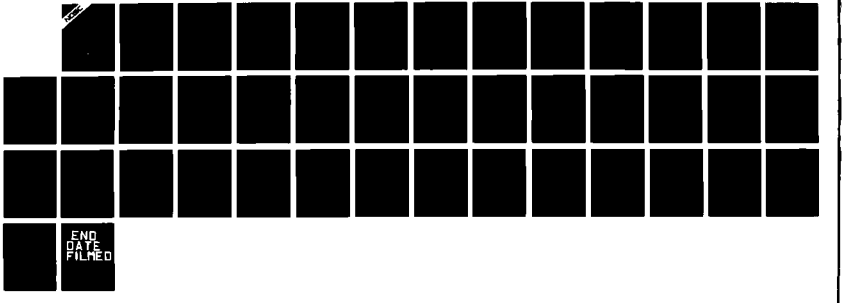
AD-A184 759

NAVAL OCEAN SYSTEMS CENTER, SAN DIEGO, CA
A SURVEY OF ARTIFICIAL NEURAL SYSTEMS
BY: PK SIMPSON, UNISYS

1 OF 1
NOSC TD 1106
UNCLASSIFIED
JUN 1987



END
DATE
FILMED

NAVAL OCEAN SYSTEMS CENTER San Diego, California 92152-5000

NOSC TD 1106

Technical Document 1106
June 1987

A Survey of Artificial Neural Systems

P. K. Simpson
UNISYS

NAVAL OCEAN SYSTEMS CENTER

San Diego, California 92152-5000

E. G. SCHWEIZER, CAPT, USN
Commander

R. M. HILLYER
Technical Director

ADMINISTRATIVE INFORMATION

The research cited in this report was compiled by the UNISYS Company of San Diego, CA, under the direction of Stephen Luse and Christine Dean, Human Factors and Speech Technology Branch, Code 441, Naval Ocean Systems Center, San Diego, CA 92152-5000. Funding was provided by the Naval Ocean Systems Center.

Released by
C.M. Dean, Head
Human Factors and Speech
Technology Branch

Under authority of
W.T. Rasmussen, Head
Advanced C^2 Technologies
Division

PK

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 is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) NOSC TD 1106
6a. NAME OF PERFORMING ORGANIZATION UNISYS 6b. OFFICE SYMBOL (if applicable)	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) San Diego Systems Engineering Center 4455 Morena Boulevard San Diego, CA 92117	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Naval Ocean Systems Center 8b. OFFICE SYMBOL (if applicable) NOSC Code 441	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER 87WR00124
8c. ADDRESS (City, State and ZIP Code) San Diego, CA 92152-5000	10. SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO: 63223L; PROJECT NO: SDI; TASK NO: ; AGENCY ACCESSION NO: DN307 445
11. TITLE (include Security Classification) Survey of Artificial Neural Systems	
12. PERSONAL AUTHOR(S) P.K. Simpson	
13a. TYPE OF REPORT Final 13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Year, Month, Day) June 1987 15. PAGE COUNT
16. SUPPLEMENTARY NOTATION	
17. COSATI CODES FIELD / GROUP / SUB-GROUP	18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) neurons; neural networks; information processing; brain speed

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This paper is a survey of the field of Artificial Neural Systems (ANS). Included is a history of ANS, examples of ANS models and areas where the technology has been applied.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT ☐ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ OTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Christine Dean	22b. TELEPHONE (include Area Code) 619-225-7372 22c. OFFICE SYMBOL Code 441

A Survey of Artificial Neural Systems

Patrick K. Simpson

Unisys
San Diego Systems Engineering Center
4455 Morena Boulevard
San Diego, CA 92117
619/483-0900

Contracted by
Naval Ocean Systems Center
Code 441 - Speech Technology Group
San Diego, CA 92152
619/225-7372

Abstract

This paper is a survey of the field of Artificial Neural Systems (ANSs). ANSs have a large number of highly interconnected processing elements that demonstrate the ability to learn and generalize from presented patterns. ANSs represent a possible solution to previously difficult problems in areas such as speech processing and natural language understanding. This paper presents a brief history of ANSs, examples of ANS models and areas where the technology has been applied. Also discussed is the connection between Artificial Intelligence (AI) and ANS, computer architectures that are evolving from this field, and ANS algorithms.

Table of Contents

1. Introduction

Speech processing, image processing and robotics are all forms of pattern matching. In each area an input is received and matched to a corresponding output. Humans are easily able to perform these pattern matching tasks, computers, however, are not. Computers, on the other hand, are faster than humans at algorithmic computational tasks. The contrast between the processing abilities of computers and humans arises because each processes its information differently.

Most computers process information with a single complex central processing unit (CPU). The human brain processes information using a large number of simple processing elements called neurons. To increase the pattern matching ability of computers requires a different approach to processing information from the current single processor architecture. Artificial neural systems (ANSs) are neurally inspired mathematical models that use a large number of simple processing elements (PEs). ANSs approach the pattern matching problem using the same processing style the brain uses. PEs are organized into layers where each PE in one layer has a weighted connection to each PE in the next layer. This organization of PEs and weighted connections creates an ANS. An ANS learns patterns by adjusting the strengths (weights) of the connections between PEs. Through these adjustments an ANS exhibits properties of generalization and classification similar to humans.

This is a survey of the field of ANS. Included is an explanation of the biological inspirations and mathematical foundations of ANS. The history of ANS and applications using ANS are presented as well as a discussion of how ANS relates to the field of artificial intelligence. At the end of this survey is an overview of models and architectures used in ANS.

2. The Neuron and Neural Networks

2.1. An Explanation of the Neuron

The basic building block of the nervous system is the *neuron*, the cell that handles intercommunication of information among the various parts of the body. A neuron consists of a cell body called a *soma* and an *axon* or *nerve fiber* that connects the cells to one another (see figure 1). Junctions between neurons occur either on the cell body or on spinelike extensions of the cell body called *dendrites*. The junctions are called *synapses*. Nerve fibers and dendrites can be treated like insulated conductors for transmitting electrical signals to the neuron [Lindsay77].

A threshold unit collects inputs and produces output only if the sum of the inputs exceeds an internal threshold value. The neuron, in its simplest form, can be considered a threshold unit. As a threshold unit, the neuron collects signals at its synapses and sums them together using its internal *summer*. If the collected signal strength is great enough to exceed the threshold, a signal is sent out from the neuron to the axons.

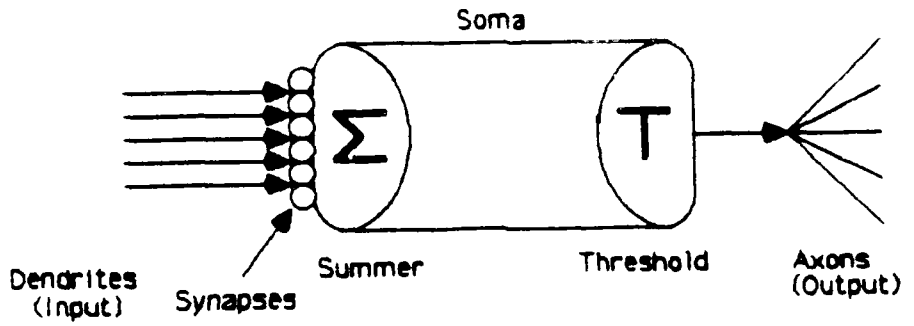


Figure 1: The neuron. *This figure shows the neuron body (soma) and the components of importance to ANS. The synapses are where signals are collected from the dendrites. The summer is where the signals are summed. The threshold is the internal value that must be exceeded for output. The axon is where any output signals are sent.*

2.2. Neurally Inspired and Mathematically Supported Models

ANS are neurally inspired models of the mind [McClelland86, Myers86, Rumelhart86a, Rumelhart86b]. ANSs are not attempts at duplicating the mechanisms of the mind, only attempts at duplicating the functionality of the mind. Drawing upon an analogy from D. Rumelhart, a leader in the field of ANS from the University of California, San Diego (UCSD), the brain is the computer hardware (mechanisms) and ANS is the computer software (functionality). Extending the analogy, learning can be considered programming the mind. The objective behind ANS is capturing the functionality of the mind.

Constantly changing systems are called dynamical systems. Dynamical systems are described by energy functions and probability distributions. ANSs are supported by the mathematics of such systems. A more precise definition of ANSs is dynamical systems with adaptive or selectable energy functions that can carry out useful information processing by means of initial response to initial or continuous input [Myers86]. Rephrased, ANSs are directed graphs that are able to change when provided input.

2.3. Constraints and Assumptions About How the Brain Processes Information

Neural models are based on how the mind processes information information [Amari71, Hecht-Nielsen86a, McClelland86]. The performance of the mind and the performance of the digital computer are compared in an attempt to understand how the mind is so adaptable, resilient, and powerful. ANS models incorporate information about brain processing during modelling. The information gathered has shown the

presence of physical brain constraints and has led to assumptions about brain processing. The following sections discuss these assumptions and constraints and contrast them with the digital computer.

2.3.1. Brain Speed Versus Computer Speed

Cycle time is the time taken to process a single piece of information from input to output. The cycle time of the most advanced computers is 1 nanosecond, corresponding to one clock cycle for the CPU. The average cycle time for a neuron in the brain is 2 milliseconds [Cottrell84]. The difference in speed is 5×10^5, the computer is five-hundred thousand times faster.

2.3.2. Parallel Order Versus Serial Order and the 100 Step Program

The most advanced computers are able to process information one million times faster than the brain, yet in some respects the brain is superior. The difference between the two machines is the processing order. The brain processes its information in parallel, the computer processes its information in serial. There is a constraint that can be extended from this information called the 100 step program constraint [Feldman82]: If the mind reacts between 1/5 and 1 second to a given stimulus (i.e. answering a true-false question) and the cycle time of a neuron averages 2 milliseconds, then in the best case in 100 cycle times of a neuron a decision is reached. To make a program that processes information like the brain, that program should not exceed 100 steps. In contrast to large software programs operating in serial on conventional computers, the mind operates with a massive number of small programs that execute in parallel.

2.3.3. Number and Complexity of Neurons

The number of neurons in the brain is approximately 10^{11} with about 10^3 to 10^4 connections between each neuron. An ANS should not simulate any more than 10^{11} neurons. Although 10^{11} is admittedly large, the size is finite and constrained. A neural model will require the ability to handle large numbers of processing units. In addition to the large numbers of neurons, studies have also found that the neuron is not a simple threshold unit [Levy82]. The neuron is actually a complex computing device. Recent studies have shown that all the computing does not take place solely inside the soma; computations also occur outside the neuron body in the dendrites and at the synapses. These two pieces of information remind ANS technologists that the brain is a complex and large device and that models will eventually have to represent such size and complexity.

2.3.4. Connections Hold the Knowledge

The number of connections between neurons in the brain is relatively fixed. Very few new pathways are formed in adult brains and these are assumed to be for long-term memory [Feldman82]. Because of the lack of new connections and the time it takes for new connections to form, new knowledge is believed to be captured by changing the strengths of the connections [Cottrell84, McClelland86]. An ANS will not need to add and remove connections to simulate the processing of the brain, only change the strength of the connections.

2.3.5. Fault-tolerant Brain Versus Fault-intolerant Computers

The brain is very resistant to noise and rather robust in the sense that damage (faults) to individual neurons does not degrade the overall performance of the brain [Cottrell84]. Because of this graceful degradation, the brain can be said to be fault-tolerant. The concept of fault-tolerance supports the theory that the brain carries a distributed representation of the world in which no one neuron carries a specific thought or idea. Thoughts and ideas are spread out through many neurons and interconnections. Most computers are fault-intolerant. Each location in computer memory holds a specific piece of information. If that memory location is corrupted then the knowledge is lost, creating a fault in the system.

2.3.6. No Executive Control in the Brain

The brain does not have any specific area with executive control [Cottrell84]. Each neuron computes an output based solely on its inputs. A neuron cannot access information held by other neurons unless it is directly connected. A neuron cannot look around to see what the other neurons are doing. When designing an ANS only the inputs to a neuron need to be considered. There is a sharp contrast in the comparison of the control in a computer versus the control in the mind. A computer uses the CPU and the mind uses distributed control throughout the brain.

3. History of ANS

ANS research began in the early 1940s. The field is young and many of the people that were instrumental in its inception are still very active in the field today. The following sections discuss the people and accomplishments of ANS from 1943 to present.

3.1. McCulloch and Pitts (1943)

McCulloch and Pitts made the first mathematical model of an ANS [Hecht-Nielsen86a, Rumelhart86a]. This model showed that an ANS could compute, a theorized but previously unproven concept. Although the model was able to compute, it could not learn.

3.2. Hebb (1949)

D. Hebb brought learning to ANS technology [Hecht-Nielsen86a, McClelland86, Rumelhart86a]. Hebb's book **Organization of Behavior**, published in 1949, describes a system of correlation learning at the synapses of the neuron. From Hebb's observations of how learning occurred in the neuron, he developed a learning rule for ANS, the Hebbian Learning Rule. The Hebb Learning Law states "If neuron A repeatedly contributes to the firing of neuron B, then A's efficiency in firing B increases." Since the formation of Hebb's Law a restatement of the law has emerged that says " the strength of the synaptic connections are changed in proportion to the difference between the target and actual output of a neuron." [Jorgensen86] To explain, if the neuron has a positive output (actual) and it is expected to be negative (target), negatively reinforce the synaptic connections to the neuron. If the neuron has a negative actual output and a positive target output, positively reinforce the synaptic connections. If the actual and target outputs are the same, leave the synaptic connections unchanged.

3.3. Lashley (1950)

Lashley's studies of the mind led to his insistence that the mind has a distributed knowledge representation [McClelland86]. Lashley's idea was that knowledge is not locally stored but rather it is stored in a distributed manner. Rephrasing this, there are no special cells for special memories; rather, many cells carry a portion of the memory.

3.4. Edmonds and Minsky (1951)

D. Edmonds and M. Minsky were the first to build a physical ANS [Bernstein81, McClelland86, Rumelhart86a]. Their model, built at Harvard in the summer of 1951, was constructed of tubes, motors and clutches. The clutches were adjusted in accordance to the Hebbian Learning Rule to store the connection strengths. The machine was able to store as many as 40 patterns of 40 binary digits, but was too inflexible for further work.

3.5. Rosenblatt (1957)

F. Rosenblatt became a prominent ANS researcher with his creation of a neural model called the *perceptron* [Hecht-Nielsen86a, Larson86, McClelland86, Rumelhart86a]. The perceptron showed remarkable promise as a computing device, being able to learn patterns and generalize from patterns learned. Rosenblatt studied his model with mathematical analysis and digital computer simulations. The percpetron brought many researchers to the field of ANS.

3.6. Minsky and Papert (1969)

M. Minsky and S. Papert were responsible for the demise of the perceptron as well as much of the ANS research during late 1960s [Hecht-Nielsen86a, Larson86, McClelland86, Rumelhart86a]. Minsky and Papert were irritated at Rosenblatt for over claiming the perceptron's ability. In their book **Perceptrons**, published in 1969, the two researchers showed that the perceptron was an inadequate model because it could not represent the basic exclusive-or (XOR) function. Minsky and Papert were so convincing that most ANS research at the time was halted.

3.7. Grossberg (1968)

Despite the scorching by Minsky and Papert, ANS research did continue on a small scale. S. Grossberg studied neurally inspired mechanisms in both perception, memory, and later vision [Grossberg68, Hecht-Nielsen86a, McClelland86]. Grossberg's research has focused on the mind, using an ANS to model his ideas. Grossberg's mathematical analysis of properties of ANS models has led to many insights that include neurally inspired models of perception and memory. Grossberg's research has recently been focused on vision. He has just finished a study that used an ANS that mimics human eye movements [Grossberg85].

3.8. Willshaw (1969)

As a member of the research group at Edinburgh University under Longuet-Higgins, D. Willshaw made important contributions toward understanding memory [McClelland86]. Willshaw did mathematical analysis of distributed memory models and found properties associated with various modeling schemes. Later in collaboration with Longuet-Higgins, Willshaw did work with holophones [Willshaw69]. A holophone is a man-made representation of memory that is useful in the analysis of memory systems.

3.9. Amari, Anderson, and Kohonen (1971)

Three researchers who did significant work in 1971 are Amari, Anderson, and Kohonen. Amari's work was with Boolean ANS theory, an ANS that contains only Boolean values [Amari71]. Anderson's work was with linear associative memory, a memory that is completely distributed [McClelland86]. Kohonen's research concerns self-organizing associative memory, studying how the mind organizes the information it stores [Kohonen84]

3.10. Rumelhart and McClelland (1977)

During the late 1970s ANS technology became prominent in the field of cognitive psychology, using ANSs for cognitive models. Two cognitive psychologists who initiated this movement are D. Rumelhart of UCSD and J. McClelland of Carnegie-

Mellon University (CMU). McClelland and Rumelhart were inspired by the HEARSAY speech understanding system at Stanford University [Hecht-Nielsen86a, McClelland86]. In their efforts toward building a cognitive model for speech understanding they rediscovered ANSs. Rumelhart and McClelland's models are called parallel distributed processing (PDP) models [McClelland85, Rumelhart86b]. Many ANS learning paradigms have been studied using PDP, namely competitive learning [Rumelhart86a], Boltzmann Machines [Hinton86b, Rumelhart86d], and most recently Error Propagation [Rumelhart86c].

3.11. Hecht-Nielsen (1977)

Commercial research is also being conducted at TRW. This research was headed by R. Hecht-Nielsen [Hecht-Nielsen86a, Hecht-Nielsen86b]. This work focused on the application aspects rather than the research aspects of ANS technology. While working at TRW, Hecht-Nielsen developed two neurocomputers, the Mark III which is commercially available [TRW86] and the DARPA-financed Mark IV. In 1986 Hecht-Nielsen left TRW and started his own neurocomputer company. His company has developed a neurocomputer called the ANZA that fits on a card and plugs into an IBM PC AT [HNC86].

3.12. Hopfield (1982)

The recent resurgence of interest in ANS technology is mostly attributed to J. Hopfield of the California Institute of Technology (CalTech) [Hecht-Nielsen86b]. Hopfield delivered a paper to the National Academy of Science in 1982 that proved that an ANS of interconnected processing elements would seek an energy minima [Hopfield82]. This paper showed that ANSs have emergent collective computational abilities; restated, as ANSs compute emerging properties are found. The emergent collective properties that are found using an ANS appealed to a wide range of disciplines, most notably physics, computer science, cognitive psychology and neuroscience [Larson86, Myers86]. Since publishing the paper, Hopfield has continued to study the neurobiological aspects of ANS [Hopfield84].

3.13. Cooper and Elbaum (1983)

Former Brown University Physicists L. Cooper, a Nobel laureate, and C. Elbaum started their own company called Nestor Inc. in 1983 [Larson86, Nestor86]. Cooper and Elbaum, like Hecht-Nielsen, are also interested in the commercial applications of ANS. Nestor's commercial projects include hand-written computer input systems [Nestor86, Reilly82], speech recognition [Epstein86], and 3-dimensional graphics [Rimey86].

3.14. Kosko (1985)

B. Kosko of Verac Corporation has done research with fuzzy logic that has overlapped into ANS [Chester86, Kosko86]. Fuzzy logic is the representation of unclear and non-specific information (fuzzy data). Kosko has worked out a way to integrate ANS and fuzzy logic using a system called fuzzy cognitive maps. One application using this melding of fuzzy logic and ANS has been in radar image processing.

4. The AI/ANS Connection

The work being done in artificial intelligence (AI) and ANS overlaps. In some instances, AI and ANS have the same goals; in others they do not. In this section a discussion of where the two fields overlap and diverge is presented as well as a discussion of how the two technologies could be melded.

ANS technology is aimed at developing human-made systems that can perform the same type of information processing that the brain performs. ANS developments include real-time performance in pattern recognition (speech recognition), knowledge processing given inexact and incomplete knowledge (image processing), and precise control in multiple constraint environments (robotics) [Hecht-Nielsen86b, Larson86]. ANS technology is so different from conventional computer technologies that it is necessary to create a new architecture to support it. ANS processors (neurocomputers) are the new architectures that have been produced to accomodate ANS technology.

There are areas in AI and ANS that have the same goals. Over the past 30 years, the AI community has studied the areas where ANS technology is currently being applied. The areas of speech recognition, image processing, and robotics have been assessed to be difficult areas in AI that yield slow progress [Hecht-Nielsen86b]. Areas where AI machines and conventional computers are superior to ANS are algorithmic, logic, and symbolic processing.

ANS technology is aimed toward the difficult areas of AI. AI computers are not well suited for adapting and generalizing, but this is an area where ANS performs well. Areas such as expert systems and symbolic computing are better suited for AI LISP-oriented machines. It is not the intent of ANS technology to replace AI technology. These two technologies are able to profit the greatest by melding themselves into one machine. By installing an ANS processor into an AI machine, and having the AI machine call upon the ANS processor when needed, a mutual environment with improved performance is created. The ANS processor is used where it is best suited, as a specialized support subsystem for an AI computing system.

5 ANS Applications

ANS has been applied in a variety of areas and many more are yet to be discovered [Port87]. ANS is good at specific tasks. One task ANS is able to perform is as an associative memory. An associative memory processes all possible outputs for

a given input simultaneously, eventually finding the proper output in constant time [Kohonen84]. Another task ANS is able to perform is creating generalized representations of presented input [Epstein86, Larson86]. Extending this idea, ANS is able to process information given only a portion of the input. This is a useful feature in that ANS does not need to be supplied all the input information to get the proper output [Larson86, McClelland86]. Similarly, if partially incorrect or unclear (fuzzy) information is given to an ANS it will make a "best guess," processing an output from the given input based upon the generalized internal representation [Chester86, Kosko86]. Yet another task where ANS is competent is multiple simultaneous constraint problems. ANS is able to process many inputs simultaneously and produce an output based on those inputs. Because of the ability to process a large amount of information simultaneously, robotics is an area that is well suited for ANS [Myers86, Nestor86].

The tasks that are performed well using ANS technology are being applied in many areas. In the following three sections, examples of ANS applications are presented. The first section discusses the NETtalk text-to-speech convertor built by Sejnowski and Rosenberg, the second presents a cognitive psychology experiment in natural language processing performed by Rumelhart and McClelland, and the final section briefly mentions other areas where ANS has been applied.

5.1. Sejnowski/Rosenberg's NETtalk

T. Sejnowski of Johns Hopkins University (JHU) and C. Rosenberg of Princeton University (PU) constructed an ANS application that did text-to-speech conversion [Sejnowski86]. A phoneme is a symbolic representation of a single syllable sound utterance. Sejnowski's ANS maps text (input) to phonemes (output). The phonemes that are output from the ANS are in turn passed to a phoneme-to-speech synthesizing device called the DECtalk. The ANS learned to read text aloud by successively being presented a window of seven letters from a text (corpus) and using the middle letter as the target that the single phoneme output would represent. By mapping seven letters to a single phoneme, the ANS was incorporating context into the conversion process. The middle letter of the seven is the letter that the phoneme represents and the three preceding and three following letters are context for the letter being represented. After the seven letters are presented, a one-letter shift of the text is done and the process is repeated. This process is continued for the whole corpus being presented. The ANS model used the Rumelhart/Williams Error Propagation algorithm (discussed in detail later) for doing the mapping from input to output.

The ANS begins in an untrained state with random connection strengths. After a short training period, the output from the DECtalk begins to make a continuous and eery babbling. At this stage of the training, all speech is connected and only one verbal sound is heard. Separation of sounds occurs and more than one verbal sound is heard as the ANS continues to learn. At this stage, the output begins to sound like an

infant. As the training continues further, the output from the DECtalk begins to sound like a young child talking and words are clearly distinguishable.

The NETtalk ANS has captured a large number of the rules necessary for speech synthesis, for example, properly pronouncing the "a" in both "say" and "ran." The same results are possible from commercial text-to-speech systems, but it has taken years of development and study to learn the same rules that the NETtalk ANS learned autonomously overnight. The development time for the NETtalk ANS was only three months, significantly less time than its commercial counterparts.

5.2. Rumelhart/McClelland's Natural Language Processing

Two noted cognitive scientists, D. Rumelhart of UCSD and J. McClelland of CMU, have created an ANS that learns the past tense of English verbs [Rumelhart86d]. The objective of the study was to determine if an ANS model acquired the rules for forming the past tense of verbs the same as children. An ANS was designed that took as input a phonemic representation of a root verb and gave as output a phonemic representation of the past tense of that root verb. The ANS operated as follows: (1) The root form of the verb was presented as input, (2) the past tense of the root form was the targeted output, and (3) errors between the root and past tense of the verb were corrected using the Boltzmann Machine Learning Rule (discussed in detail later).

A child progresses through three stages when acquiring the rules for forming the past tense of English verbs [Brown73, Ervin64]: In stage one there is no evidence of any rules being formed by the child. Stage two shows an implicit knowledge of linguistic rules; an ability to apply the rules to both nonsense and real words is noticed as well as over-regularization of verbs, for example, regularizing the verb "come" to "comed." In the final stage both the regular and irregular forms of the verbs exist; the child has learned both the rules and the exceptions.

Tests were conducted at various stages throughout the training of the ANS. With a limited amount of training, the ANS exhibited stage one results, showing no rule formations. As the training continued, stage two results were seen. At this point in the training, the ANS exhibited the same mistakes that children exhibited, regularizing all forms of the root verb. Finally after an abundance of training, stage three results were achieved. The ANS had learned both the rules for forming the past tense of root verbs and it had learned the exceptions to the rule.

Another result of the training was that the ANS was able to respond to verbs it had never seen before. This result showed that an ANS was able to abstract from what it had learned, applying its knowledge to unknown root verb forms and forming the proper past tense. In summary, this study has shown that an ANS is able to learn and generalize from information given, as well as apply itself to information previously unknown. This study shows promise in real-world situations that involve inexact data

and where a "best guess" is sufficient. Current computer software does not handle inexact data well; it is fault-intolerant and requires specific input. Alternatively, ANS has the intrinsic property of being fault-tolerant and able to handle unspecific data gracefully.

5.3. Other Application Areas

Natural language processing (NLP) as an ANS application has been explored by G. Cottrell of UCSD and S. Small of the University of Rochester (UR) [Cottrell84]. Cottrell and Small built an ANS that did word sense discrimination, teaching an ANS to determine which sense of a word is correct from the context. An example is that the ANS can understand the difference in meaning that the word "threw" conveys in the sentences "Bob threw a fight" and "Bob threw a ball." Other work in NLP that involves ANS has been done by M. Fanty at UR [Fanty85]. Fanty developed an algorithm that constructs an ANS for a context-free grammar and uses the created ANS as a parser.

The ANS approach has been well received in the area of image processing. One example is the Mingolla/Grossberg Vision Processing Network, which has demonstrated that template-driven image segmentation and shift/scale/rotation invariant image pattern recognition are possible using an ANS [Hecht-Nielsen86b].

In another image processing application using ANS, N. Farhat of the University of Pennsylvania (UPenn) has trained an ANS to discern radar images of various aircraft [Larson86]. Results of the application have shown recognition of a bomber with only 20 percent of the image supplied.

Nestor Inc. has developed an application of ANS that accepts writing on a digitized pad as computer input [Nestor86]. The ANS learns the idiosyncracies of anyone's handwriting, allowing a more direct input to the computer. The company's focus is on an ANS that takes kanji (Japanese lettering) as input and converting it to a computer character form, thus eliminating the difficult task of computer entry.

Current applications of ANS in speech research has been toward discovering a method for speaker-independent recognition. J. Elman and D. Zipser have used ANS in an attempt to discover the hidden features in speech that allows humans to distinguish words [Elman87]. Another speech-related project is to use spatiotemporal pattern matching to achieve speaker-independent recognition [Hecht-Nielsen86c].

6. ANS Models

There are many different ANS models. This paper will present three of the prevalent models in ANS technology. The first model is the Hebb/Hopfield model [Hopfield82, Jorgensen86], the second is the Boltzmann Machine model [Hinton86b, Rumelhart86d], and the last is the Error Propagation model [Hecht-Nielsen86a, Rumelhart86c]. Each model is successively more complex and more successful. Each

model has processing elements (PEs) with adjustable strength connections from other PEs. Each successive model varies in how the connection strengths are adjusted and how the connections and PEs are arranged into an ANS.

The following section explains the PE and its similarities to the neuron. Following the explanation of the PE are three sections dedicated to explaining each of the forementioned ANS models.

6.1. The Processing Element

The PE (see figure 2) consists of weighted input connections ($w_0,...,w_N$), a summation function, a threshold function, and an output value [Hecht-Nielsen86a, McClelland86, Rumelhart86b].

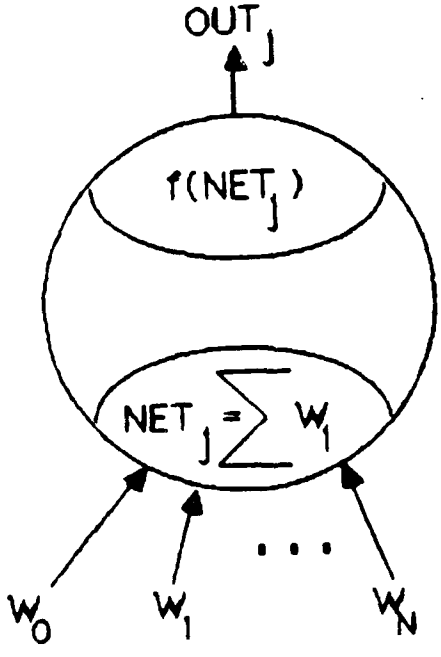


Figure 2: An ANS PE. *The inputs are the weighted connections from the N elements ($w_0,...,w_N$) of the layer below to the j^{th} PE. The weights are added together using a summation function that produces the value NET_j. The output of the summation function is passed through a threshold function $f(NET_j)$. The output of the threshold function is the net value of the PE OUT_j.*

Each of the components of this PE corresponds to a component of the neuron discussed in section 2 [Jorgensen86]. The correspondence is summarized in table 1.

Table 1: Comparison of components of a PE and a neuron.

Correspondence Between Neuron and PE	
Neuron	PE
Synapses	Weights
Summer	Summation Function
Threshold	Threshold Function
Axon	Net Output

By combining many of these PEs, we form an ANS. The ANS is constructed in layers. The inputs from the environment (external to the ANS) enter the input layer of the ANS. The outputs enter the environment from the output layer of the ANS. Any layers that exist between the input and output layers are called hidden layers. Hidden layers are not accessible from the environment; they are only accessible by the ANS. The connections are made from input toward output, but not from output to input. Using this definition, an ANS can be considered a hierarchical directed graph that only allows the flow of information from parent (input) to child (output). Connections in an ANS are made from every PE on the parent level to every PE on its child level, creating a completely interconnected ANS. Information is stored (learning) by adjusting the weights (connections strengths) between PEs. Figure 3 shows a completely interconnected ANS that has 5 PEs on the input layer ($in_0,...,in_4$) and 5 PEs on the output layer ($out_0,...,out_4$).

6.2. The Hebb/Hopfield Model

ANSs are dynamical (constantly changing) systems. Energy functions and probability relationships are used to mathematically describe and model ANSs because of their dynamic nature. An ANS is a mathematical model in an N-dimension energy space, where N is the number of connections in the most interconnected PE. The Hebb and Hopfield models describe an energy space that seeks a local energy minima from the point of entry [Hopfield82, Jorgensen86]. Once the system is given a specific input pattern (entry point into the energy space), the system will compute the output pattern (seek its local minima).

An ANS must be able to adjust itself to seek the proper energy minima upon entry. The adjustments of the weights (connection strengths) in an ANS create the proper energy minima for a given input. The Hebb and Hopfield models are different in respect to the methodology used to adjust the weights. The Hebb model uses an algorithm that increases the strength of connections between PEs that are both positive, decreases the strength of PEs that are both negative, and leaves the connections

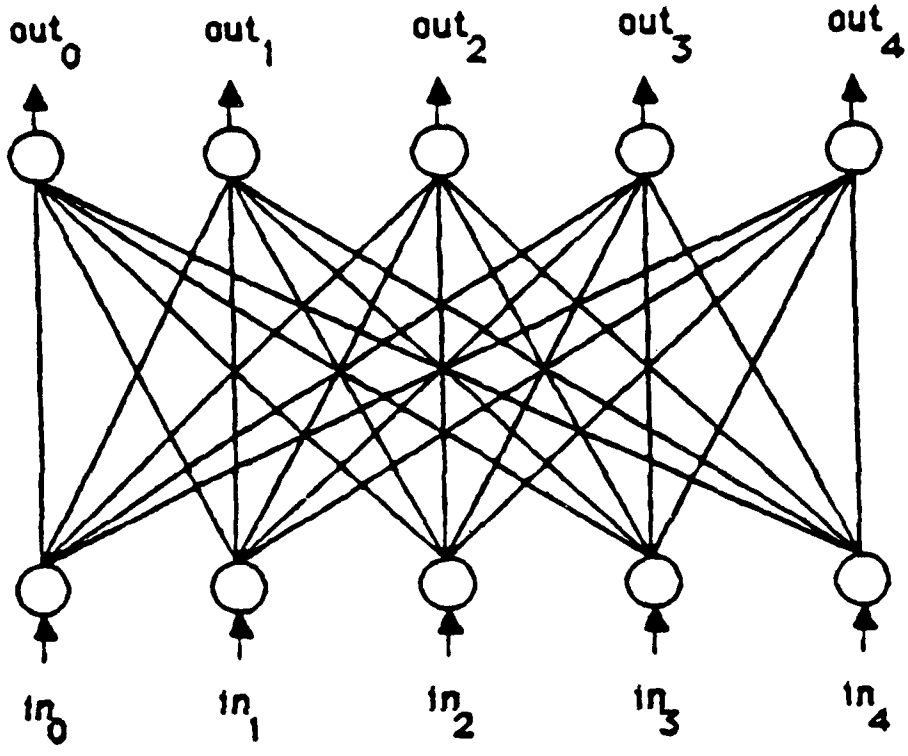


Figure 3: A two layer ANS. *A two-layer ANS with 5 PEs on the input layer (in_0,...in_4) and 5 PEs on the output layer (out_0,...out_4). Each PE on the input layer is connected to a PE from the output layer, creating a completely interconnected ANS.*

between positive/negative PEs unchanged. The Hopfield model uses an algorithm tha assigns a positive value to connections between two PEs that are either both positive or negative, and assigns a negative value to PEs that have mismatched (positive/negative) values. With the exception of how the weights are adjusted, the Hebb and Hopfield models are the same.

One use of the Hebb/Hopfield models is as an associative memory (discussed in section 3). An associative memory is able to reconstruct a complete pattern from only a portion [Kohonen84]. An example of the actions of such an ANS is to store (adjust the weights in the ANS) binary value "101." When the ANS is presented with the incomplete input "1?1" (where the ? means unknown) it will reconstruct the missing information and output "101." The Hebb and Hopfield models that accomplish this are described in appendix 1.

6.3. The Boltzmann Machine Model

The Boltzmann Machine model is an ANS that uses the Boltzmann probability distribution function to adjust connection strengths. The Boltzmann distribution function is a ratio of probabilities:

$$\frac{P_\alpha}{P_\beta} = e^{\frac{-(E_\alpha - E_\beta)}{T}}$$

where P_α is the probability of being in energy state α, P_β is the probability of being in energy state β, E_α is the energy of state α, E_β is the energy of state β, and T is the temperature of the energy system [Jorgensen86, Rumelhart86d].

The probability of being in energy state α or energy state β is equal to 1 (i.e. $P_\alpha + P_\beta = 1$). In the ANS form of the distribution equation, the associated energy of state β is assigned to the variable Θ. Solving for the probability of being in state α results in the following:

$$P_\alpha = \frac{1}{1 + e^{\frac{-(E_\alpha - \Theta)}{T}}}$$

The ANS equivalent for the probability of being in state α of the above equation is the output value desired of the j^{th} PE, where j is one of the N output PEs in the output layer of a two-layer ANS. This value is mathematically referred to as out_j, the j^{th} element of the desired output vector out. The ANS equivalent of the energy state of α for the above equation is net_j, the computed output for the j^{th} PE of the output layer. This value can be computed by multiplying each connection (i) into the j^{th} PE by the corresponding output value from in_i, the i^{th} PE of the input layer. The equation for computing net_j is as follows:

$$net_j = \sum_i in_i w_{ij}$$

In this equation w_{ij} is the connection strength (weight) of the connection from input PE i to output PE j. Using the ANS equivalence, the ANS form of the Boltzmann distribution function is as follows:

$$out_j = \frac{1}{1 + e^{\frac{-(net_j - \Theta_j)}{T}}}$$

The value of Θ_j is a bias that is associated with each PE. The values this function produces are between 0 and 1. Plotting the output of this function against the probability that it is on ($p(on)$) produces the plot shown in figure 4. Because the curve has a "S" (sigmoid) shape and is bounded between 0 and 1, the function can be considered a threshold function. A threshold function has an output of 1 when it is firing and 0 when it is not.

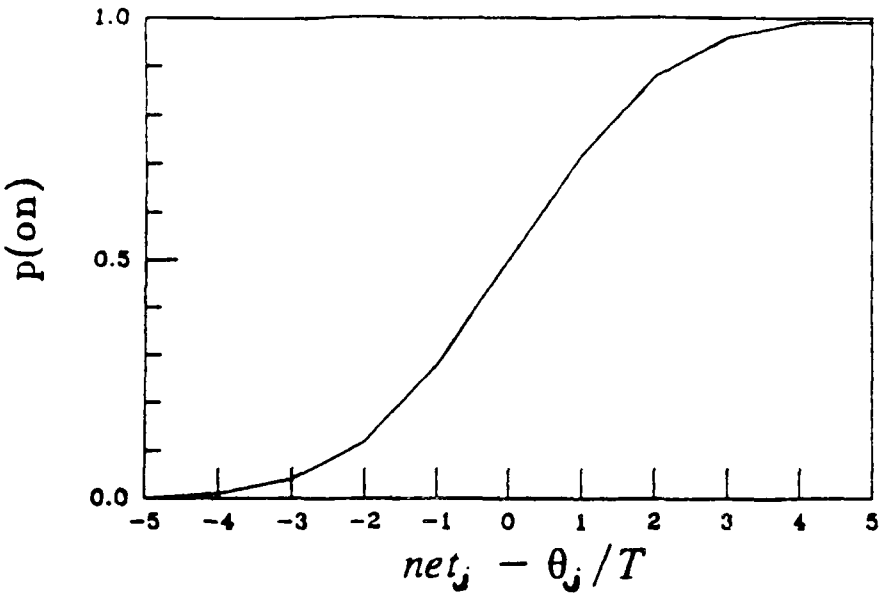


Figure 4: The threshold function curve. *The threshold function used to calculate the probability of firing. The x-axis shows values of* $net_j - \Theta_j/T$ *and the y-axis indicates the corresponding probability of the PE being on (p(on)).*

The Boltzmann Machine ANS also uses a different process for learning, applying the concepts of simulated annealing to the learning process. By using the Boltzmann Machine equation above as a threshold evaluation function for each PE, and by regulating the temperature value T, the ANS can learn more effectively. Creating an ANS creates an energy terrain. By adjusting the weights, the ANS energy terrain is changed. The best energy terrain is one that will have a deep energy well for each entry point. Sometimes the entry point into the ANS is not at a good location; thus the energy minima that is needed is not available. By adding energy to the ANS, called adding noise, the energy well has a better chance of being found from the entry point. Restated, when an ANS is started at an entry point in the energy terrain, the local minima will be sought. What is wanted is not the local energy minima but the global energy minima. By adding noise to the energy terrain, it is possible to bounce out of the the local minimas and eventually find the global minima. Noise is added to the system by increasing the temperature T, and by slowly dropping the temperature (reducing the amount of noise) the ANS is simulating annealing.

In contrast to the Boltzmann Machine model, the Hebb/Hopfield models are confined to there point of entry and are easily trapped in a local minima. The Boltzmann Machine model is able to overcome that problem and find the global minima. Appendix 2 gives an algorithm for mapping patterns of input vectors to

output vectors using this model.

6.4. The Rumelhart/Williams Error Propagation Model

The models presented up to this point, the Hebb/Hopfield and Boltzmann Machine models, are two-layer ANSs. There are many problems in the real world that cannot be represented in a two layer system. One is the exclusive-or (XOR) function [Rumelhart86c]. Because there exist no values that the connection strengths can assume that will give the appropriate output for all inputs, the two-layer ANS is inadequate. Table 2 describes the XOR function, showing the input values in_0 and in_1 and the corresponding output value out_0.

Table 2: The XOR function. *The input values in_0 and in_1 and corresponding output values out_0 can not be represented in a two layer ANS. Using a three layer ANS this mapping is possible.*

in_0	in_1	out_0
0	0	0
0	1	1
1	0	1
1	1	0

Using the two-layer ANS shown in figure 5, no weight assignments can be made to w_{00} and w_{01} that will give a proper output for each of the four XOR inputs patterns shown in table 2.

The solution to this problem is to introduce a third layer, called the hidden layer, between the input and output layers. The hidden layer creates the ability to incorporate an internal representation that facilitates difficult mappings between input and output patterns. By adding the middle layer shown in figure 6 to the ANS shown in figure 5, the XOR function is now representable [Rumelhart86c].

R. Hecht-Nielsen has taken this idea a step further by applying a mathematical existence theorem to ANS. Hecht-Nielsen, using the *Kolmogorov Existence Theorem* which states that any continuous mapping can be done in a three-layer system, has shown that a three-layer ANS exists for any continuous mapping. If there is a continuous mapping from input to output, there exist a three-layer ANS that can represent that mapping [Hecht-Nielsen86a].

D. Rumelhart and R. Williams of UCSD discovered an algorithm that could do the mapping. The Rumelhart/Williams Error Propagation algorithm can do the mapping for a three-layer (or more) ANS. In the three-layer system, the weights are adjusted for the output layer according to an error function that calculates a weight

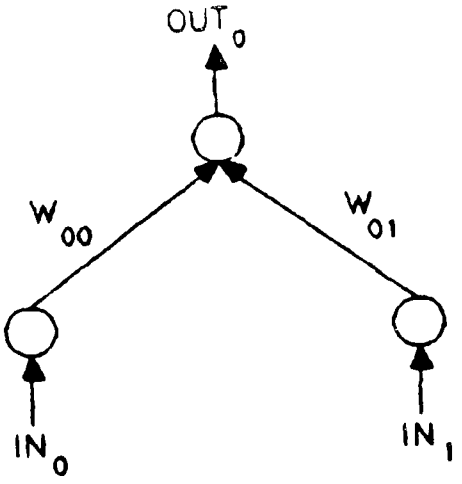


Figure 5: A two-layer ANS for the XOR function. *The weights w_{00} and w_{01} cannot assume values that will produce the proper output for all four input patterns of the XOR function.*

adjustment based upon the difference between the target output and the computed output of each output PE. Each error value for each output PE is then propagated backward to the hidden-layer and used to adjust the values of the weights to the hidden PEs. The error adjustments for the weights to the hidden layer PEs is calculated using the derivative of the error function used to adjust the weights for the output-layer PEs. By using the derivative, the hidden-layer PEs' values are properly adjusted.

6.5. Summary

These models represent an adequate cross-section of ANS technology for the purposes of this survey. In summary, each model is successively more complex and is more able at storing representations. The Hebb/Hopfield models are the least complex and the Error Propagation model is the most complex. The Error Propagation model is the best at storing representations and the Hebb/Hopfield models are the most limited. The Boltzmann Machine falls between the the Hebb/Hopfield and the Error Propagation model in both representation ability and complexity.

7. The Neurocomputers

One inexorable problem associated with implementing ANS is the massive amount of computing that is necessary. As the models get larger and more complex, the computing time becomes exponentially larger. One solution to the problem is to build a computer that is architecturally suited to handle ANS. Because ANS is

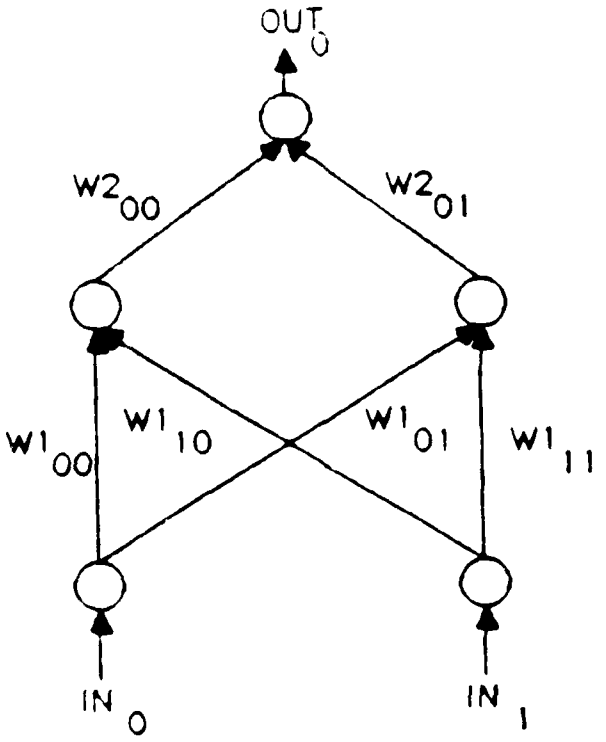


Figure 6: The three-layer ANS for the XOR function. *This ANS allows the representation of the XOR function. The addition of the hidden units between the input and output layer creates an internal representation that makes the difficult XOR mapping possible.*

massively parallel in nature, a computer built with thousands of processors, where each processor takes the place of one PE, would solve the problem.

Many attempts are being made at building ANS computers. The computers being designed for ANS modeling are called neurocomputers and are currently being implemented in two frameworks: electro-optical and electronic [Hecht-Nielsen86b].

Electro-optical computers are designed to use light for the connections between PEs. Because light is able to overlap without interfering, it is a good medium for implementing the high number of connections needed between PEs. Leaders in this research include C. Guest of UCSD and B. Kosko [Kosko87], D. Psaltis and Y. Abu-Mostafa CalTech with N. Farhat of the University of Pennsylvania [Abu-Mostafa87, Brown86a], and Szu of Naval Research Laboratories [Brown86b].

The other method used to implement neurocomputers is electronically. Such neurocomputers have all the interconnections hard-wired and use available transistors and hardware in their implementation. Size and cost are no longer the inhibiting factors

they had been 10 years ago because of the dramatic advancements in electronic circuitry. One leader in this area is R. Hecht-Nielsen, who has started his own company that produces neurocomputers, the Hecht-Nielsen Neurocomputer Corporation (HNC). HNC will market a board that fits into an IBM PC/AT card slot and can be used to implement many different neural models, including those presented in this paper [Brown87, HNC86]. This neurocomputer has a capacity for 30,000 PEs with 300,000 interconnections.

Other leaders in electronic neurocomputers include Nestor and TRW. Nestor is the only company that has marketed an ANS application. Nestor has a patented hardware system that allows handwritten input to a computer via a digitized pad [Nestor86]. TRW has entered the ANS market place with its Mark III neurocomputer [TRW86], designed by Hecht-Nielsen before he left TRW in late 1986 and formed his own company.

Neurocomputers will not replace the existing computer. The neurocomputers being designed are subservient members of a Von Neumann computer. Programs that use neurocomputers make a subroutine call to the neurocomputer to do its specialized work. Neurocomputers will have their own software which will be able to integrate with existing software to create a machine with added capability and potential.

Acknowledgements

I would like to thank Stephen Luse, Stephen Nunn, and Dennis Kocher of Naval Ocean Systems Center's Speech Technology Group (Code 441) as well as Dave Voldal and Dale Larson of Unisys's San Diego Systems Engineering Center for their technical advice, comments, and criticisms. Sharon Walker of Arizona State University also deserves thanks for helping me to get this paper into its final form. Most importantly I would like to thank my wife Christalyn for her patience, support, and comments while writing this paper.

References

[Abu-Mostafa87]
Abu-Mostafa, Y., "Optical Neural Computers", *Scientific American*. Pgs. 88-95 (March 1987)

[Amari71]
Amari, S., "Characteristics of Randomly Connected Threshold-Element Networks and Network Systems", *Proceedings of the IEEE*. Vol. 59, No. 1, Pgs. 35-47 (January 1971)

[Bernstein81]
Bernstein, J., "Profiles: Marvin Minsky", *The New Yorker*. Pgs. 50-126 (December 14, 1981)

[Brown73]
Brown, R., **A First Language.** Cambridge, MA: Harvard University Press (1973)

[Brown86a]
Brown, C., "Parallel Optics: Solution to Von Neumann Bottleneck?", *Electronic Engineering Times*. Pg. 35 (November 24, 1986)

[Brown86b]
Brown, C., "Parallel Optics Computer Solves Mulitvariable Problems in Real-time", *Electronic Engineering Times*. Pgs. 31-32 (December 1986)

[Brown87]
Brown, C., "Neural Network Startups Backed by Venture Capital", *Electronic Engineering Times*. Pgs. 23-24 (January 1987)

[Chester86]
Chester, M. "Fuzzy Logic's Soft Start Belies a Cutting Edge", *Electronic Products*. Pgs. 19-20 (June 1986)

[Cottrell84]
Cottrell, G. and Small, S., "Viewing Parsing as Word Sense Discrimination: A Connectionist Approach", **Computational Models of Natural Language Processing.** Bara, B. and Guida, G. (Eds.), Elsevier Science Publishers, B.V. North-Holland (1984)

[Crick79]
Crick, F., "Thinking About the Brain", *Scientific American*. Vol. 241, Pgs. 219-232 (1979)

[Elman87]

Elman, J. and Zipser, D., "Learning the Hidden Structure of Speech", *University of California, San Diego Technical Report*. Department of Linguistics and Institute for Cognitive Science, University of California: San Diego (1987)

[Epstein86]

Epstein, G., "Nestor Learning System Applied to a Speaker Independent Voice Application", *Nestor Inc.* One Richmond Square, Providence, RI 02906 (November 1986)

[Ervin64]

Ervin, S., "Imitation and Structural Change in Children's Language", **New Directions in the Study of Language.** Lenneberg, E. (Ed.), Cambridge, MA: MIT Press (1964)

[Fanty85]

Fanty, M., "Context-Free Parsing in Connectionist Networks", *University of Rochester Technical Report*. Computer Science Department, University of Rochester: New York (1985)

[Feldman82]

Feldman, J., "Dynamic Connections in Neural Networks", *Biological Cybernetics*. Vol. 46 (1982)

[Froelich86]

Froelich, W., "Dawn Glimmers for Day of the Man-made Brain", *The San Diego Union*. San Diego, CA: Pgs. A9-A11 (July 6, 1986)

[Gold86]

Gold, B., "Hopfield Model Applied to Vowel and Consonant Discrimination", *Lincoln Laboratory Technical Report 747*. Massachusetts Institute of Technology, Lexington, MA: (June 1986)

[Grossberg68]

Grossberg, S., "Some Nonlinear Networks Capable of Learning a Spatial Pattern of Arbitrary Complexity", *Proceedings National Academy of Sciences*. Vol. 59, Pgs. 368-372 (1968)

[Grossberg85]

Grossberg, S. and Kuperstein, M., **Neural Dynamics of Adaptive Sensory-Motor Control: Ballistic Eye Movements.** Elsevier/North Holland: Amsterdam (1985)

[HNC86]

Hecht-Nielsen Neurocomputer Corporation, "Advanced Neurocomputer Applications Course", *Hecht-Nielsen Neurocomputer Corporation*, 5893 Oberlin Drive, San Diego, CA 92121 (1986)

[Hecht-Nielsen86a]

Hecht-Nielsen, R., "Artificial Neural System Design", *UCSD Extension Class Notes*. Visiting Professor: TRW Rancho Carmel Artificial Intelligence Center, One Rancho Carmel, San Diego, CA 92128 (1986)

[Hecht-Nielsen86b]

Hecht-Nielsen, R., "Performance of Optical, Electro-Optical, and Electronic Neurocomputers", *TRW Rancho Carmel Artificial Intelligence Center Technical Report*. One Rancho Carmel, San Diego, CA 92128 (1986)

[Hecht-Nielsen86c]

Hecht-Nielsen, R., "Nearest Matched-Filter Classification of Spatiotemporal Patterns", *HNC Technical Report*. Hecht-Nielsen Neurocomputer Corporation: San Diego (1986)

[Hinton86a]

Hinton, G., McClelland, J., and Rumelhart, D., "Distributed Representations", **Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume One: Foundations.** Pgs. 45-76, Cambridge, MA: Bradford Books/MIT Press (1986)

[Hinton86b]

Hinton, G., and Sejnowski, T., "Learning and Relearning in Boltzmann Machines", **Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume One: Foundations.** Pgs. 282-317, Cambridge, MA: Bradford Books/MIT Press (1986)

[Hopfield82]

Hopfield, J., "Neural Networks and Physical Systems With Emergent Collective Computational Abilities", *Proceedings of the National Academy of Science USA*. Vol. 79, Pgs. 2554-2558 (April 1982)

[Hopfield84]

Hopfield, J., "Neurons with Graded Response have Collective Computational Properties Like Those of Two-State Neurons", *Proceedings of the National Academy of Science USA*. Vol. 81, Pgs. 3088-3092 (May 1984)

[Johnson87]

Johnson, R., "Neural Systems Make a Comeback After 50-Year Gestation", *Electronic Engineering Times*. Pg. 23 (January 5, 1987)

[Jorgensen86]

Jorgensen, C. and Matheus, C., "Catching Knowledge in Neural Nets", *AI Expert*. Vol. 1, No. 4, Pgs.30-41 (December 1986)

[Kohonen84]

Kohonen, T., **Self-Organization and Associative Memory.** Springer-Verlag, Berlin. (1984)

[Kosko86]

Kosko, B., "Fuzzy Entropy and Conditioning", *Information Sciences*. (1986)

[Kosko87]

Kosko, B. and Guest, C., "Optical Bidirectional Associative Memories", *Society for Photo-optical and Instrumentation Engineers (SPIE) Proceedings: Image Understanding*. Vol. 75 (January 1987)

[Larson86]

Larson, E., "Neural Chips", *Omni*. Pgs. 113-116, 168-169 (December 1986)

[Levy82]

Levy, W., "Associative Encoding at the Synapses", *Proceedings of the 4th Annual Conference of the Cognitive Science Society*. Ann Arbor, MI (1982)

[Lindsay77]

Lindsay, P. and Norman, D., **Human Information Processing: An Introduction to Psychology** Orlando, FL: Academic Press, Inc. (1977)

[McClelland86a]

McClelland, J., Rumelhart, D., and Hinton, G., "The Appeal of Parallel Distributed Processing", **Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume One: Foundations.** Pgs. 3-44, Cambridge, MA: Bradford Books/MIT Press (1986)

[McClelland86b]

McClelland, J., and Rumelhart, D., "Distributed Memory and the Representation of General and Specific Information", *Journal of Experimental Psychology: General*. Vol. 114, No. 2, Pgs. 159-188 (1985)

[Myers86]

Myers, M, "Some Speculations on Artificial Neural System Technology", *Proceedings of the IEEE National Aerospace and Electronics Conference - NAECON*. Pgs. 1298-1302 (May 1986)

[Nestor86]

Nestor Inc., "Background: Nestor and the Nestor System", *Nestor Inc*. One Richmond Square, Providence, RI 02906 (1986)

[Port87]

Port, O., "They're Here: Computers that 'Think'", *Business Week*. Pgs. 94,98, (January 26, 1987)

[Reilly82]

Reilly, D., Cooper, L. and Elbaum, C., "A Neural Model for Category Learning", *Biological Cybernetics*. Vol. 45, Pgs. 35-41, Springer-Verlag: (1982)

[Rimey86]

Rimey, R., Gouin, P., Scofield, C. and Reilly, D. "Real Time 3-D Classification Using a Learning System", *Proceedings of the SPIE Cambridge Symposium on*

Intelligent Robots and Computer Vision. Cambridge, MA: (October 1986)

[Rumelhart86a]

Rumelhart, D. and Zipser, D., "Feature Discovery by Competitive Learning", **Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume One: Foundations.** Pgs. 151-193, Cambridge, MA: Bradford Books/MIT Press (1986)

[Rumelhart86b]

Rumelhart, D., Hinton, G., and McClelland, J., "A General Framework for Parallel Distributed Processing", **Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume One: Foundations.** Pgs. 45-76, Cambridge, MA: Bradford Books/MIT Press (1986)

[Rumelhart86c]

Rumelhart, D., Hinton, G., and Williams, R., "Learning Internal Representations by Error Propagation", **Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume One: Foundations.** Pgs. 318-362, Cambridge, MA: Bradford Books/MIT Press (1986)

[Rumelhart86d]

Rumelhart, D. and McClelland, J, "On Learning the Past Tense of Verbs", **Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume Two: Psychological and Biological Models.** Pgs. 216-271, Cambridge, MA: Bradford Books/MIT Press (1986)

[Sejnowski86]

Sejnowski, T. and Rosenberg, C., "NETtalk: A Parallel Network that Learns to Read Aloud", *Johns Hopkins University Electrical Engineering and Computer Science Technical Report JHU/EECS-86/01*. Johns Hopkins University, Baltimore, MD 21218 (1986)

[TRW86]

TRW Rancho Carmel Artificial Intelligence Center, "Mark III: Artificial Neural System Processor", *TRW Rancho Carmel Artificial Intelligence Center*. One Rancho Carmel, San Diego, CA 92128 (1986)

[Willshaw69]

Willshaw,D. and Longuet-Higgins, H., "The Holophone - Recent Developments", **Cognitive Processes: Methods and Models.** Pgs. 349-357, University of Edinburgh (1969)

Appendix 1

1. The Hebb/Hopfield Algorithms

The following are learning and recall algorithms for the Hebb and Hopfield ANS models. These algorithms are designed to store a single pattern. A pattern is a mapping of an input vector $\vec{in}$ to an output vector $\vec{out}$. If the input and output vectors are the same, as they are in these models, the model is acting as an associative memory. Although these models are designed to store one binary valued vector, they can easily be extended to store several binary vectors. These models can also be expanded to store mappings between vectors differing in length, value, or both.

1.1. The Hebb Model Learning Algorithm

The Hebb model learning algorithm is as follows:

1. Given an input vector of length N called $\vec{in}$ with binary values from in_0 to in_{N-1}.
2. Construct a duplicate vector to the input vector, call this vector $\vec{out}$; it is also indexed from 0 to N-1.

```
for i = 0 to (N-1) do
        out_j = in_j
enddo
```

3. Construct a weight matrix N x N that is initialized to all zeroes called W.

```
for i = 0 to (N-1) do
        for j = 0 to (N-1) do
                W_ij = 0
        enddo
enddo
```

4. Generate appropriate values for each position in the matrix W_{ij} where $i \neq j$ according to their similarity as follows:
   - If in_i and out_j are both equal to 1, add strength to the connection between them (via an increase in the value stored for this connection in the weight matrix).
   - If in_i and out_j are both equal to 0, subtract strength from the connection between them (via a decrease in the value stored for this connection in the weight matrix).
   - Otherwise, continue.

The algorithmic form is as follows:

```
for i = 0 to (N-1) do
    for j = 0 to (N-1) do
        if i ≠ j then
            if in_i = out_j = 1 then
                W_ij = W_ij + 1
            elseif in_i = out_j = 0 then
                W_ij = W_ij - 1
            endif
        endif
    enddo
enddo
```

The following example illustrates how this algorithm works:

1. Consider the five-element input vector $\vec{in}$ = 01110.
2. Creating a duplicate vector gives the output vector $\vec{out}$ = 01110.
3. Creating a weight matrix W of dimensions 5×5 with each slot in the vector initialized to zero yields the following matrix:

$$W = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

4. Generating the values for W according to the Hebb Learning Rule results in the following computations:

$in_0 = 0$ and $out_1 = 1$, so W_{01} is unchanged
$in_0 = 0$ and $out_2 = 1$, so W_{02} is unchanged
$in_0 = 0$ and $out_3 = 1$, so W_{03} is unchanged
$in_0 = 0$ and $out_4 = 0$, so $W_{04} = -1$

Continuing with this for in_1, in_2, and in_3 will yield the adjusted matrix:

$$W = \begin{bmatrix} 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

1.2. The Hebb Model Recall Algorithm

The weight matrix should now hold the vector 01110. To recall this vector from the matrix, the following algorithm is used:

1. Given a weight matrix W that has dimensions $N \times N$ that is storing a vector of length N.
2. Sum over each row of W (the equivalent of summing up all the connection strengths entering output element j); store each sum in a vector $\vec{net}$ at the j^{th} index.

```
for j = 0 to (N-1) do
        sum = 0
        for i = 0 to (N-1) do
                sum = sum + W_ij
        enddo
        net_j = sum
enddo
```

3. Test each element of the vector $\vec{net}$ and reset its value as follows:

```
for j = 0 to (N - 1) do
        if net_j > 0 then
                net_j = 1
        elseif net_j < 0 then
                net_j = 0
        else
                net_j = 1 or net_j = 0 with a probability of 0.5
        endif
enddo
```

Using the W computed for the vector $\vec{in}$ = 01110, we get the following:

$net_0 = 0 + 0 + 0 + 0 + (-1) = -1$
$net_1 = 0 + 0 + 1 + 1 + 0 = 2$
$net_2 = 0 + 1 + 0 + 1 + 0 = 2$
$net_3 = 0 + 1 + 1 + 0 + 0 = 2$
$net_4 = (-1) + 0 + 0 + 0 + 0 = -1$

3. From the values computed above, we reset each value of the vector $\vec{net}$ as follows:

$net_0 < 0$, *so* $net_0 = 0$
$net_1 > 0$, *so* $net_1 = 1$
$net_0 > 0$, *so* $net_2 = 1$
$net_0 > 0$, *so* $net_3 = 1$
$net_0 < 0$, *so* $net_4 = 0$

and the vector recalled from the matrix is the vector 01110, the same vector the ANS was taught.

1.3. The Hopfield Model Learning Algorithm

In the Hopfield model, step 4 of the Hebb learning algorithm is changed to the following:

4. Generate appropriate values for each position in the matrix W_{ij} where $i \neq j$ using the following equation:

$$W_{ij} = (2in_i - 1)(2out_j - 1)$$

This equation does the following:

- If in_i and out_j are both equal to 1 or both are equal to zero, store a 1 at the matrix position W_{ij}.
- Otherwise, store a -1 at matrix position W_{ij}.

The algorithm for this step is as follows:

```
for i = 0 to (N-1) do
        for j = 0 to (N-1) do
                if i≠j then
                        W_ij = (2in_i - 1)(2out_j - 1)
                endif
        enddo
enddo
```

1.4. The Hopfield Model Recall Algorithm

In the Hopfield model, step 3 of the Hebb recall algorithm is changed. When constructing the recall vector $\vec{net}$, use the following:

3. Test each element of the vector $\vec{net}$ and reset its value as follows:

```
for j = 0 to (N - 1) do
        if net_j >= 0 then
                net_j = 1
        else
                net_j = 0
        endif
enddo
```

Using the same example given before, given the vector $\vec{in} = 01110$, we get the following adjusted matrix W:

$$W = \begin{bmatrix} 0 & -1 & -1 & -1 & 1 \\ -1 & 0 & 1 & 1 & -1 \\ -1 & 1 & 0 & 1 & -1 \\ -1 & 1 & 1 & 0 & -1 \\ 1 & -1 & -1 & -1 & 0 \end{bmatrix}$$

Appendix 2

1. The Boltzmann Machine Model Algorithms

The following are the learning and recall algorithms for the Boltzmann Machine ANS model. In the learning stage this ANS model associates an input vector with a different output vector. In the recall stage this ANS model is given an input vector and the associated output vector is recalled. This type of model is considered a pattern association model.

1.1. The Boltzmann Machine Learning Algorithm

The Boltzmann Machine Learning Algorithm is implemented in this model using a tolerance value to test when the model has satisfactorily learned all the patterns it has been presented. This model will continue to learn until *all* the differences between the computed output value net_j and the target value out_j are within the specified tolerance. This model will start with a tolerance of 0.1 and as the annealing process progresses the tolerance value will decrease.

This model assigns a base temperature (T) of 25 and a base learning rate (η) of 0.1 at the start. These two values are related to each other as follows: The higher the temperature the lower the learning rate needs to be and vice-versa. In this model the learning rate (η) increases by 0.1 each time the temperature (T) decreases by 5. The changes in T and η occur each time the tolerance is satisfied, and each time the tolerance is satisfied it is decreased by 0.02.

A new notation is also introduced to show the pattern number being learned. In the algorithm that follows, in_{pi} represents the p^{th} input vector's i^{th} element. The notation out_{pj} represents the p^{th} output vector's j^{th} element. The threshold is set to be zero throughout this model. The weight matrix W is initialized to random values between 0 and 0.3 to prevent any oscillations that might occur from a weight matrix of all zeroes. An *epoch* is a complete cycle through all weight adjustments for all patterns the ANS is being presented. The learning algorithm is as follows:

1. Set $T = 25$, $\eta = 0.1$, tol = 0.3, epoch = 0 and $\Theta = 0$.
2. Initialize the weight matrix to hold random values between 0 and 0.3. The length of the input vectors is N and the length of the output vectors is M, so the weight matrix is an $N \times M$ matrix.

```
for i = 0 to (N-1) do
        for j = 0 to (M-1) do
                w_ij = random value from 0 to 0.3
        enddo
enddo
```

3. Get the input and output vectors. The input vector is of length N, the output vector is of length M, and there are P of these associations (patterns).

```
/* Get input vectors */
for p = 0 to (P-1) do
        for i = 0 to (N-1) do
                in_pi = binary value
        enddo
enddo
/* Get output vectors */
for p = 0 to (P-1) do
        for j = 0 to (M-1) do
                in_pj = binary value
        enddo
enddo
```

4. Now that the weight matrix W is initialized and the patterns are stored, the learning can now begin. To anneal the ANS, two flags are used. The first is the *learn_flag* that will tell when the whole ANS is done learning. The other flag is *tol_flag* which tells when the ANS has learned within the specified tolerance and is ready to have T, *tol*, and η adjusted for the next step in annealing.

```
set learn_flag = TRUE
while learn_flag ≠ TRUE do
        set tol_flag = FALSE
        while tol_flag = FALSE do
                set tol_flag = TRUE
                for p = 0 to (P-1) do
                        for j = 0 to (M-1) do
                                net_j = Σ_i w_ij in_pi
                        enddo
                        net_j = f_sigmoid(net_j)
                        for i = 0 to (N-1) do
                                δ_ij = η * (out_pj - net_j) * in_pi
                                w_ij = w_ij + δ_ij
                                if tol_flag = TRUE
                                    and |out_pj - net_j| > tol then
                                        tol_flag = FALSE
                                endif
                        enddo
                enddo
        endwhile
        T = T - 1
        η = η + 0.1
        tol = tol - 0.05
        if tol = 0 then
                learn_flag = TRUE
        endif
endwhile
```

This fourth step of the learning algorithm continues learning each pattern at each tolerance until all the tolerances are satisfied. This ANS follows an annealing schedule that starts at a $T = 25$, $\eta = 0.1$, and $tol = 0.3$, finishing with $T = 5$, $\eta = 0.5$, and $tol = 0.1$.

1.2. The Boltzmann Machine Recall Algorithm

The recall algorithm is much simpler than the learning algorithm. This algorithm is given an input vector $\vec{test}$ and computes an output vector $\vec{recall}$ from the $\vec{test}$ and the connection strengths stored in W.

1. Get the input vector $\vec{test}$.

for i = 0 to (N-1) do
$test_i$ = some binary value
enddo

2. From the given input vector calculate an output vector $\vec{recall}$ by doing the following:

for j = 0 to (M-1) do
$net_j = \sum_i w_{i,j} test_i$
enddo
$recall_j = f_{threash}(net_j)$

The output vector $\vec{recall}$ should resemble the output vector originally associated with the given input vector $\vec{test}$. An example of this ANS is as follows. If you train the ANS with the three patterns

10001→01110
11100→00111
10101→01010

the ANS takes approximately 1700 epochs to satisfy all the specified tolerances. The final weight matrix W is:

$$W = \begin{bmatrix} -11.88 & 2.75 & 26.52 & 11.80 & -2.87 \\ -3.42 & -18.54 & 33.53 & 3.57 & 18.72 \\ -6.45 & -4.23 & -40.04 & 6.25 & 4.24 \\ 0.30 & 0.10 & 0.30 & 0.10 & 0.20 \\ -8.16 & 21.50 & -6.50 & 8.23 & -21.39 \end{bmatrix}$$

This is the same model used by Rumelhart and McClelland in their study in which the past tense of English verbs was learned. Their model was much larger (more input and output PEs), but it computes the same way.

END
DATE
FILMED

8-12-87