ions in electrochromic thin film of WO₃ modulates its resistance. An array of 16 such devices has been fabricated on a SiO₂-covered silicon wafer and has been tested in the classic "Adaline" configuration.

The true analog-resistive nature, excellent stability, cyclability, and extremely simple device structure are the distinguishing marks of this synaptic array.

A hygroscopic thin film of Cr₂O₃, separated from the active WO₃ layer by a thin "blocking" SiO layer, acts as a source of H⁺ ions. Aluminum is used as the gate electrode and Ni provides the contact leads to WO₃. The resistance of the device can be tailored and stabilized at any value over a wide dynamic range (~4 orders of magnitude). The programming speed is controlled by the selected gate voltage.

Primary Reference:

None yet.

A.1.14 MIT/LL CCD Gaussian Retina

Jay Sage invented and, together with Analisa Lattes, demonstrated a retina chip that performs the computationally-intensive Gaussian convolutions on a detected image that are required for the difference-of-Gaussians (DOG) edge detection algorithm. Although originally conceived as a special-purpose imaging device, the characteristics of the chip are those of a neural network.

The network was implemented using a standard four-phase charge-coupled device (CCD) frame-transfer imaging chip with a 384×288 array of pixels. Image detection was performed in the conventional way. However, before the image was clocked out of the chip, some special charge transfer operations that allowed neighboring pixels to interact were performed.

Each photosensor on the chip can be regarded as a neuron that is coupled to its nearest neighbors in two dimensions. Thus the chip contained approximately 110,000 neurons. To conform to NTSC television standards, only 256 of the 288 lines were used for the experimental demonstration, resulting in just under 100,000 active neurons. The special CCD clocking operation established connection weights to the neighbors such that in each processing cycle the image represented by the activity pattern in the neurons was convolved with a small Gaussian kernel. The processing cycle could be repeated to achieve a wide range of Gaussian kernels.

The area imaging chip used in the experimental demonstration could not actually perform parallel processing in two dimensions; instead the horizontal processing was performed sequentially on each line. Each step in the sequence involved 384 neurons operating at an interconnect rate of 7.2 MHz (twice the color subcarrier frequency), for a total of about 3 Gc/s (3 billion interconnects per second). A full two-dimensional implementation running at a connection rate of only 1 MHz would have achieved an interconnect rate of 100 Gc/s. In practice, this was higher speed than was needed. The semi-parallel implementation was already fast enough to allow all of the processing to be performed during the blanking intervals (retrace times) of the NTSC television signal.

This implementation illustrates a neural network with fully-analog neurons operating synchronously with sampled inputs. The effectively binary synaptic weights were fixed by the structure of the network, with nearest-neighbor neurons interconnected (weight +1) and all other neurons not connected at all (weight 0). The chip represents an imbedding of a computational algorithm in a neural network circuit. Standard CCD imager technology was employed. The implementation approach is limited to this particular problem.

Primary Reference:

Sage, J.P. and A.L. Lattes, "A High-speed Two-dimensional CCD Gaussian Image Convolver," *MIT Lincoln Laboratory Solid State Research Quarterly Technical Report*, August-October, 1986.

A.1.15 MIT/LL MNOS/CCD Learning Network

Jay Sage, Karl Thompson, and Richard Withers of Lincoln Laboratory have built a prototype neural network based on MNOS (metal nitride oxide semiconductor) and CCD (charge-coupled device) technologies. The chip implements a 13×13 fully connected coupling matrix built using $4\text{-}\mu\text{m}$ CCD design rules. Only the coupling matrix and its input/output drivers are on the chip. External differential amplifiers and sample-and-hold circuits were used to produce the neural output signals.

Two novel features distinguish this design from other electronic applications built to date. First is the programmable analog representation of the stored coupling weights. Weights are stored as long-term charge in the nitride layer of the MNOS devices. In other designs, weights are either fixed binary values or quantized to a few bits of digital storage.

Second is the summation of neural inputs by accumulation of charge on a conductor rather than by summing of currents at a node. The network operates by a multi-phase clocked iteration cycle. The axonal signal on each row of the coupling matrix determines which of two sets of MOS capacitors gets charged, while the amount of charge in each packet is controlled by the coupling weights (in the form of stored charges in the nitride layer). After the charge packets are formed, they are clocked onto output diffusions and linearly sensed by on-chip amplifiers. There are separate summing diffusions and outputs for excitation and for inhibition. Off-chip comparators provide differential sensing and thresholding. Because of the gated charge transfer in this specific implementation of the technology, the neuronal states are limited to binary values.

The chip is structured to implement the outer product associative memory algorithm. A vector to be stored is presented to the coupling network in both the row and column dimension, and the outer product is formed at the matrix points in the form of a fixed amount of charge wherever the outer product has a 1. This charge is then transferred into the nitride layer by a special, high-voltage bias signal. Experiments showed the ability not only to learn a vector in this way but also to learn independently and incrementally a second vector. This was the first neural network chip to perform its learning operations on the chip.

In terms of the implementation taxonomy, this approach offers synapses with continuous analog values of the weighting coefficients. These weights are electrically reprogrammable and can even be incrementally programmed. Although only binary neurons were supported in this specific chip, continuous analog neural states could be supported. The operation of the chip is fully synchronous, in terms of both the presentation of external inputs and the internal processing. The technology is entirely electronic, using a special MNOS memory device and CCD operations.

Primary Reference:

Sage, J.P. and K. Thompson and R. S. Withers, "An Artificial Neural Network Integrated Circuit Based on MNOS/CCD Technologies," *AIP Conference Proceedings 151: Neural Networks for Computing*, J. Denker, ed., 1986.

A.1.16 MIT/LL Wafer-scale Neuromorphic Architecture

This work consists of a design for a wafer-scale neural network and the building and testing of a chip embodying the coupling components. The wafer design is based on a Lincoln Laboratory technology for making and cutting connections between conductors on a silicon wafer using laser pulses. The coupling elements on the wafer will be arranged in fully-connected blocks having a modest number of inputs and outputs (say, 32). Between the blocks will be groups of uncommitted conductors that run the length (on one metal level) or width (on the other metal level) of the wafer. The neural amplifiers at the output edge of each block could be wired, by laser linking and cutting of the conductors, to inputs on their own block or other blocks. In this way a generic wafer design could be customized by laser structuring into a variety of different wafers with different network topology.

The coupling elements are CMOS multiplying digital-to-analog converters (MDACs). The test chip had 1,024 MDACs arranged to couple 32 inputs to 16 neural amplifiers (external to the chip). Each coupling site needed two MDACs – one for excitation and one for inhibition. The programmable, discrete synaptic weights for the coupling sites were stored in 4-bit-plus- sign digital registers configured as a static RAM. The MDAC operated by successive bits turning on or off 1-, 2-, 4-, or 8-unit currents through CMOS transistors. The size of a unit current was proportional to the input voltage to the MDAC (the output voltage of an amplifier representing the continuous state of some neuron). Therefore, the MDAC effectively multiplied the input voltage by the stored weight, with the product in the form of a current. These currents were summed (excitation and inhibition separately) by current-to-voltage transducer circuits and then subtracted in a differential amplifier. Another test chip with a modified MDAC design and on-chip amplifiers, both based on current-mirror circuits, is being fabricated. The first chip was used on a circuit board to implement some neural networks, each made of a correlator in series with a max-picker. The circuits performed satisfactorily, and uniformity of MDAC properties to within a few percent was observed across the chip.

In terms of the implementation taxonomy presented earlier, this approach offers synapses with multiple, discrete, reprogrammable values and neurons with continuous analog levels. External inputs are applied at discrete times, but internal neuronal processing takes place on a continuous (asynchronous) basis. The implementation is based on an all-electronic technology with conventional CMOS circuit elements interconnected in a special, non-standard way.

Primary Reference:

Raffel, J. and J. Mann, R. Berger, A. Soares, and S. Gilbert, "A Generic Architecture for Wafer-scale Neuromorphic Systems," *Proceedings of the IEEE First Annual International Conference on Neural Networks*, San Diego, CA, June, 1987.

A.1.17 MIT/LL CCD Neural Network Processor

A single-chip neural network processor that can be used to compute the matching score between two layers of neural nodes and to select and enhance the maximum has been proposed by Alice M. Chiang of

Lincoln Laboratory. The input layer consists of N analog nodes and the output layer has M analog nodes. Every input is connected to every output node via a programmable connection weight. For example, if the input layer contains 1,000 nodes and the output layer 20 nodes, the processor will provide 20,000 programmable connections. At each of the output nodes, the weighted sum of the N inputs will pass through an on-chip, charge-domain, nonlinear sigmoid-type detection circuit. A decision can then be made on the selected output.

The proposed processor consists of an N -stage floating-gate tapped delay line for shifting and holding the N analog input values, N CCD multiplying digital-to-analog converters (MDACs), and an $N \times M$ -stage digital shift register (SR) for shifting and holding the digital connection weights. For design plasticity, the N inputs will be loaded and stored in R CCD analog shift registers, each R stages long, where R equals the square root of N . For each MDAC, there are M words stored in bit-parallel digital shift registers, providing sequential access to the M words. At each stage of delay, the floating gate is coupled to the analog input port of the corresponding MDAC, and the output from each MDAC is a charge packet proportional to the product of the analog input and the digital weight. All the MDACs on the same row share a common output port. The outputs from each row can be summed together in the charge domain and form a single output. In addition, a one-stage bucket-brigade device (BBD) is placed at the MDACs' common output port. The BBD is to be designed with non-ideal charge transfer efficiency.

The CCD architecture for the two-layer neural network implementation is very plastic. For example, for a CCD network with 1,024 (32^2) input nodes and 20 output nodes, the processor will provide about 20,000 programmable connections. With minor modification at the device output port, the processor can be reconfigured into a two-layer network with 32 input nodes and 640 (32×20) output nodes. The device is capable of performing 10 billion computations per second with a 10 MHz clock rate. This kind of performance clearly demonstrates the adaptability and the computational power offered by CCD technology.

Primary References:

Chiang, A.M., "A New CCD Parallel Processing Architecture," in *VLSI Systems and Computations*, H. T. Kung, ed., pp. 408-418, Computer Science Press, 1981.

Chiang A.M. and B.E. Burke, "A High Speed CCD Digitally Programmable Transversal Filter," *IEEE J. Solid-State Circuits*, Vol. SC-18, p. 745, 1984.

Chiang, A.M., "A CCD Parallel Pulse-Doppler Radar Processor," *GOMAC Digest*, pp. 505-509, 1986.

Chiang, A.M. and P.C. Bennett, B.B. Kosicki, R.W. Mountain, G.A. Lincoln, and J.H. Reinold, "A 100-ns CCD 16-Point Cosine Transform Processor," *ISSCC-87 Tech. Digest*, pp. 306-307, February 1987.

Chiang, A.M., "CCD Retina and Neural Net Processor," Workshop on Hardware Implementation on Neuron Nets and Synapses, San Diego, CA, Jan., 1988.

A.1.18 MIT/LL CCD Retinas

The integration of signal processors with photosensors makes it possible to perform simultaneous, parallel computations on each pixel in real time. An interline-transfer area imager can be designed with integrated CCD signal-processing elements between neighboring pixels to simulate a solid state or a biological system with locally-connected interactions between its neighboring cells. A 128×128 Gaussian or a 64×64 Lorentzian retina is proposed by Alice M. Chiang of Lincoln Laboratory to perform such computations as time derivatives and minimization of the global spatial differences between neighboring pixels. The retina can be used for image restoration of a corrupted, noisy optical input, and also for motion detection.

Exploiting the equivalence between images and statistical physical systems, Geman and Geman have developed a highly parallel image restoration algorithm. Vold has extended this algorithm so that it can be implemented on a neural computing network. Each pixel (photodiode) on this retina would be connected to its nearest neighbors with the interconnection strength proportional to the gradient of a pair-interaction energy function $h(x)$, where x is the difference between the intensities of the two neighboring pixels. The energy function can be chosen to correspond to the probability distribution function of any typical image. For example, the energy function can be chosen as a Gaussian function or a Lorentzian function. If such a retina is used as an imager, the state (intensity) of all pixels will evolve to a minimum energy or maximum probability state. Therefore, this retina can be used for image restoration from a corrupted noisy optical input. The interconnection strength between pixels will be implemented by using single-channel CCDs developed by A. M. Chiang.

The read-out organization of the retina is based on an interline transfer CCD imager where the signal generated from each pixel is transferred to a vertical CCD linear shift register and all the vertical lines are clocked in parallel. This read-out scheme is chosen because it provides high quantum efficiency at the sensors, lower noise read-out than the conventional xy-scanner, and the possibility of integrating anti-blooming control at each sensor for large dynamic range applications.

For motion-detection purposes, the retina will be designed to have two CCD shift register stages at each pixel site rather than one, so that two consecutive time samples can be taken at each pixel. The time difference between the two samples can be programmable and controllable by the CCD clock rate and can be as fast as 50-100 ns. The device would have a horizontal CCD shift register to read out the image line by line. At the output diffusion node, a correlated double sampler would take the difference between two time samples and perform the motion detection function on each pixel in real time as it is read out.

The basic design concept can be extended to implement an on-center/off-surround, or an off-center/on-surround retina for edge detection. Next-nearest-neighbor or even higher-order interactions can be incorporated on a chip to allow one to build retinas for image segmentation or feature detection.

A.1.19 Rockwell Low-level Vision Networks

Bimal Mathur, Taihi Wang, Tom C. Tsen, and Emory Walton of Rockwell are designing chips for real-time image understanding systems.

Since low-level vision processes operate on sparse data, involve noise operations, and may not produce a unique output for a given input image, these processes fall in the category of "ill-posed" problems. These problems can be converted to "well-posed" problems by including certain constraints. It has been proposed that these problems can be solved using resistive networks. In practice, these networks can be realized as Hopfield networks with linear neurons. However, these researchers' investigations point to the necessity of using nonlinear neurons.

These researchers are in the process of designing chips which will perform edge detection, stereopsis, and estimation of the motion field. Each of these chips is visualized as a custom analog chip which will be flip-chip bonded (using indium columns) via a matrix of $N \times N$ connections to an I/O chip. The I/O chips will consist of suitable circuits to read/write analog voltages to the analog chip. In addition, the I/O chips may be used for programming weights and setting binary switches to perform data fusion. Since edge detection is believed to be the first process in low level vision, the researchers have started with this process.

The edge-detection algorithm is implemented by selecting a suitable interconnection pattern and the steady-state response of the network is the desired solution. The interconnection pattern is translation-invariant and needs connections up to fourth neighbors. However, it has been shown by UCLA that a good approximation can be achieved by using empirically derived second-neighbor connections. A chip based on this approach is being designed by Joe White, Bruce Furman, Assad Abidi, and Rich Baker of UCLA in collaboration with Rockwell. This chip is based on analog circuits and will be fabricated in a MOSIS 3μ CMOS process. The array size will be 16×16 .

An analysis of the edge-detection algorithm which is implemented in the chip described above shows that the operation of the network is equivalent to a two-dimensional convolution of the image with a Gaussian function. The analog network uses active components to achieve this.

An alternate architecture which depends on the transient response of a non-uniform RC plane is being developed at Rockwell. In this approach, the researchers are relying on generating a circularly symmetric point-spread-function by driving the RC plane for a length of time and then measuring voltage distributions after a suitable delay time. Each of these samples is a two-dimensional convolution of the image with the point spread function. The array size for this chip will be 32×32 .

It is visualized that several of the I/O chips will be integrated on a single wafer to solve the problem of inter-module communication. On different sites of this wafer, chips based on different technologies (designed to perform different tasks) will be bonded using indium-column technology. This technology is currently being used to manufacture 128×128 HgCdTe-based infrared (IR) focal plane arrays which are mated to a CMOS I/O device.

Primary Reference:

1988 Custom Integrated Circuits Conference.

A.2 OPTICAL IMPLEMENTATIONS

A.2.1 BDM/NRL Attentive Associative Memory Network

Ravi Athale and Carl Friedlander of BDM and Harold Szu of Naval Research Labs (NRL) have developed a prototype neural network based on off-the-shelf LED (light-emitting diode) and PIN photodiode components and electronic amplifiers. This prototype implements a novel associative memory model, called attentive associative memory, that stores and retrieves four 16-bit binary vectors. In this prototype, the attention (or the *a priori* expectation of stored states) is manually controlled using amplifier gain and/or off-set.

Two features distinguish this prototype from the earlier opto-electronic auto-associative memory model demonstrated by Psaltis and Farhat. The first is the storage of vectors individually, as against the outer product rule. This makes the inner product values of the stored vectors with the input vector accessible. A non-uniformly nonlinear operation on these values allows suppression of crosstalk and incorporation of *a priori* expectation.

The second feature is a lensless design, which in its fully developed form will involve a two-dimensional spatial light modulator with elongated fingers. This allows for a compact system that is light-efficient and permanently aligned.

The prototype consists of two opto-electronic planes, each containing a two-dimensional array of LEDs and photodiodes. The LEDs and the photodiodes along a column (row) are interconnected such that the output of the photodiode column (row) is proportional to the sum of light intensities on the individual elements; this output is nonlinearly amplified with electronic amplifiers and applied to the adjacent LED column (row). The light outputs of all the LEDs along that column (row) are identical. The two opto-electronic panels are assembled with a film mask sandwiched between them such that the LEDs in one panel illuminate the photodiodes in the other panel after being modified by the film mask. The data vectors are encoded along rows of the film mask. The LEDs in the first panel and photodiodes in the second panel calculate the inner products between the input vectors and the stored vectors in parallel. The LEDs in the second panel and the photodiodes in the first panel calculate a linear superposition of all the stored vectors, which is then input (after thresholding) to the inner product operation described before, thus completing one iteration.

The attentive associative memory model, in essence, finds a state that is simultaneously consistent with the *a priori* knowledge constraints in the inner product domain and in the data domain, as well as the input values. The stored vectors are among the stable states, although for a distorted input, the stable state need not correspond to the stored vector with highest correlation, making this model distinct from other associative memory models. The ultimate implementation of this compact architecture will use electro-optic materials in the spatial light modulators with inherent nonlinear transfer function, obviating the need for electronic amplifiers.

Primary Reference:

Athale, R.A. and C.B. Friedlander and B.G. Kushner, *Proceedings of SPIE*, Vol. 625, p. 179, 1986.

A.2.2 NRL Adaptive Realtime Holographic Network

A group headed up by Arthur Fisher in the Optical Sciences Division of NRL has, since 1984, been developing optical implementations of artificial neural networks with powerful adaptive learning capabilities. Other participants in this research have included: Wendy Lippincott and John Lee at NRL, Lee Giles of AFOSR, and Robert Fukuda of Sachs Freeman Associates. A variety of versatile adaptive learning optical architectures have been investigated and a few have been constructed and experimentally demonstrated. Realtime holographic implementations and five distinct optical architectures employing spatial light modulators (SLMs) have been investigated. The SLM-based architectures implement three types of learning dynamics: Widrow-Hoff (also known as LMS, or least mean square), Hebbian conjunction of activation (adaptive outer product), and conjunction of differences. The holographic architectures also have adaptive learning capabilities, but trade some aspects of performance for reduced implementational complexity. A variety of means have been proposed for tackling the “holographic adaptive encoding problem” (see primary reference, including hashing techniques, look-up table encoding holograms, and one-dimensional encoding).

Both Widrow-Hoff and Hebbian SLM-based architectures have been built and successfully demonstrated. These experimental architectures are being operated with up to 10 vector elements and 100 synapses. The current technology could conceivably support up to about 1,000 neurons and 10^6 synapses, with adaptation of all the synapses occurring in 1 ms (10^9 synapses/s) and all the recall operations (10^6 multiplications and 10^6 additions) occurring in 10^{-9} s (10^{15} ops/s, limited only by light propagation speeds).

These are, essentially, all-optical architectures, with both the learning and recall operations being accomplished optically, without the aid of any adjunct electronic computations. In the optically-addressed SLM architectures, the requisite additions, subtractions, multiplications for inner and outer matrix products, and actual updating and storage of the synaptic weight matrix are accomplished by the SLMs. All these architectures can also be operated in either a spatially-discrete matrix algebraic mode or with continuous resolution where the primitive operations become overlap integrals, convolutions, and correlations. Continuous resolution operation generally results in higher information capacities.

This research is not an exercise in the optical implementation of algorithms, but has attempted to develop optical associative networks which offer the performance capabilities which will ultimately be required by anticipated applications. Hence the emphasis on adaptive learning, characterized by such advanced capabilities as (see primary ref.): incremental learning, gated learning, gradient-driven learning, multi-layer and temporal credit assignment, controlled forgetting with no arbitrary decay, non-saturating weights, error-correcting feedback, convergence to optimum pseudoinverse associations, and minimal *a priori* data constraints. These associative architectures can be configured to implement a variety of recall formulations, ranging from simple matrix-vector inner products to nonlinear, recursive dynamic models (e.g., Hopfield) or optical resonator recall configurations. The required nonlinear operations are implemented by SLMs. Most of these architectures are also optically cascadable modules, with all input and output information patterns in optical form. These cascadable modules have been designed to be optically interconnected and cascaded to construct more sophisticated multi-layer artificial neural network architectures for solving particular problems.

Primary Reference:

Fisher, A.D. and W. Lippincott and John N. Lee, "Optical Implementations of Associative Networks with Versatile Adaptive Learning Capabilities," *Appl. Opt.*, 26, p. 5039, 1987.

Related Publications:

Fisher, A.D. and W. Lippincott, and John N. Lee, "Optical Implementations of Associative Networks with Versatile Adaptive Learning Capabilities," *Appl. Opt.*, 26, p. 5039, 1987.

Fisher, A.D., "On Applying Associative Networks: An Approach for Representation and Goal-directed Learning," *Proceedings of the IEEE First Annual International Conference on Neural Networks*, San Diego, CA, June, 1987.

Fisher, A.D. and John N. Lee, "Optical Associative Processing Elements with Versatile Adaptive Learning Capabilities," *Technical Digest of the Second OSA Topical Meeting on Optical Computing*, p. TuA-5, 1987.

Fisher, A.D. and R.C. Fukuda and J.N. Lee, "Implementations of Adaptive Associative Optical Computing Elements," *Proceedings of the SPIE*, Vol. 625, p. 196, 1986.

Fisher, A.D. and C.L. Giles and J.N. Lee, "An Adaptive Associative Optical Computing Element," *Technical Digest of the Topical Meeting on Optical Computing*, p. WB4, 1985.

Fisher, A.D. and C.L. Giles, "Optical Adaptive Associative Computing Architectures," Invited Paper, *Digest of Papers, IEEE COMPCON '85*, p. 342, 1985.

Fisher, A.D. and C.L. Giles, and J.N. Lee, "Associative Processor Architectures for Optical Computing," OSA Annual Meeting, *Journal Opt. Soc. Am.*, A, 1, p. 1337, 1984.

A.2.3 CalTech/JPL GaAs Optoelectronic Neural Chip

J. Katz, J.H. Kim, and A. Nouhi of the Jet Propulsion Laboratory, CalTech, and D. Psaltis and S. Lin of CalTech are in the process of developing a prototype neural network based on a GaAs monolithic two-dimensional array of opto-electronic neurons operating in conjunction with a volume hologram. The opto-electronic integrated circuit (OEIC) under development will implement 100 neuron elements (in a 10×10 array configuration), and both interconnects and input/output signals will be optical.

The main novel feature of this effort is that it is the first opto-electronic implementation of neural networks that is based on monolithic integrated opto-electronic circuits, rather on discrete devices. Since all the neuron elements on the OEIC are not connected electrically, and the interconnects are specified optically, the OEIC can be a basic building block for several types of neural networks. For example, the use of a reprogrammable hologram (e.g., based on semiconductor photorefractive crystals) can lead to adaptable networks.

Each neuron element of the OEIC consists of a photodetector, a saturating amplified (to implement the thresholding function), and a light source. In the current OEIC under development, the light source

is a light-emitting diode (LED), and the saturating amplifier is a bipolar-transistor Darlington pair (the high current gain is required because of the lower efficiency of the LED). The front transistor is a phototransistor, so it also serves as the photodetector. This helps simplify the OEIC design and fabrication. A common electrical connection to all the bases of the transistors enables control of the threshold of the neuron elements. The current OEIC layout also enables testing of subgroups within the entire array. The next generation of the OEIC will incorporate vertically-emitting laser diodes instead of LEDs in order to increase the OEIC power efficiency.

The status of this effort in early 1988: fabrication of several initial OEICs, which involves a six-mask process, has been accomplished, and testing and evaluation is starting. A process, based on reactive-ion-etching (RIE), is being developed for fabrication of etched mirrors for vertically-emitting lasers.

Since this work has just started recently, no published reference in the open literature is yet available.

A.2.4 Northrop All-optical Network

Harold M. Stoll, Li-Shing Lee, and Michael Tackitt of the Northrop Research and Technology Center have built an all-optical, continuous-time recurrent neural network. The network is a ring resonator which contains a saturable, two-beam amplifier (barium titanate); two-volume holograms (iron-doped lithium niobate); and a linear, two-beam amplifier. The saturable amplifier permits, through the use of a spatially-patterned signal beam, the realization of a 23×23 -neuron array; the two-volume holograms provide global network interconnectivity (279,841 interconnections); and the linear amplifier supplies sufficient cavity gain to permit resonant, convergent operation of the network.

Novel features which distinguish this network from other all-optical architectures include fully-adaptive; bipolar, and (potentially) asymmetric interconnects; low-noise recall of stored attractors; and enhanced algorithmic flexibility. Adaptive, bipolar interconnects result from the use of coherent training techniques to store attractors as volume holograms within crystals of photorefractive lithium niobate. Low-noise recall results from operating the network in a super-radiant or above-threshold mode. Algorithmic flexibility is achieved through the use of a true network architecture (with nonlinear processing in neural state space) which permits the execution of a broad class of energy-minimizing neural network algorithms.

The two-volume holograms are used to realize an inner-product-type neural connectivity with Hebbian learning. Fourier and state-space holograms of a given training set pattern are generated using a common, plane reference beam which is made coherent with respect to all other training pattern reference beams. As a result of their mutual coherence, training set patterns may therefore be used to structure an interconnect matrix which incorporates inhibition (negative matrix elements) as well as excitation (positive matrix elements). Moreover, through the use of a (nominal) 90-degree angle between reference and object beams, the network is capable of accommodating very large training pattern sets (1,000-10,000 members).

Both simulations and laboratory demonstrations have been performed. Two-neuron simulations indicate that, despite considerable internal differences, the network behaves in much the same way as a continuous-time version of either Anderson's BSB model or Hopfield's spin-glass model. Laboratory demonstrations have included storage and low-noise (speckle-free) recall of one of two non-orthogonal stored images (M-60 tanks viewed from different angles).

Primary Reference:

Lee, L-S. and H.M. Stoll and M. Tackitt, "A Continuous-time Optical Neural Network," (in preparation for submission to *Optics Letters*).

A.2.5 Hughes Opto-electronic Resonator Network

Both all-optical and hybrid optical/electronic nonlinear holographic associative memories (NHAMS) have been successfully demonstrated at Hughes Research Laboratories by Yuri Owechko, Gil Dunning, and Bernard Soffer. The NHAM consists of a hologram situated in an optical cavity which provides gain and feedback. The hologram defines the stable modes of the resonator. Multiple angularly-multiplexed optical reference beams are used to record a multiplicity of images in the hologram. When the hologram is subsequently addressed by a partial or distorted version of one of the stored images, the system settles into the stable state "closest" to the input image. The system thus performs as an error-correcting, fully parallel associative memory.

Nonlinear phase conjugate mirrors are used in the all-optical NHAM to threshold, amplify, and back-propagate reference beams which are generated by the input image. Storage and discriminative associative recall of two high-resolution gray-scale images has been demonstrated in the all-optical NHAM. In the hybrid NHAM, feedback and nonlinear processing of the reference beams are provided by liquid crystal light valves (LCLVs), vidicon detectors, and a video image processor. The electronics allows interfacing to a host computer which can program the nonlinearities and associative pathways. Successful operation of the hybrid NHAM was also recently demonstrated. The error-correction properties of the system were evident as the input image could be rotated over a range of 10° with no observable degradation in the associated output image. In these initial proof-of-principle experiments, the stored images contained approximately 10^5 pixels, or "neurons," with each of the input neurons connected to a reference-layer neuron.

Such optical associative memories, with their full parallelism and large number of interconnects, are extendable to high-speed implementations of large multi-layer neural network models. Variations on the hybrid NHAM, including the use of photorefractive crystals as the holographic medium, will permit the realtime updating of interconnection weights, which is necessary for the implementation of neural network learning algorithms such as, for example, backpropagation networks or self-organizing feature maps. Both optics and electronics are used to best advantage in hybrid NHAM systems, in that neurons are interconnected optically for parallelism, but the nonlinear operations are performed electronically for programmability. Based on the sensitivity of photorefractive media and performance of current detectors, within several years a hybrid neural network NHAM should be capable of implementing neural network models containing several million neurons with up to 10^8 to 10^9 interconnects.

Primary References:

Owechko, Y., "Opto-electronic Resonator Neural Networks," *Applied Optics*, Vol. 26, p. 5104, December 1, 1987.

Owechko, Y. and G.J. Dunning, E. Marom, and B.H. Soffer, "Holographic Memory with Nonlinearities in the Correlation Domain," Vol. 26, p. 1900, May 15, 1987.

A.2.6 Rockwell/CalTech Holographic Networks

John Hong of the Rockwell Science Center and Demetri Psaltis and Cheol Hoon Park of CalTech have developed optical implementations of associative memory networks which use quadratic or higher-order interconnections. To realize the required interconnections, the architectures use volume holograms, and, in one system, planar holograms are used to achieve shift-invariant operation. The holographic media are readily available (e.g., photorefractive crystals on $LiNbO_3$, gelatin, thermoplastic holographic plate, etc.). The other components required for the threshold nonlinearity (e.g., liquid crystal light valve) are not readily available. A trainable system which uses a higher-order generalization of the perceptron algorithm has also been developed using an adaptable volume hologram (e.g., photorefractive crystals).

In general, an associative memory maps a set of M N_1 -bit vectors to a set of M N_2 -bit output vectors in a one-to-one correspondence. The most familiar associative memories to date (e.g., the Hopfield memory) are based on a linear discriminant structure where each output bit results from a threshold decision made on a linear combination of the input bits. It is well known that the linear structure limits the capacity of such systems to less than N_1 . The capacity can be significantly increased by using higher-order polynomial expansions of the input. A particular output of a quadratic memory, for example, is computed based on a quadratic expansion of the input bit. The capacity of such higher-order networks is greater than the linear systems at the expense of requiring many more interconnections.

The interconnection-intensive memories just described can be efficiently realized by holographic techniques. In one of the systems, a spatial light modulator is used to first produce every possible pair-wise product of the input bits. Thus, an N_1 -bit one-dimensional input vector is transformed into an $N_1 \times N_1$ -bit two-dimensional representation. A volume hologram which has $N_1 \times N_1 \times N_1$ degrees of freedom is used to interconnect the light emanating from these points to the N_1 output detectors. The unique feature of many of these optical systems is that the realization of the interconnections is not confined to the plane, as would be the case in electronic systems. The interconnections are implemented in a volume which offers more degrees of freedom in a more compact way.

Another system has an output response that is invariant with respect to shifts of the input vector. Here, planar-Fourier-transform holograms are used in a multi-channel version of the classical Van der Lugt correlator. In that arrangement, the input vector is correlated simultaneously against all of the memory vectors, and a spatial light modulator, such as a liquid crystal light valve whose output amplitude is the square of its input, is placed at the correlation plane so that each cross correlation is squared. The complete correlations are then used to read out a second, identical hologram whose outputs are properly summed on an output detector array. Since the input-output relationship is basically a pointwise nonlinear composition of correlation and convolution operations, the overall response is shift-invariant. Moreover, it was shown that the nonlinear operation in the correlation domain is crucial to the operation of the system. The system is equivalent to a quadratic associative memory which has been set up to store every shifted version of its memory vectors. At present, the critical components for the implementation of the systems described thus far are being evaluated, and actual implementations are under way.

Primary Reference:

Psaltis, D. and J. Hong, "Shift-invariant Optical Associative Memories," *Optical Engineering*, 26(1), p. 10, Jan., 1987.

A.2.7 University of Pennsylvania Optical Networks

Nabil Farhat and graduate student Zon Yin Shae at the University of Pennsylvania have built a bimodal opto-electronic neural network that can be used in two distinct modes, depending on the noise level (temperature) of the network. At finite noise levels, the network is stochastic and can be used as a Boltzmann machine with extremely fast annealing time (about 30 times the neuron time constant). Fast annealing is achieved with a novel noisy thresholding scheme that utilizes optical noise (snow pattern on a television screen), and any annealing profile can be implemented.

In this mode, the network is useful for stochastic learning and self-organization, and for the solution of combinatorial optimization problems. Connectivity weights, synaptic modifications, and learning are realized using a non-volatile magneto-optic spatial light modulator (MOSLM), but other non-volatile SLMs can be used. Having learned the entities presented to it in a supervised manner through a process of adaptive synaptic weight modification, the network is "frozen" by reducing the noise level to a minimum. It then acts then as an associative memory of the entities learned.

Distinctive features of the network include:

- Stochastic learning with binary weights employing multiple time-scale annealing profiles and dead zone limiting at the neurons, which permits the use of binary SLMs.
- Full programmability in the sense that the network can be partitioned with the aid of an external computer controller into any number of layers with any desired pattern of communication among neurons in different layers and within the same layer. The present prototype consists of 24 binary neurons with 48x48 fully programmable interconnections as limited by the available 48x48 MOSLM and its drive circuitry.
- Potential for being compacted into a small module consisting of a non-volatile SLM sandwiched between a pair of nonlinear reflector arrays employing internal feedback and external ambient light illumination. Such chips are efficient (as they utilize available ambient light) and are clusterable to form large neural networks of 10^3 – 10^6 neurons by exploiting their inherent optical interconnectability. The lack of capacitive or other loading of optical interconnects means the time response of the entire cluster is that of the individual modules and suggests that massive neural networks might be feasible with this clustering approach. A neural network cluster of nine modules is presently in the prototyping stage.

Primary Reference:

Farhat, N.H., "Opto-electronic Analogs of Self-programming Neural Nets: Architectures and Methodologies for Implementing Fast Stochastic Learning by Simulated Annealing," *Applied Optics*, Vol. 26, pp. 5093-5103, Dec., 1987.

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified		1b. RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.	
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE			
4. PERFORMING ORGANIZATION REPORT NUMBER(S) Technical Report 840		5. MONITORING ORGANIZATION REPORT NUMBER(S) ESD-TR-88-311	
6a. NAME OF PERFORMING ORGANIZATION Lincoln Laboratory, MIT	6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Electronic Systems Division	
6c. ADDRESS (City, State, and Zip Code) P.O. Box 73 Lexington, MA 02173-0073		7b. ADDRESS (City, State, and Zip Code) Hanscom AFB, MA 01731	
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Air Force Systems Command, USAF	Bb. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8c. ADDRESS (City, State, and Zip Code) Defense Advanced Research Projects Agency 1400 Wilson Boulevard Arlington, VA 22209		10. SOURCE OF FUNDING NUMBERS	
		PROGRAM ELEMENT NO. DARPA/TTO	PROJECT NO. 320
		TASK NO.	WORK UNIT ACCESSION NO.
11. TITLE (Include Security Classification) DARPA Neural Network Study Final Report, (October 1987 — February 1988)			
12. PERSONAL AUTHOR(S)			
13a. TYPE OF REPORT Technical Report	13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Year, Month, Day) 22 March 1989	15. PAGE COUNT 646
16. SUPPLEMENTARY NOTATION None			
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	
19. ABSTRACT (Continue on reverse if necessary and identify by block number)			
<p>The Neural Network Study, sponsored by the Tactical Technology Office of the U.S. Defense Advanced Research Projects Agency (DARPA/TTO), was conducted under the auspices of the Massachusetts Institute of Technology's Lincoln Laboratory (MIT/LL) from October, 1987 through February, 1988 with government, industry, and academic participants.</p> <p>The goals of the study, all of which were achieved, were</p> <ul style="list-style-type: none"> ● To identify potential applications for neural networks in Department of Defense (DoD) systems, ● To determine the current neural network technology base, ● To identify technology requirements, and ● To identify a DoD program plan for the next five years. 			
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT <input type="checkbox"/> UNCLASSIFIED/UNLIMITED <input checked="" type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS		21. ABSTRACT SECURITY CLASSIFICATION	
22a. NAME OF RESPONSIBLE INDIVIDUAL Lt. Col. Hugh L. Southall, USAF		22b. TELEPHONE (Include Area Code) (617) 981-2330	22c. OFFICE SYMBOL ESD/TML