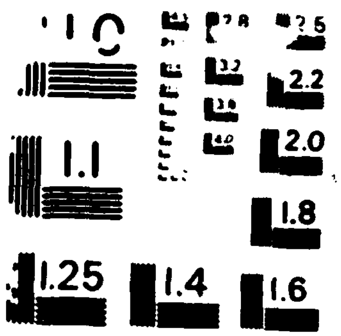
AD-A193 827 ANALYSIS OF THE HOPFIELD NEURAL NETWORKS AND THEIR APPLICATION TO PATTERN RECOGNITION(U) ILLINOIS UNIV AT URBANA DEPT OF ELECTRICAL AND COMPUTER ENGINEERING 1/1

UNCLASSIFIED B F GERASIMAS 1988 F/G 6/4 NL



TEST CHART

ANALYSIS OF THE HOPFIELD NEURAL NETWORKS AND THEIR APPLICATION TO PATTERN RECOGNITION

BY

BERNARD F. GERASIMAS, JR.

B.S., United States Military Academy, 1977

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1988

Urbana, Illinois

ACKNOWLEDGEMENTS

I would like to thank my advisor, Professor W.K. Jenkins, for his support and guidance during the writing of this thesis. I greatly appreciate his time and insight, and the opportunity to be introduced to the exciting topic of neural networks. Also, I would like to express special thanks to Cindy, my wife, for her great patience and support during my entire graduate program.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

For centuries, man has tried to explain the biological functioning of the human brain as related to memory and the senses. As biomedical technology advanced at the turn of the century, man discovered that the human brain and nervous system were made up of cells that were similiar in structure and in function. These cells, called neurons, were responsible for gathering, passing, and storing of information. In 1943, the first mathematical model [1] of the operation of a neuron was introduced. This was soon followed by new theories [2], [3], and [4] on the interaction between neurons. With further advances in biomedical technology, it became evident that the paralleled structure and interaction between neurons were the essential factors in the overall operation of the human brain and nervous system [5] and [6]. In the late 1970's, J.J. Hopfield's research and findings in modeling a neural network spurred new interest in this area [7], [8], [9], [10], and [11]. Of all the well-known neural network models, the Hopfield discrete and continuous models adapt most readily to the task of pattern recognition [12], [13], and [14].

This thesis will analyze the Hopfield discrete and continuous neural network models in representing the functions of memory and pattern recognition in the brain. Models of three different sizes will be simulated on a digital computer using the Microsoft Fortran

V4 Compiler. They will be required to learn and recall several patterns that vary in both shape and proportion. The models will then be tested on the ability to recognize distorted versions, with and without noise, of a learned pattern. Based on the results obtained, the performance of the models will be examined and a comparison will be made.

CHAPTER 2

BACKGROUND

This three-part discussion provides the necessary background in understanding the make-up and operation of the generalized biological neuron and its role in memory and pattern recognition in the brain. There are many specific types of neurons which have adapted to perform specialized operations as part of the central nervous system. Motor neurons deal with the operation of the muscles; optical neurons deal with the operation of the eyes. Although there are differences in physical make-up, their basic operation is similiar and is known as the generalized biological neuron. The physical structure and operation of this neuron will be examined. A brief discussion will follow on how the inter-connection and paralleled structure of neural networks in the optical system and cortex of the brain are able to send, store, and recall information dealing with pattern recognition. This will be followed by a brief discussion of the first mathematical model developed [1] of the generalized biological model. Its make-up is the building block for the vast majority of neural network models which followed. During the remainder of this section, unless otherwise noted, the term neuron will represent the generalized biological neuron model. Most of the information for this section is obtained from references [1], [2], [3], [4], [6], [10], [12], [15], and [16].

2.1 Generalized Biological Neuron

As depicted in Fig. 2.1, the neuron is made up of four basic parts. The main central body is known as the SOMA. It performs an analog computation in response to inputs and produces an output which is functionally related to the inputs. When dormant, it remains at a normal potential of -70 mV with respect to the local

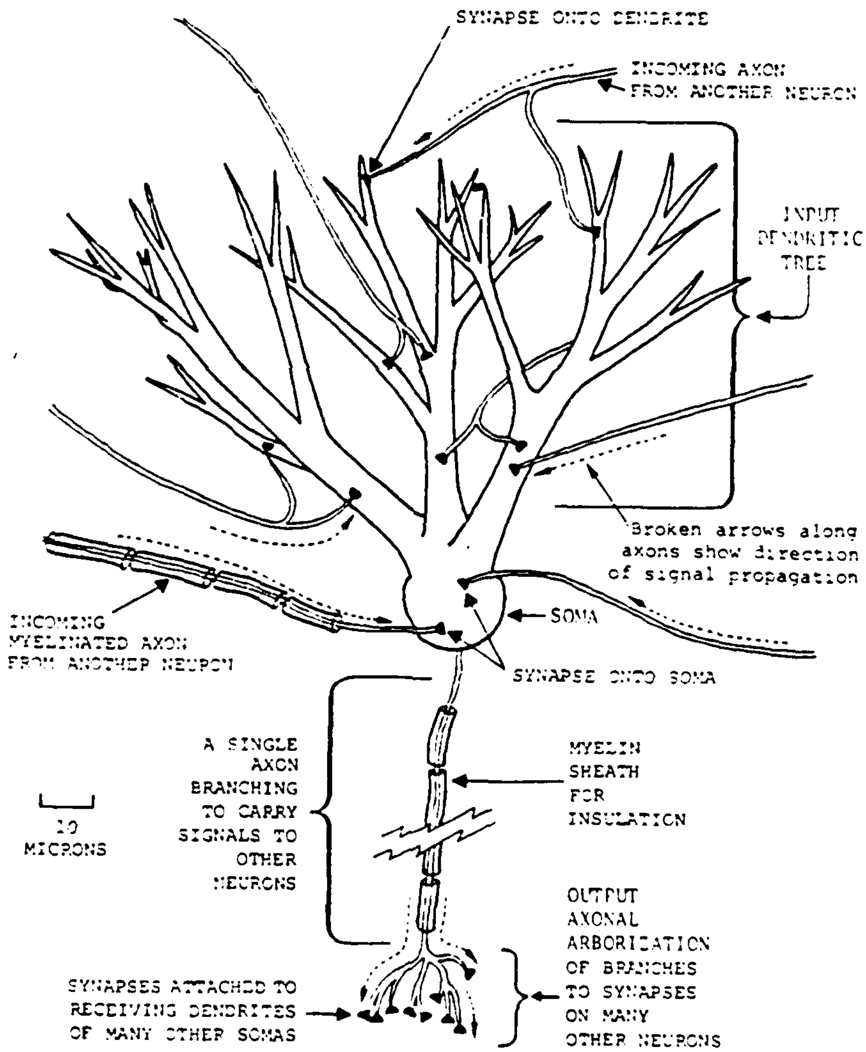


Fig. 2.1 Biological Neuron (From Wittie [11])

intercellular fluid used as ground. When excited, it can have a positive rising amplitude of about 100 mV. At excitation, the soma's output consists of a sequence of short, constant amplitude pulses with a variable repetition rate. The frequency of repetition is the vehicle by which information is encoded into the signal.

This output is transmitted from the soma through its output organ, the AXON. The axon is a single fiber which branches out into many fibers. Because of this, it is able to transmit the same data to many other neurons. Signals travel unattenuated in the axon at velocities ranging from one meter per second to as much as 120 meters per second. Velocities are dependent upon the axon diameter and length. Unattenuated transmission is achieved by a process analogous to the propagation of a short down a charged transmission line. In addition to the soma, the axon is maintained at a normal potential of -70 mV with respect to its outer fluid. This potential appears across its thin walls which are like an insulator during the nontransmission periods. These walls also have a very high capacitance due to their thinness. Thus the axon is like a long cylindrical hollow capacitor whose walls are charged at -70 mV with respect to its outer fluid. Once the soma has exceeded a certain "firing" potential threshold, it excites the axon. The capacitor depolarizes and a ring of potential change propagates down the axon utilizing the energy stored in the distributed capacitance. Once the pulse has passed, the axon chemistry starts to recharge the capacitor.

The pulse is transmitted to neurons that have formed a connection through their input fibers. These input fibers act as antennae for the soma. This group of extensively interconnected input branches, which extend out of the soma and are opposite to the axon, are known as DENTRITES. They gather incoming data to be transmitted into the soma. Output data from other neurons may also be received directly through the soma wall. The point of connection between axons (output devices) and dentrites or the soma (input devices) are called SYNAPTIC JUNCTIONS. The magnitude and number of inputs raises the potential of the soma above -70 mV. At a certain threshold potential, the soma will emit or fire a signal impulse. For a short period of time, just after firing, it is impossible for the soma to be fired by any of its usual stimuli. This period of time is known as the "Absolute Refractory Period." Following this period, the refractoriness dissipates and the neuron becomes increasingly easy to fire. This is how the soma can encode intensity of stimulus into rate of firing. Pulse widths are typically about 0.50 ms, and pulse repetition rates range from zero to about 1000 per second. A major point of disagreement between many experts in this area is with the significance of the synaptic junctions [1], [2], [3], [4], [5], [10], [12], [15], [16], and [17]. This dispute will be discussed in Section 2.3.

2.2 Pattern Recognition and Memory in the Brain

The path in processing optical information for pattern recognition consists of the eye, optic nerve, and cortex. The cortex is located in the upper part of the brain just below the skull. The process begins when receptor cells, rods and cones, transform incoming light signals to electrochemical signals which resemble spike potentials. Previous microelectrode studies have indicated that the information represented by these spike potentials is a rather sophisticated extraction of pattern features of the light stimulus. This coding is accomplished by a mosaic of approximately 125 million receptor cells which function independently. Their output is an enormous amount of data most of which cannot be transmitted to the brain due to the limited channel capacity of the optic nerve. No such high capacity channel exists in biological systems; instead, many low capacity channels are paralleled and ed simultaneously. The receptor cell output is transmitted as input to a large number of relay or recoding neurons located behind the eye. These neurons are called ganglion cells. The ganglion cells respond to inputs from the receptor cells with a short burst of pulses to both the onset or cessation of stimulating illumination. A reduction of data and transmission channels takes place at this junction because there are only one million ganglion cells available to process this enormous amount of data. The outputs of the ganglion cells travel through the optic nerve to the cortex of the brain.

There is a price that must be paid by this system due to the limited channel capacity which cannot relay all data accumulated by the receptor cells. The peripheral vision is not nearly as detailed as that from the central region of the eye. Sharp central vision is obtained for about two degrees of the total visual field seen by the eye. In this central region, there are as many optic nerve channels as there are receptors and ganglion cells. The portion of the receptors surrounding the central region accounts for the remainder of the visual field. The outputs of more than 140 receptors are relayed into a single channel of the optic nerve to the brain.

The outputs from the optic nerve are mapped on a two-dimensional surface known as the cortex which is capable of highly adaptable behavior. The cortex is the outer part at the highest region of the brain. It is 2.5 mm thick, and consists of massive amounts of interconnected vertically structured neurons. There are at least a total of 10^{12} neurons located in the brain; and on the average, each neuron receives and sends 10^3 to 10^4 variable interconnections to other neurons. The density of neurons in the cortex varies from a maximum of about 1740 for each 0.01 mm^2 of cortex surface to a minimum of about 910 for the same area. On the cortex, there is duplicated, in a pattern of neuron firing rates, the images of the receptor cells as though they had been laid out flat in a two-dimensional picture. As input signals are applied to the cortex, there is a connection process by which output data are produced at the same spot on the cortex, but on a different axon from where it entered and at a different frequency of pulse repetition rate from the original frequency. From the discussion of neuron operation in Section 2.1, there is no doubt that the cortex is constructed to perform such a frequency conversion function. It is likely that these networks have at least a short term memory if not a long term memory. The mechanics of how a long term memory could exist in this region are a point of disagreement previously mentioned and will be discussed in the next section. A common point of agreement is that the frequency conversion process that occurs in the cortex is due to the memory of the cortex. For modeling and simulation purposes, pulse repetition rates may be rendered by different numerical values.

2.3 Initial Modeling of Biological Neurons

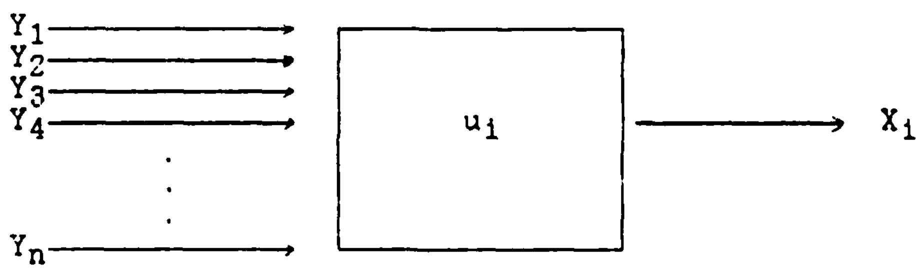


Fig. 2.2 McCulloch and Pitts Model [1]

The mathematical model of the neuron, as depicted in Fig. 2.2, was produced by the team of McCulloch and Pitts [1], in 1943. They modeled all inputs (Y_n), to the neuron, as having either an excitatory (+) potential effect or an inhibitory (-) potential effect on the overall ground potential of -70 mV maintained by the soma. McCulloch and Pitts postulated that the analog operation of the neuron could be simulated by a unit step function. If the summation of all positive and negative inputs to the soma were greater than some threshold potential (u_i), then the neuron would give a constant continuous amplitude output (X_i) until the soma potential fell below the threshold due to a decrease in positive inputs. A neuron with potential below the threshold would have no output.

Shortly after McCulloch and Pitts produced their model, there was a disagreement among experts as to the mechanics of how long term memory was maintained in the interconnection within neural networks in the cortex of the brain. In 1952, Eccles [4] postulated that the actual formation or deformation of the interconnections between neurons was responsible for the storage of memory. Formation or deformation of the interconnections would take place based on frequency of use between neurons. In 1959, Bok [3] postulated that the synaptic junctions, at which neural input and output fibers were connected, influenced data flow through them. This was more in line with the McCulloch and Pitts model and was later adopted in the Hopfield model. If there were frequent interactions between two neurons, the input from neuron i to neuron j would be weighted with an excitatory (+) potential. The opposite weighting would occur between neurons of infrequent interaction. In the remainder of the 1950's and 1960's, research and modeling in this field were slowed due to a technological lag in gaining further information about the brain as well as the shift of interest to the newly invented Boltzman machine (otherwise known as the digital computer).

In the middle 1970's, work done by Kohonen [7] on the adaptive associative memory principle (commonly known as content addressable memory) and a linear analog neural model developed by Minsky and Papert [5], from MIT, influenced J. J. Hopfield. In 1982, Hopfield published his work and introduced his discrete neural network model [12]. This model renewed academic interest in the neural networking and parallel computing field. References [17] and [18] give an excellent listing and explanation of the popular neural network models which are in use.

CHAPTER 3

APPLICATION OF THE HOPFIELD NEURAL NETWORKS TO PATTERN RECOGNITION

This two-part discussion will examine the Hopfield discrete and continuous neural network models, and how they are applied to a digital computer simulation representing neural network memory and pattern recognition in the cortex of the brain. This discussion will include the technique used to discretize the continuous model for representation by a Fortran (77) program. The neural networks will take the shape of symmetric arrays with one neuron occupying each space. Each neuron will be interconnected with every other neuron in the network array. Most of the information for this section is obtained from references [12], [13], [14], [18], and [19].

3.1 The Hopfield Discrete Neural Network Model

In 1982, J.J. Hopfield published the algorithm and findings for the model of the discrete neural network. This model was revised in 1984. The 1984 discrete model will be used in this thesis for it adapts more readily to pattern recognition.

As previously discussed, the Hopfield model assumes the inputs to a neuron i from neurons j are of either excitatory (+) potential or inhibitory (-) potential. This weighting is accomplished through the synaptic junctions between neurons. Learning and storage of memory in the neural network are accomplished through the learned synaptic weights (T_{ij}) as follows

$$T_{ij} = \sum_{k} (2V_i^k - 1)(2V_j^k - 1) \qquad (3.1)$$

$$i = 1,N \qquad T_{ij} = \emptyset$$
$$j = 1,N \qquad T_{ij} = T_{ji}$$

where V_i represents the output of neuron i, V_j represents the output of interconnected neurons j, k represents the number of individual patterns learned by the neural network, and N represents the number of neurons in the network.

By the learning algorithm, if neuron i is excited by the same pattern as neuron j, then there is an excitatory (+) learned synaptic weight T_{ij} between them. If neuron i is not excited by the same pattern as neuron j, then there is an inhibitory (-) learned synaptic weight T_{ij} between them. As the number of patterns k increases for the network, the magnitude of the learned synaptic weights T_{ij} between individual neurons will also change. Thus, the storage of memory in the neural network model is accomplished through the learned synaptic weight matrix T_{ij}. $T_{ij} = T_{ji}$ represents the assumption that an individual neuron has no feedback circuit of its output onto itself. An example of learned synaptic weight T_{ij} matrices for neuron a_{11} and neuron a_{13} of a 25 neuron neural network array, which has memorized a cross pattern, is depicted in Fig. 3.1.

CROSS	T_{ij} of a_{11}	T_{ij} of a_{13}
Ø Ø 1 Ø Ø	Ø 1 -1 1 1	-1 -1 Ø -1 -1
Ø Ø 1 Ø Ø	1 1 -1 1 1	-1 -1 1 -1 -1
1 1 1 1 1	-1 -1 -1 -1 -1	1 1 1 1 1
Ø Ø 1 Ø Ø	1 1 -1 1 1	-1 -1 1 -1 -1
Ø Ø 1 Ø Ø	1 1 -1 1 1	-1 -1 1 -1 -1

Fig. 3.1 Learned Synaptic Weights

The state (S_i) of neuron i remains at ground potential unless it is changed by the potential inputs. Thus, the state (S_i) of neuron i in the network is the summation of all the inputs at a particular time. The state (value of potential charge) of a neuron is as follows

$$S_i = \sum_{j=1}^{N-1} T_{ij}V_j + I_i \qquad i = 1,N \qquad (3.2)$$

where I_i represents an input current. In order to adapt the model for pattern recognition, I_i represents a positive input current to neuron i if neuron i sees a bit of a pattern array placed before the neural network. Thus, that portion of the pattern tries to continuously excite the neuron i to output. From Equation (3.2), the state (S_i) of neuron i can be positive, negative, or zero potential magnitude. The positive or negative nature of the potential inputs is determined by the learned synaptic weights, T_{ij}, between each neuron. The magnitude of the potential state of neuron i is determined primarily by the magnitude of T_{ij} with neurons j and the number of neurons j that have been excited to output.

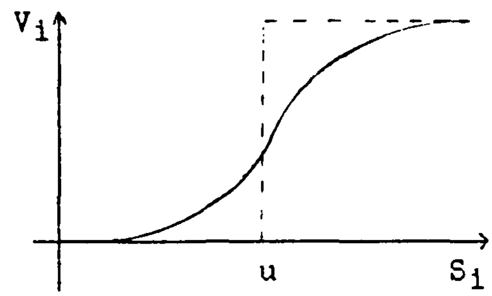


Fig. 3.2 Neuron Output Vs. State Potential

As depicted in Fig. 3.2, the relationship between the neural firing rate output (V_i) and the state potential (S_i) is analog in nature. It is represented by a sigmoid function. Once the biological neural state potential (S_i) has risen above the threshold potential (u_i), fewer inputs are required to maintain a high firing rate. For modeling purposes, the neuron can be thought of as a two-state system. It is either firing or not firing. The rapid rise in response can be approximated by a unit step, and the two-state neuron model can be represented by

$$V_i = 1 \quad \text{if} \sum_j T_{ij}V_j + I_i > u_i \qquad (3.3)$$

$$V_i = 0 \quad \text{if} \sum_j T_{ij}V_j + I_i < u_i \qquad (3.4)$$

where u_i represents the threshold potential and is equal to zero.

The Hopfield discrete neural network model uses the principle of content addressable memory in recalling storage of memory. Since an entire learned pattern is stored in the T_{ij} matrix between interconnected neurons, then the model should converge to this memorized pattern when only a portion of the learned pattern is placed before the network. Thus, the memory is addressable by content and not location. More than one pattern can be learned and stored in memory by the network. However, there is a limit to the number of patterns that a constant size neural network can learn. Since learned patterns of one shape represent noise to learned patterns of another shape, there is a point at which the neural network has a degradation in its ability to recall patterns stored in memory. This is similar to the inability of the human brain to recall previously learned detailed material due to information overload. When the learned synaptic weight matrix, T_{ij}, has reached this saturation point, the neural network will converge to a shape which does not closely resemble any of the patterns stored in memory. Based on the general results of previous experimental data, a general rule [12] that defines this saturation point is

$$k = 0.15\ N \tag{3.5}$$

where k represents the maximum number of patterns learned and N represents the number of neurons in the neural network.

In addition to the assumptions previously mentioned in the derivation of the algorithm for the discrete neural network model, other simplifications to the operation of the biological neuron were made. The model assumes that there is no time loss in transmission from the output of one neuron to the input of another. The current state potential (S_i) of a neuron is based on the summation of its current inputs. Thus, the neurons previous state potentials do not directly affect the current state potentials. Like the biological neural network, the model can assume random or asynchronous neural updates during which time a neuron readjusts the firing rate (output) according to the current state potential. However, for simplification in the computer simulation of this thesis, neurons in the model network update outputs simultaneously after each iteration.

3.2 The Hopfield Continuous Neural Network Model

In 1986, J.J. Hopfield transformed his discrete neural network model into the continuous form. The change of the continuous time domain state potential $S_i(t)$, of a neuron in the network, is described by the differential equation

$$C_i \frac{dS_i(t)}{dt} = \sum_{j=1}^{N} T_{ij}V_j + I_i - \frac{S_i(t)}{R_i} \qquad i = 1,N \qquad (3.6)$$

where C_i represents the input capacitance to neuron i and R_i represents the input resistance to neuron i. Equation (3.6) can easily be transformed to

$$\frac{dS_i(t)}{dt} = \sum_{\substack{j=1 \\ i=1}}^{N} T_{ij}V_j = I_i - (1/R_iC_i)S_i(t) \qquad (3.7)$$

where $1/R_iC_i$ represents the time constant for the refractory period a neuron experiences immediately after firing. The impulse response to Equation (3.7) is

$$h(t) = e^{-(1/RC)t} \quad t \geq 0 \qquad (3.8)$$
$$= 0 \quad t < 0$$

and is depicted in Fig. 3.3.

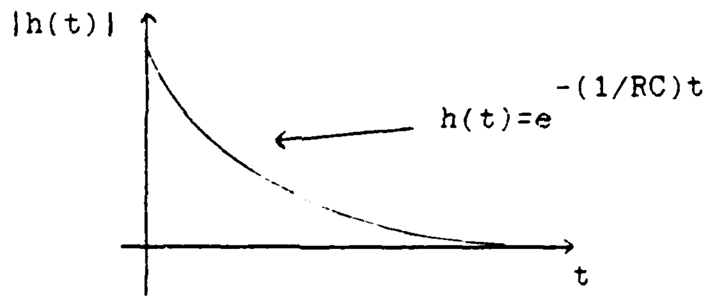


Fig. 3.3 Impulse Response h(t) Vs. Time

This indicates that 1/RC is also the damping factor for the response of the continuous differential Equation (3.7) which represents the mechanics of a neural change of state. The effect on the operation of the continuous neural network model, by changing values of the 1/RC time constant, will be examined in the computer simulation of the continuous model.

To convert Equation (3.7) from a differential equation (continuous time domain) to a difference equation (discrete time domain), the Euler Backward Formula is employed with the period T=1.

$$\frac{S_i(k) - S_i(k-1)}{T = 1} = \sum_{\substack{j=1 \\ i=1}}^{N} T_{ij} V_j(k) + I_i - \frac{S_i(k)}{RC} \qquad (3.9)$$

Combining terms yields the difference equation

$$S_i(k) = \frac{RC}{RC+1} \left[\sum_{\substack{j=1 \\ i=1}}^{N} T_{ij} V_j(k) + I_i + S_i(k-1) \right] \qquad (3.10)$$

where $S_i(k)$ represents the current state potential of neuron i and $S_i(k-1)$ represents the previous state potential. Thus, the current state potential, of a neuron, is based on the current inputs plus the previous state potential and is affected by the 1/RC time constant representing the refractory period. As compared to the discrete model, the continuous neural network model is a closer approximation of the actual operation of a biological neuron because of these features. All other portions of the discrete model algorithm remain the same for the continuous model.

CHAPTER 4

EXPERIMENTAL COMPUTER SIMULATION AND RESULTS

This three-part discussion will examine the computer simulation and list the results for the Hopfield discrete and continuous neural network models adapted for pattern recognition. The program simulating the models was written in Fortran 77 [20] and compiled on the Microsoft Fortran Compiler (V4). The Fortran code listings for the 25 neuron, two patterns learned, discrete and continuous models can be found in Appendix B. The program was executed on an IBM XT compatible personal computer.

4.1 The Experimental Computer Simulation of the Models

Neural network arrays of 25, 49, and 100 neurons were examined for both the discrete and continuous models. Only one neuron occupied each row-column space in the array. These sizes were chosen due to the square array symmetry and the multiple sizes of two. The basic program consisted of two phases. The first phase was a learning and memorization of specified patterns. The second phase was a recognition of the memorized patterns and recognition of distorted cross patterns, with and without noise added. Both learned patterns and distorted cross patterns were varied in length and width. A listing of patterns, by figure and neural network array size, that were used in the computer simulation can be found in Appendix A.

The learning and memorization phase was identical for both the discrete and continuous neural network models. Each network was required to memorize two (cross and square), three (cross, square, and X), and four (cross, square, X, and diamond) patterns in separate exercises. This memorization process was started by a cross-pattern array being placed in front of a neural network of equal array size. Neurons directly opposite any portion of the cross were excited to output ($V_i=1$). All other neurons remained in the zero state ($V_i=0$). Equation (3.1) was then used to determine the learned synaptic weight matrix A(I,J) for each neuron in the network. This same procedure was followed for each additional pattern the network was required to memorize. After the last required pattern had been memorized, a final learned synaptic weight matrix T(I,J,K), for every neuron, was summed from the previously learned synaptic weight matrices that represented the patterns. This memory matrix T(I,J,K) was brought forward into the second phase of the computer simulation.

The pattern recognition phase for the discrete neural network model was conducted in the following manner. Initially, all previously learned patterns, from phase one, were read into appropriate identification arrays for comparison with the converged neural network output pattern array, HOP(I,J). The first learned pattern was then read into the pattern to be recognized array, PAT(I,J). On the first iteration, the only input affecting the neural state potential array, S(I,J), was the input current I_i, from PAT(I,J). All neurons that saw a bit of the pattern would have the state potential raised to a value of +1. All other neurons in the network remained at zero potential, as per Equation (3.2). All neural state potentials, S(I,J), were compared with the threshold potential equal to zero. If the neural state potential was greater than zero, it would be considered in the excited state and would be given an output value of +1 in the neuron output matrix, V(I,J), as per Equations (3.3) and (3.4). The neuron output matrix, V(I,J), was stored in the network output matrix, HOP(I,J). On the second iteration, each neuron in the network was assigned a product matrix, PROD(I,J,K). The product matrix held the values of all inputs to neuron i due to the outputs from the other neurons j times the individual learned synaptic weights T(I,J,K) associated between neuron i and the neurons j. These input products for neuron i were added, and the value stored in matrix SUM(I). The input sum value was added to the input current from matrix PAT(I,J), for the neurons i seeing one bit from the pattern to be recognized. The summatiom was the updated state potential value S(I,J) for each neuron in the network array, as per Equation (3.2). The values of the preevaluated neuron output matrix, V(I,J), were stored into the network output matrix HOP(I,J) for a future test of convergence.

The updated neuron state potential matrix, S(I,J), was evaluated, as per Equations (3.3) and (3.4). The neuron output matrix, V(I,J), was updated accordingly and compared to the preevalauated network output matrix, HOP(I,J). If both matrices matched, the discrete neural network model had converged and produced an output pattern, from the neurons, contained in the network output matrix, HOP(I,J). Convergence also implied that the model was stable under the specific test condition. Normally, the model had not converged on the second iteration, and a third iteration was initiated. At the beginning of the third iteration, the neural state potential matrix S(I,J) was printed. Observations concerning the behavior of this matrix and its effect on the pattern recognition capability of the model will be covered in Chapter 5. The remainder of the third iteration was identical to the second iteration. The discrete model continued to iterate until the comparison test for convergence was passed. It was observed throughout the course of many experiments that if the network model had not converged by the seventh iteration, it oscillated between two totally different neuron output patterns and was considered unstable under that specific test condition. The neuron output patterns were viewed by printing the neural network output matrix, HOP(I,J), at iterations 50, 51, and 52. The neural network model was stopped at iteration 100 and redirected to the next test pattern.

Once the discrete neural network model converged to a particular network output pattern in HOP(I,J), this matrix was compared to each of the memorized pattern matrices learned in the first phase of the computer simulation. The criterion used for a pattern to be recognized was 100% for all bits between the two matrices. This criterion was the simplest to use because the Hopfield discrete and continuous neural networks behaved as an all or nothing type of network. Either the network converged to an exact image of a previously learned pattern, or, the network diverged to a pattern that had no resemblance to any pattern memorized in the first phase of the program. After the comparison was made for pattern recognition, the network output matrix, HOP(I,J), and the number of iterations required for convergence were printed. The discrete neural network model would then read the next previously memorized pattern into matrix, PAT(I,J), and would begin the process of pattern recognition.

Ø Ø 1 Ø Ø	Ø Ø Ø Ø Ø	Ø Ø Ø 1 Ø
Ø Ø 1 Ø Ø	Ø Ø Ø Ø Ø	Ø Ø Ø 1 Ø
1 1 1 1 1	1 1 1 1 1	1 1 1 1 1
Ø Ø 1 Ø Ø	Ø Ø Ø Ø Ø	Ø Ø Ø 1 Ø
Ø Ø 1 Ø Ø	Ø Ø 1 Ø Ø	Ø Ø 1 Ø Ø
CROSS (LEARNED)	CROSS (NN)	CROSS (NC)

Fig. 4.1 Learned, No Noise (NN), Noisy (NC) Crosses

After all previously learned patterns from the first phase had passed through the network, the discrete neural model was tested for its ability to recognize distorted cross patterns. As depicted in Fig. 4.1, two types of distortion were used and tested. The first type of distortion was a top-down-left-right ordered removal of bits from a cross pattern. This type of distorted cross pattern was termed and listed, in the results, as the no noise (NN) cross. No noise cross matrices were presented to the neural network model only during the pattern recognition phase of the program. The neural network model tried to recognize the distorted cross pattern by the same procedure previously mentioned in this section. The desired result was for the model to recognize the no noise cross pattern as the memorized cross pattern learned during phase one. No noise cross patterns of proportions different from the memorized cross patterns were also used to see if the neural network model pattern recognition capability could be enhanced. The neural network model was defined as unstable (US) for a specific test condition if it oscillated on more than 50% of the no noise cross patterns presented. The neural network model was defined as semistable (SS) for a specific test condition if it oscillated on less than 50% of the no noise cross patterns presented. The model was defined as stable (S) if there were no oscillations. In all cases, the no noise cross patterns were presented to the neural network until the network failed to recognize two specific patterns as the memorized cross pattern.

The second type of distortion was a top-down to the right ordered movement of bits in a cross pattern. This type of distorted cross pattern was termed and listed in the results as the noisy (NC) cross. The movement of one bit to the right of its position in the pattern caused one normally excited neuron to be initially turned off and one normally unexcited neuron to be initially turned on. Thus, there was a double effect on the network with every bit moved. The procedure for evaluating the ability of the discrete neural network to recognize the noisy cross pattern was the same procedure explained for the no noise cross pattern.

As previously stated, the learning and memorization phase was identical for both the discrete and continuous neural network models. There were two required additions for the continuous neural network model during the pattern recognition phase. As per Equation (3.10), the time constant 1/RC and the previous state potential PS(I,J) were added into the calculation of the update to the state potential S(I,J). For each specific test condition, the continuous model was evaluated with the time constant 1/RC assigned values equal to zero, one, and 2000.

The codes used in the result tables for the discrete and continuous neural network models are as follows:

1. Column Headings

   C = Cross pattern

   S = Square pattern

   X = X pattern

   D = Diamond pattern

   NN = Maximum number of removed bits for no noise cross in which pattern was still recognized

   NC = Maximum number of moved bits for noisy cross in which pattern was still recognized

   AINN = Average number of iterations to convergence on no noise cross patterns

   AINC = Average number of iterations to convergence on noisy cross patterns

SNN = Stability of model on no noise cross patterns

SNC = Stability of model on noisy cross patterns

2. Row Headings

PAT = Number of patterns memorized in first phase for the test

1/RC=# = Value of time constant used for the test

ORG = ORG pattern used for memorized, no noise cross, and noisy cross patterns

THIN = THIN pattern used for memorized, no noise cross, and noisy cross patterns

MED = MED pattern used for memorized, no noise cross, and noisy cross patterns

PROP = PROP pattern used for memorized, no noise cross, and noisy cross patterns

MED-THIN = MED patterns memorized in first phase; THIN no noise and noisy cross patterns used in recognition phase

PROP-ORG = PROP patterns memorized in first phase; ORG no noise and noisy cross patterns used in recognition phase

PROP-THIN = PROP patterns memorized in first phase; THIN no noise and noisy cross patterns used in recognition phase

PROP-MED = PROP patterns memorized in first phase; MED no noise and noisy cross patterns used in recognition phase

3. Table Entries

Y = Specific memorized pattern in first phase was recognized in second phase

N = Specific memorized pattern in first phase was not recognized in second phase

S = Model stable under the test condition

SS = Model semistable under the test condition

NS = Model not stable under the test condition

4.2 The Hopfield Discrete Neural Network Results

The following section lists the results obtained from the computer simulation of the Hopfield discrete neural network models of 25, 49, and 100 neurons. The pattern matrix arrays used in the test are listed in Appendix A. The Fortran code listing for the discrete neural network model is listed in Appendix B. An explanation of the codes used is located in the previous section.

<u>25 NEURONS DISCRETE</u>

	C	S	X	D	NN	NC	AINN	AINC	SNN	SNC
2 PAT [ORG]	Y	Y			7	3	3.0	N/A	S	SS
3 PAT [ORG]	Y	Y	Y		6	0	3.0	N/A	S	NS
4 PAT [ORG]	Y	Y	Y	Y	6	0	3.5	5.0	S	S

49 NEURONS DISCRETE

	C	S	X	D	NN	NC	AINN	AINC	SNN	SNC
2 PAT [ORG]	Y	Y			7	3	3.Ø	3.Ø	S	SS
3 PAT [ORG]	Y	Y	Y		6	Ø	3.Ø	4.Ø	S	S
4 PAT [ORG]	Y	N	Y	Y	3	Ø	4.Ø	5.Ø	S	S
2 PAT [THIN]	Y	Y			11	3	3.Ø	3.Ø	S	SS
3 PAT [THIN]	N	N	N		Ø	Ø	4.Ø	4.Ø	S	S
4 PAT [THIN]	Y	Y	N	Y	11	Ø	3.5	4.Ø	S	S
2 PAT [PROP]	Y	Y			19	6	3.Ø	3.Ø	S	SS
3 PAT [PROP]	Y	Y	Y		18	5	3.Ø	3.5	S	S
4 PAT [PROP]	Y	Y	Y	N	17	4	3.Ø	4.Ø	S	S
2 PAT [PROP-ORG]	Y	Y				4		3.Ø		SS
3 PAT [PROP-ORG]	Y	Y	Y			3		4.Ø		S
4 PAT [PROP-ORG]	Y	Y	Y	N		3		3.5		S
2 PAT [PROP-THIN]	Y	Y				5		3.Ø		SS
3 PAT [PROP-THIN]	Y	Y	Y			5		3.5		S
4 PAT [PROP-THIN]	Y	Y	Y	N		2		4.Ø		S

100 NEURONS DISCRETE

	C	S	X	D	NN	NC	AINN	AINC	SNN	SNC
2 PAT [THIN]	Y	Y			17	3	3.Ø	3.Ø	S	SS
3 PAT [THIN]	N	N	N		Ø	Ø	4.Ø	4.Ø	S	S
4 PAT [THIN]	N	N	N	N	Ø	Ø	5.Ø	5.Ø	S	S
2 PAT [MED]	Y	Y			31	15	3.Ø	3.Ø	S	SS
3 PAT [MED]	Y	Y	Y		27	2	3.Ø	3.5	S	S
4 PAT [MED]	Y	Y	N	Y	27	2	3.Ø	4.Ø	S	S

100 NEURONS DISCRETE (Cont.)

	C	S	X	D	NN	NC	AINN	AINC	SNN	SNC
2 PAT [PROP]	Y	Y			41	12	3.0	3.0	S	SS
3 PAT [PROP]	Y	Y	Y		38	10	3.0	3.0	S	S
4 PAT [PROP]	N	Y	Y	N	0	0	5.0	4.0	S	S
2 PAT [MED-THIN]	Y	Y				8		3.0		SS
3 PAT [MED-THIN]	Y	Y	Y			1		3.0		SS
4 PAT [MED-THIN]	Y	Y	N	Y		4		3.0		S
2 PAT [PROP-THIN]	Y	Y				8		3.0		SS
3 PAT [PROP-THIN]	Y	Y	Y			7		3.0		S
4 PAT [PROP-THIN]	N	Y	Y	N		0		6.0		S
2 PAT [PROP-MED]	Y	Y				11		3.0		SS
3 PAT [PROP-MED]	Y	Y	Y			9		3.0		S
4 PAT [PROP-MED]	N	Y	Y	N		0		6.0		S

4.3 The Hopfield Continuous Neural Network Results

The following section lists the results obtained from the computer simulation of the Hopfield continuous neural network models of 25, 49, and 100 neurons. The pattern matrix arrays used in the test are listed in Appendix A. The Fortran code listing for the continuous neural network model is listed in Appendix B. An explanation of the codes used is located at the beginning of this chapter.

25 NEURONS CONTINUOUS

		C	S	X	D	NN	NC	AINN	AINC	SNN	SNC
2 PAT	1/RC=Ø	Y	Y			7	3	3.Ø	3.Ø	S	SS
[ORG]	1/RC=1	Y	Y			7	3	3.Ø	3.Ø	S	SS
	1/RC=2ØØØ	Y	Y			7	3	3.Ø	3.Ø	S	SS
3 PAT	1/RC=Ø	Y	Y	Y		6	3	3.Ø	4.Ø	S	S
[ORG]	1/RC=1	Y	Y	Y		6	3	3.Ø	4.Ø	S	S
	1/RC=2ØØØ	Y	Y	Y		6	Ø	3.Ø	4.Ø	S	NS
4 PAT	1/RC=Ø	Y	Y	Y	Y	6	1	3.5	4.Ø	S	S
[ORG]	1/RC=1	Y	Y	Y	Y	6	1	3.5	4.Ø	S	S
	1/RC=2ØØØ	Y	Y	Y	Y	6	Ø	3.5	5.Ø	S	S

49 NEURONS CONTINUOUS

		C	S	X	D	NN	NC	AINN	AINC	SNN	SNC
2 PAT	1/RC=Ø	Y	Y			7	3	3.Ø	3.Ø	S	SS
[ORG]	1/RC=1	Y	Y			7	3	3.Ø	3.Ø	S	SS
	1/RC=2ØØØ	Y	Y			7	3	3.Ø	3.Ø	S	SS
3 PAT	1/RC=Ø	Y	Y	Y		6	Ø	3.Ø	5.Ø	S	S
[ORG]	1/RC=1	Y	Y	Y		6	Ø	3.Ø	4.Ø	S	S
	1/RC=2ØØØ	Y	Y	Y		6	Ø	3.Ø	4.Ø	S	S
4 PAT	1/RC=Ø	Y	N	Y	Y	3	Ø	4.Ø	4.Ø	S	S
[ORG]	1/RC=1	Y	N	Y	Y	3	Ø	4.Ø	5.Ø	S	S
	1/RC=2ØØØ	Y	N	Y	Y	3	Ø	4.Ø	5.Ø	S	S
2 PAT	1/RC=Ø	Y	Y			11	3	3.Ø	3.Ø	S	SS
[THIN]	1/RC=1	Y	Y			11	3	3.Ø	3.Ø	S	SS
	1/RC=2ØØØ	Y	Y			11	3	3.Ø	3.Ø	S	SS

49 NEURONS CONTINUOUS (Cont.)

		C	S	X	D	NN	NC	AINN	AINC	SNN	SNC
3 PAT	1/RC=Ø	N	N	N		Ø	Ø	5.Ø	5.Ø	S	S
[THIN]	1/RC=1	N	N	N		Ø	Ø	5.Ø	5.Ø	S	S
	1/RC=2ØØØ	N	N	N		Ø	Ø	4.Ø	4.Ø	S	S
4 PAT	1/RC=Ø	Y	Y	N	Y	11	Ø	3.5	4.Ø	S	S
[THIN]	1/RC=1	Y	Y	N	Y	11	Ø	3.5	4.5	S	S
	1/RC=2ØØØ	Y	Y	N	Y	11	Ø	3.5	4.Ø	S	S
2 PAT	1/RC=Ø	Y	Y			19	6	3.Ø	3.Ø	S	SS
[PROP]	1/RC=1	Y	Y			19	6	3.Ø	3.Ø	S	SS
	1/RC=2ØØØ	Y	Y			19	6	3.Ø	3.Ø	S	SS
3 PAT	1/RC=Ø	Y	Y	Y		18	5	3.Ø	3.5	S	S
[PROP]	1/RC=1	Y	Y	Y		18	5	3.Ø	3.Ø	S	S
	1/RC=2ØØØ	Y	Y	Y		18	5	3.Ø	3.Ø	S	S
4 PAT	1/RC=Ø	Y	Y	Y	N	17	4	3.Ø	3.Ø	S	S
[PROP]	1/RC=1	Y	Y	Y	N	17	4	3.Ø	3.Ø	S	S
	1/RC=2ØØØ	Y	Y	Y	N	17	4	3.Ø	4.Ø	S	S
2 PAT	1/RC=Ø	Y	Y				4		3.Ø		SS
[PROP-ORG]	1/RC=1	Y	Y				4		3.Ø		SS
	1/RC=2ØØØ	Y	Y				4		3.Ø		SS
3 PAT	1/RC=Ø	Y	Y	Y			3		4.Ø		S
[PROP-ORG]	1/RC=1	Y	Y	Y			3		4.Ø		S
	1/RC=2ØØØ	Y	Y	Y			3		4.Ø		S
4 PAT	1/RC=Ø	Y	Y	Y	N		3		3.5		S
[PROP-ORG]	1/RC=1	Y	Y	Y	N		3		3.5		S
	1/RC=2ØØØ	Y	Y	Y	N		3		3.5		S

49 NEURONS CONTINUOUS (Cont.)

		C	S	X	D	NN	NC	AINN	AINC	SNN	SNC
2 PAT	1/RC=0	Y	Y				5		3.0		SS
[PROP-THIN]	1/RC=1	Y	Y				5		3.0		SS
	1/RC=2000	Y	Y				5		3.0		SS
3 PAT	1/RC=0	Y	Y	Y			5		3.0		S
[PROP-THIN]	1/RC=1	Y	Y	Y			5		3.0		S
	1/RC=2000	Y	Y	Y			5		3.0		S
4 PAT	1/RC=0	Y	Y	Y	N		2		3.0		S
[PROP-THIN]	1/RC=1	Y	Y	Y	N		2		3.0		S
	1/RC=2000	Y	Y	Y	N		2		4.0		S

<u>100 NEURONS CONTINUOUS</u>

		C	S	X	D	NN	NC	AINN	AINC	SNN	SNC
2 PAT	1/RC=0	Y	Y			17	3	3.0	3.0	S	SS
[THIN]	1/RC=1	Y	Y			17	3	3.0	3.0	S	SS
	1/RC=2000	Y	Y			17	3	3.0	3.0	S	SS
3 PAT	1/RC=0	N	N	N		0	0	4.5	4.5	S	S
[THIN]	1/RC=1	N	N	N		0	0	4.0	4.0	S	S
	1/RC=2000	N	N	N		0	0	4.0	4.0	S	S
4 PAT	1/RC=0	N	N	N	N	0	0	4.0	4.0	S	S
[THIN]	1/RC=1	N	N	N	N	0	0	6.0	6.0	S	S
	1/RC=2000	N	N	N	N	0	0	5.0	5.0	S	S
2 PAT	1/RC=0	Y	Y			31	15	3.0	3.0	S	SS
[MED]	1/RC=1	Y	Y			31	15	3.0	3.0	S	SS
	1/RC=2000	Y	Y			31	15	3.0	3.0	S	SS

100 NEURONS CONTINUOUS (Cont.)

		C	S	X	D	NN	NC	AINN	AINC	SNN	SNC
3 PAT	1/RC=0	Y	Y	Y		27	2	3.0	3.0	S	S
[MED]	1/RC=1	Y	Y	Y		27	2	3.0	3.0	S	S
	1/RC=2000	Y	Y	Y		27	2	3.0	3.0	S	S
4 PAT	1/RC=0	Y	Y	N	Y	27	2	3.0	4.5	S	S
[MED]	1/RC=1	Y	Y	N	Y	27	2	3.0	4.5	S	S
	1/RC=2000	Y	Y	N	Y	27	2	3.0	4.5	S	S
2 PAT	1/RC=0	Y	Y			41	12	3.0	3.0	S	SS
[PROP]	1/RC=1	Y	Y			41	12	3.0	3.0	S	SS
	1/RC=2000	Y	Y			41	12	3.0	3.0	S	SS
3 PAT	1/RC=0	Y	Y	Y		38	10	3.0	3.0	S	S
[PROP]	1/RC=1	Y	Y	Y		38	10	3.0	3.0	S	S
	1/RC=2000	Y	Y	Y		38	10	3.0	3.0	S	S
4 PAT	1/RC=0	N	Y	Y	N	0	0	3.5	3.0	S	S
[PROP]	1/RC=1	N	Y	Y	N	0	0	3.5	3.0	S	S
	1/RC=2000	N	Y	Y	N	0	0	3.5	3.0	S	S
2 PAT	1/RC=0	Y	Y				8		3.0		SS
[MED-THIN]	1/RC=1	Y	Y				8		3.0		SS
	1/RC=2000	Y	Y				8		3.0		SS
3 PAT	1/RC=0	Y	Y	Y			1		3.0		S
[MED-THIN]	1/RC=1	Y	Y	Y			1		3.0		S
	1/RC=2000	Y	Y	Y			1		3.0		S
4 PAT	1/RC=0	Y	Y	N	Y		4		3.5		S
[MED-THIN]	1/RC=1	Y	Y	N	Y		4		3.5		S
	1/RC=2000	Y	Y	N	Y		4		3.5		S

100 NEURONS CONTINUOUS (Cont.)

		C	S	X	D	NN	NC	AINN	AINC	SNN	SNC
2 PAT	1/RC=0	Y	Y				8		3.0		SS
[PROP-THIN]	1/RC=1	Y	Y				8		3.0		SS
	1/RC=2000	Y	Y				8		3.0		SS
3 PAT	1/RC=0	Y	Y	Y			7		3.0		S
[PROP-THIN]	1/RC=1	Y	Y	Y			7		3.0		S
	1/RC=2000	Y	Y	Y			7		3.0		S
4 PAT	1/RC=0	N	Y	Y	N		0		4.5		S
[PROP-THIN]	1/RC=1	N	Y	Y	N		0		4.5		S
	1/RC=2000	N	Y	Y	N		0		4.5		S
2 PAT	1/RC=0	Y	Y				11		3.0		SS
[PROP-MED]	1/RC=1	Y	Y				11		3.0		SS
	1/RC=2000	Y	Y				11		3.0		SS
3 PAT	1/RC=0	Y	Y	Y			9		3.0		S
[PROP-MED]	1/RC=1	Y	Y	Y			9		3.0		S
	1/RC=2000	Y	Y	Y			9		3.0		S
4 PAT	1/RC=0	N	Y	Y	N		0		4.5		S
[PROP-MED]	1/RC=1	N	Y	Y	N		0		4.5		S
	1/RC=2000	N	Y	Y	N		0		4.5		S

CHAPTER 5

OBSERVATIONS AND CONCLUSIONS

In this six-part section, results obtained for both the Hopfield discrete and continuous neural network models are discussed. Based on these observations, conclusions are made about the internal properties and external factors that greatly affect the ability of the Hopfield models to memorize and recognize patterns. A comparison is made of the Hopfield discrete and continuous neural network models, and their capabilities are discussed.

5.1 Observations of Discrete Results

The following is a list of observations on the performance of the Hopfield discrete neural network model:

1. The pattern recognition capability of the model is increased, more than 100%, if no noise is added to the pattern to be recognized.

2. Increasing the magnitude (number of bits) of a pattern, for a particular number of neurons, will substantially improve the pattern recognition capability of that set of neurons.

3. Given a particular pattern or patterns learned, increasing the number of neurons of the model will not increase the

pattern recognition capability for the same particular pattern or patterns. In fact, it will decrease the pattern recognition capability as the number of patterns learned is increased.

4. Increasing the magnitude (number of bits) of a learned particular pattern, for a given number of neurons, will substantially improve the performance of the model in recognizing a smaller version of the same pattern. However, this increase will not be as great as increasing, to the same proportion, both the magnitude of the pattern learned and the magnitude of the pattern to be recognized.

5. Increasing the number of neurons and the magnitude (number of bits) of the pattern, proportionally, will improve the performance of the model more than the changes in Item 4 above. However, if the increase in the number of neurons and bits of a pattern is not proportional, then there will be a greater decrease in the performance of the model than the changes in Item 4 above.

6. Increasing the number of patterns learned, for a particular set of neurons, will increase the stability of the model and will increase the number of iterations required to reach stability. However, it will also decrease the pattern recognition capability of the model.

7. Increasing the number of patterns learned will decrease the capability of the model to recognize the original learned patterns. However, models with a smaller number of neurons are more capable of recognizing an increased number of learned patterns which is in contradiction to Equation (3.5).

5.2 Discussion: Discrete Neural Network Model

The following is a discussion about the Hopfield discrete neural network model based on the results obtained:

1. As noise is introduced into a pattern to be recognized, for a particular set of neurons, the number of incorrectly turned-on neural states increases and the pattern recognition capability of the model decreases. This is accomplished through the neural interconnections and the learned synaptic weights. When a noiseless distorted pattern to be recognized is presented to the model, the only neurons in the on (positive) state are those that correctly represent portions of the learned pattern. These neurons will excite (turn on) the neurons in the incorrect off state through the positive learned synaptic weights while keeping the correct unexcited neurons in the off state through the negative learned synaptic weights. As noise is added into a pattern to be recognized, the number of incorrectly turned on neurons is increased. There will be an increased inhibitory effect on all neurons which are correctly in the on state or should be in the on state through the negative learned synaptic weights. The strength (value) of the excited neural states will decrease while the strength (value) of the unexcited neural states will increase. Thus, there is a greater effect forcing the neurons in the on state to the off state while keeping the remainder of the neurons in the off state.

2. As the magnitude (number of bits) of a learned pattern increases for a given number of neurons, the number of neurons

that see and remember a portion of the pattern increases. The pattern recognition capability of the model will be increased. through the neural interconnections and the learned synaptic weights. When a distorted pattern to be recognized is presented to the model, there are more neurons that will correctly identify those portions of the learned pattern which appear. After summing initial neural outputs multiplied by their learned synaptic weights, neurons that are in an incorrect on or off state will be driven to the correct state because of the increased positive or negative value of the summation. A neuron that is in the initial incorrect off (negative) state will be turned on, in future states, by the increased number of correct on neurons, which will produce a larger excitatory (positive) value through their learned synaptic weights. This holds true when the pattern to be recognized is proportional to the pattern learned.

When a smaller pattern in magnitude than the learned pattern is placed before a given number of neurons, a similar reaction occurs. Since the learned pattern is larger in magnitude, there is a greater number of neurons that will recognize a portion of the pattern and will have a positive learned synaptic weight interconnection. Thus, even though the pattern presented to the model is small, it excites a larger group of neurons which will support each other through future state changes. The model will be able to recognize a more distorted version of a particular pattern than it would be able to if the learned pattern were small and equal in size to the presented pattern. However, the pattern recognition capability of the model, in this situation, is not as effective as it would be if the magnitude of the pattern presented was increased in proportion to the learned pattern. The strength (value) of the excited neural states is not as great as it would be in the proportional case. Thus, it would take less noise added, into the observed pattern, to drive the weaker excited neural states into the off state. This same reasoning explains why the pattern recognition capability of the model is increased by proportionally increasing both the number of neurons in the model and the magnitude of the pattern and why the pattern recognition capability of the model is not increased by increasing the number of neurons of the model while maintaining a given pattern at the same magnitude.

3. As the number of learned patterns is increased, the stability and the number of iterations required to reach stability are increased for the model. However, the pattern recognition capability of the model is decreased. As additional patterns are learned by a given number of neurons, the strength (value) of the learned synaptic weights is not a proportional ± 2, ±1, or Ø, as it would be for only one or two patterns learned. There is a large variation in the values of the learned synaptic weights between neurons. The large variation between initial neural states require more iterations to reach stability. Noise added into a given pattern has a greater effect because of the larger variations in value of the learned synaptic weights. The large variations cause wider swings in neural states as the model iterates. There will be

an increase in the number of iterations to stability and a decrease in the pattern recognition capability of the model as neurons, which should be in the excited state, are forced into the off state and vice versa. As more patterns are learned, the bits that represent these patterns are noise to other patterns and distort the internal memory (learned synaptic weights) of the model. Thus, there is a decrease in the number of original learned patterns that the model can recognize. However, the advantage in this situation is that the model is more stable. With small variations of state, certain neurons oscillate between the on and off states because of low-valued incoming learned synaptic weights. These neurons will be forced to the on or off state because of large variations in state forced by large-valued and large-varied incoming learned synaptic weights. The effect can be attributed to the number of patterns learned and to the number of neurons available in a given model. A smaller number of neurons available to the model causes the summation of incoming learned synaptic weights to be less and enhances the opportunity of a neuron to oscillate between the on and off states because of smaller variations in state. Thus, the model is less stable than one with a larger number of neurons. However, this smaller variation in neural state will increase the capability of the model to recognize a greater number of learned patterns. Neurons that should be in an on state will not be driven deep into the off state because of a large-valued summation of learned synaptic weights and vice versa. Thus, more learned patterns are recognized.

5.3 Observations of Continuous Results

The observations for the continuous model are the same for the discrete model with the following additions:

1. The value of the time constant (1/RC) affected the pattern recognition capability of the model with the smallest number of neurons as the number of learned patterns increased. This behavior was not evident in the 49 neuron or 100 neuron models.

2. In the 25 neuron and 49 neuron models, there was a slight increase in the number of iterations required to reach steady state when the models attempted to recognize a noisy cross pattern versus a noiseless cross pattern. This behavior was not evident in the 100 neuron model.

3. As the value of the time constant (1/RC) increased for the 49 neuron model, there was both an increase and decrease in the number of iterations required to reach steady state. This behavior was not evident in the 100 neuron model, and there was only one case in the 25 neuron model.

5.4 Discussion: Continuous Neural Network Model

The following is a discussion about the Hopfield continuous neural network model based on the results obtained:

1. For the continuous model, the noise, magnitude, and number of learned patterns and their relationships with the pattern recognition capability, stability, and number of neurons in the

model are virtually the same as for the discrete model. The only differences are the introduction of the time constant (1/RC) and the new neural state is determined partially by the magnitude of the previous state.

2. The time constant (1/RC) affected the pattern recognition capability of the 25 neuron model. As previously mentioned, the time constant is the damping factor for the impulse response of the continuous differential equation which represents the mechanics of a neural change of state. It also represents the refractory period of a neuron. During this period, any changes to the external stimulus will have no effect on the output or state of a neuron.

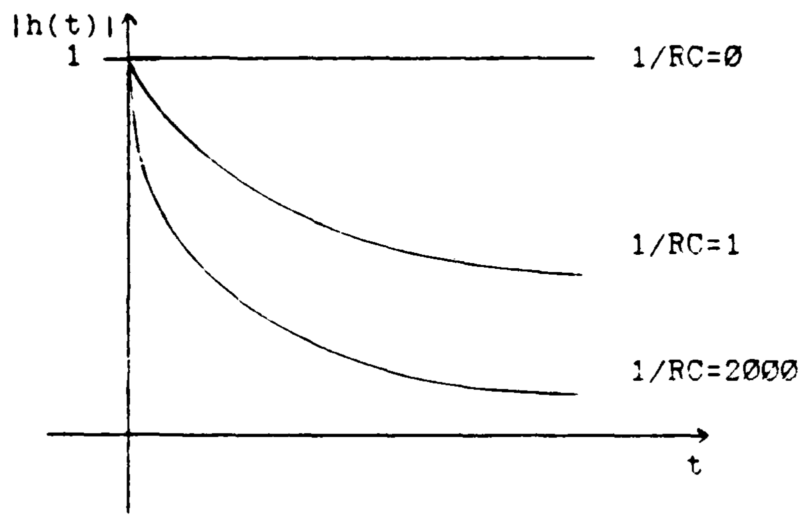


Fig. 5.1 Impulse Response h(t) Vs. Time

As depicted in Fig. 5.1, an increase in the time constant (1/RC) increases the damping factor and shortens the refractory period for a neural change of state. Thus, a neuron will change state more quickly with an increase in the time constant. For 25 neurons, as the time constant increased, the ability to recognize a noisy cross was decreased. Therefore, a longer refractory period for the neurons increased the pattern recognition capability of the 25 neuron model. It is believed that the longer refractory period stops the critical neurons, that are in the correct on state, from being driven into the incorrect off state by neurons that are incorrectly turned on through added noise. Since the 25 neuron model is the smallest model, the magnitude of the state of each neuron has a smaller value. Thus, it is closer to the zero value decision point. Added noise will have a greater effect on a critical neuron jumping into an incorrect state through the negative (inhibitory) value received through its learned synaptic weight with the incorrectly turned on neurons. The longer refractory period, for these critical neurons, delays this change and allows the model to correctly converge on noisier patterns.

3. The time constant (1/RC) did not affect the pattern recognition capability of the 49 neuron and 100 neuron models. The increased number of neurons available to the model causes the summation of incoming learned synaptic weights to increase. Thus, the overall value of the state is increased. As depicted in Fig. 5.2, the magnitude of the neural state is changed by increasing or decreasing the value of the time constant (1/RC). However, due to the increased value of the summation of inputs, the magnitude of the neural state will remain farther away from the zero value

$$|S_i(t)| = \left| \frac{RC}{RC+1} * \sum_{j=1}^{N-1} \right|$$

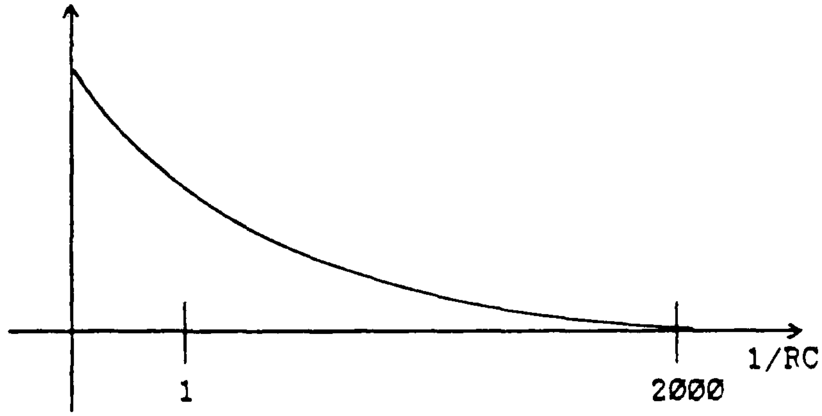


Fig. 5.2 State Value Vs. Time Constant

decision point. The process is enhanced by the fact that the new neural state is also based on the value of the old neural state. With the magnitude of the previous state increasing, the value of the summation increases. Thus, critical neurons are driven more quickly into a stable on or off state that will not be affected as readily by a change in the value of the time constant.

4. It is believed that the time constant did affect the number of iterations required to reach steady state for the 49 neuron model; however, there was no set pattern of increase or decrease to explain how it was affected. Further research is required in this particular area.

5.5 Comparison of Discrete and Continuous Results

The following is a list of comparisons on the performance of the Hopfield discrete and continuous neural network models:

1. The 25 neuron continuous model, with two patterns

learned, acted identically to its discrete counterpart.

2. The 25 neuron continuous model, with three patterns learned, had an increased pattern recognition capability, compared to the discrete counterpart, of three additional bits of the noisy cross and an increase of one iteration to stability when a smaller value of the time constant (1/RC) was applied. With the time constant at its largest value, the continuous model had the same pattern recognition capability but took one additional iteration to reach stability.

3. The 25 neuron continuous model, with four patterns learned, showed the same behavior as the discrete counterpart; except, there was an increased pattern recognition capability of one additional bit of the noisy cross when a smaller value of the time constant (1/RC) was applied.

4. The 49 neuron continuous model acted in the same manner as the discrete counterpart with the only difference being a few sporadic changes in the number of iterations to steady state.

5. The 100 neuron continuous model also acted in the same manner as the discrete counterpart. However, the continuous model, with three MED patterns learned, was more stable than the discrete model while recognizing THIN noisy crosses. The continuous model also, on an average, took one less iteration to stability when it had learned four MED or PROP patterns.

6. All continuous models, independent of the number and type of patterns learned, had the same pattern recognition capability as the discrete models when the time constant (1/RC)

was at the largest value. The only differences were a few sporadic changes in the number of iterations to steady state.

5.6 Discussion: Discrete and Continuous Models

The following is a discussion about the Hopfield discrete and continuous neural network models based on the comparisons of performance:

1. The continuous model is a better representation of the internal operation of a biological neuron (as we know it) and of the interaction between neurons that form a basis of memory and pattern recognition in the brain. The continuous model contains the refractory period that occurs in a biological neuron. The continuous model bases the value of the new neural state partially on the value of the previous state. Upon firing, the neural state potential does not instantaneously drop to the uncharged state of -70 mV which occurs in the discrete model when the neural state instantaneously goes to zero upon firing.

2. Because of the better representation of a biological neuron, the 25 neuron continuous model had an increased pattern recognition capability compared to that for the discrete counterpart. For the same reasons explained in Section 5.4 of the continuous model, the longer refractory period experienced, due to the smaller values for the time constant (1/RC), enhanced the ability of the smaller neuron model to recognize noisier crosses after it had learned three and four patterns. The same model, with

only two patterns learned, did not show this increased ability due to the instability of all models with only two patterns learned. The summation value of the incoming learned synaptic weights multiplied by the outputs of the adjacent neurons was not sufficient to drive the neural states far enough away from the zero value decision region. Thus, critical neurons would oscillate near or would be incorrectly driven across this decision line.

3. As the number of neurons was increased, the pattern recognition capability of both the continuous and discrete models was the same. As explained in Section 5.4 of the continuous model, the time constant (1/RC), in this situation, had virturally no effect on the model due to the increased magnitude of the neural state values. Again in the continuous case, the addition of the previous neural state value helped to counterbalance the reduction in state value experienced by the multiplication of the summation of inputs by the value of RC/RC+1. Without the effect of the time constant (1/RC), the calculations of the continuous and discrete neural states were virtually the same.

4. Because of the better representation of a biological neuron, the 100 neuron continuous model was more stable compared to the discrete counterpart. As in Number 3 above, it was concluded that the addition of the previous neural state value, which increased the magnitude of the neural state values, was large enough to initially drive the critical neurons into a steady state region on either side of the zero decision value region. An example of this was the three-pattern MED-THIN test which showed

the continuous model to be stable while the discrete model was semistable. Since the learned patterns were not proportional, the value of the incoming excitatory (positive) summation was lower in magnitude; therefore, the model had an increased chance to go unstable with critical neurons oscillating across the zero decision value region. The continuous model also had increased stability for it took one less iteration to reach steady state when four PROP learned patterns were used.

5. In all cases, the continuous model had virtually the same results as the discrete model when the largest value of the time constant (1/RC) was used. The discrete model was based on no refractory period occurring in the neuron. Thus, neurons could change their states and outputs instantaneously. The largest value of the time constant (1/RC), in the continuous model, represented the shortest refractory period a neuron could have. It also represented the strongest damping factor that the impulse response to the continuous differential state change equation could have. Thus, with a shortened response to an input, the system was able to recover quickly to its original steady state before the next input to the system. Based on these observations and the results obtained from the continuous model using the large-valued time constant, Hopfield's continuous and discrete models are analogous in their respective domains.

REFERENCES

[1] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas imminent in nervous activity," in Bul. of Math. Biophys., no. 5, 1943, pp. 115-133.

[2] D. O. Hebb, The Organization of Behavior. New York: John Wiley & Sons, 1949.

[3] S. T. Bok, Histonomy of the Cerebral Cortex. Springfield, MA: Charles C. Thomas, 1959.

[4] J. C. Eccles, Neurophysiological Basis of Mind. Oxford: Oxford University Press, 1952.

[5] M. Minsky and S. Papert, Perceptions: An Introduction to Computational Geometry. Cambridge, MA: MIT Press, 1969.

[6] M. Kabrisky, A Proposed Model for Visual Information Processing in the Human Brain. Ph.D. dissertation, Urbana, IL: University of Illinois Press, 1966.

[7] T. Kohonen, Associative Memory: A System Theoretical Approach. New York: Springer, 1977.

[8] T. Kohonen, Self-Organization and Associative Memory. Berlin: Springer-Verlag, 1984.

[9] S. A. Amari, "A mathematical approach to neural systems," in Sys. Neurosci., J. Metzler , Ed. New York: Academic Press, 1977, pp. 67-117.

[10] J. MacGregor and E. R. Lewis, Neural Modeling. New York: Plenum Press, 1977.

[11] L. D. Whittie, "Large scale simulation of brain cortices," in Simulation, vol. 31, no. 4, September 1978, pp. 73-78.

[12] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," in Proc. Natl. Acad. Sci. USA, vol. 79, April 1982, pp. 2554-2558.

[13] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," in Proc. Natl. Acad. Sci. USA, vol. 81, May 1984, pp. 3088-3092.

[14] J. J. Hopfield and D. W. Tank, "Computing with neural circuits: a model," in Sci., vol. 233, August 1986, pp. 625-633.

[15] J. Daly, R. Binggelli, M. Uemura, S. Viglione, and H. Wolf, "Biological processing of visual information," in Biophys. & Cybernetic Sys.: Biol. Proc. of Vis. Inf., D. Maxfield and J. Myles, Eds. Washington DC: Spartan Press, 1965, pp. 53-62

[16] J. A. Anderson and G. E. Hinton, "Models of information processing in the brain," in Par. Models of Associative Memory, J. A. Anderson and G. E. Hinton, Eds. Hillside, NJ: Erlbaum, 1981, pp. 9-46.

[17] D. E. Rumelhart and J. L. McCelland, Eds., Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations. Cambridge, MA: MIT Press, 1986.

[18] R. P. Lippmann, "An introduction to computing with neural nets," in IEEE ASSP, April 1987, pp. 4-21.

[19] D. J. Wallace, "Memory and learning in a class of neural network models," in <u>Proc. of the Workshop on Lattice Gauge Theory, Wuppertal, 1985</u>, B. Bunk and K. H. Mutter, Eds. New York: Plenum Press, 1986.

[20] D. M. Etter, <u>Structured Fortran 77 for Engineers and Scientists</u>. Menlo Park, CA: Benjamin and Cummings, 1987.

APPENDIX A

COMPUTER TEST PATTERNS

This appendix lists the pattern matrices used for the computer simulation of the models. Pattern matrices are listed by figures A.1 - A.7. Each figure represents specific named pattern matrices and the size of neural network in which the matrices were used. The same pattern matrices were used for both the discrete and continuous neural network models.

00100	11111	10001	00100
00100	10001	01010	01010
11111	10001	00100	10001
00100	10001	01010	01010
00100	11111	10001	00100
CROSS	SQUARE	X	DIAMOND

Fig. A.1 25 Neuron ORG Pattern

0000000	0000000	0000000	0000000
0001000	0111110	0100010	0001000
0001000	0100010	0010100	0010100
0111110	0100010	0001000	0100010
0001000	0100010	0010100	0010100
0001000	0111110	0100010	0001000
0000000	0000000	0000000	0000000
CROSS	SQUARE	X	DIAMOND

Fig. A.2 49 Neuron ORG Pattern

```
0001000     1111111     1000001     0001000
0001000     1000001     0100010     0010100
0001000     1000001     0010100     0100010
1111111     1000001     0001000     1000001
0001000     1000001     0010100     0100010
0001000     1000001     0100010     0010100
0001000     1111111     1000001     0001000

 CROSS       SQUARE        X        DIAMOND
```

Fig. A.3 49 Neuron THIN Pattern

```
0011000     1111111     1100011     0001000
0011000     1111111     0110110     0011100
1111111     1100011     0011100     0110110
1111111     1100011     0011100     1100011
0011000     1100011     0011100     0110110
0011000     1111111     0110110     0011100
0011000     1111111     1100011     0001000

 CROSS       SQUARE        X        DIAMOND
```

Fig. A.4 49 Neuron PROP Pattern

```
0000100000     1111111111     1000000001     0000100000
0000100000     1000000001     0100000010     0001010000
0000100000     1000000001     0010000100     0010001000
0000100000     1000000001     0001001000     0100000100
1111111111     1000000001     0000110000     1000000010
0000100000     1000000001     0000110000     1000000001
0000100000     1000000001     0001001000     0100000010
0000100000     1000000001     0010000100     0010000100
0000100000     1000000001     0100000010     0001001000
0000100000     1111111111     1000000001     0000100000

   CROSS         SQUARE           X           DIAMOND
```

Fig. A.5 100 Neuron THIN Pattern

0000110000	1111111111	1000000001	0000110000
0000110000	1111111111	1100000011	0001111000
0000110000	1100000011	0110000110	0011001100
0000110000	1100000011	0011001100	0110000110
1111111111	1100000011	0000110000	1100000011
1111111111	1100000011	0000110000	1100000011
0000110000	1100000011	0011001100	0110000110
0000110000	1100000011	0110000110	0011001100
0000110000	1111111111	1100000011	0001111000
0000110000	1111111111	1000000001	0000110000
CROSS	SQUARE	X	DIAMOND

Fig. A.6 100 Neuron MED Pattern

0000111000	1111111111	1100000011	0001110000
0000111000	1111111111	1110000111	0011111100
0000111000	1111111111	0111001110	0011111100
1111111111	1110000111	0001111000	0111001110
1111111111	1110000111	0001111000	1110000111
1111111111	1110000111	0001111000	1110000111
0000111000	1110000111	0001111000	0111001110
0000111000	1111111111	0011111100	0011111100
0000111000	1111111111	1110000111	0011111100
0000111000	1111111111	1100000011	0001110000
CROSS	SQUARE	X	DIAMOND

Fig. A.7 100 Neuron PROP Pattern

APPENDIX B

FORTRAN CODE LISTING
FOR HOPFIELD MODELS

```
C***********************HOPFIELD DISCRETE MODEL*****************
C************************25 NEURON - 4 PATTERN******************
C
C                        PROGRAM MAIN
C
C
        INTEGER T(25,5,5)
C
        OPEN (UNIT=1, FILE='PATC', STATUS='OLD')
        OPEN (UNIT=2, FILE='PATSQ', STATUS='OLD')
        OPEN (UNIT=3, FILE='PATX', STATUS='OLD')
        OPEN (UNIT=4, FILE='PATD', STATUS='OLD')
        OPEN (UNIT=5, FILE='ADATA', STATUS='OLD')
        OPEN (UNIT=6, FILE='BDATA', STATUS='OLD')
        OPEN (UNIT=7, FILE='CDATA', STATUS='OLD')
        OPEN (UNIT=8, FILE='DDATA', STATUS='OLD')
        OPEN (UNIT=10, FILE='LPT1', STATUS='OLD')
C
        CALL LEARN (T)
C
        CALL PATRECOG (T)
C
        END
C
C
C***********COMPUTING LEARNED SYNAPTIC WEIGHTS PHASE*************
C
        SUBROUTINE LEARN (T)
C
        INTEGER I, J, K, L, M
        INTEGER V1(5,5),V2(5,5),T(25,5,5),A(25,5,5),B(25,5,5)
        INTEGER C(25,5,5),D(25,5,5)
C
        DO 11 I=1,5
          READ(1,*) (V1(I,J), J=1,5)
11      CONTINUE
        REWIND 1
        DO 12 I=1,5
        DO 13 J=1,5
          V2(I,J)=V1(I,J)
13      CONTINUE
12      CONTINUE
        I=0
        DO 14 J=1,5
        DO 15 K=1,5
        I=I+1
          DO 16 L=1,5
          DO 17 M=1,5
            A(I,L,M)=((2 * V1(J,K)) - 1) * ((2 * V2(L,M)) - 1)
17        CONTINUE
16        CONTINUE
15      CONTINUE
14      CONTINUE
        I=1
        DO 18 J=1,5
        DO 19 K=1,5
          A(I,J,K)=0
          I=I+1
19      CONTINUE
18      CONTINUE
        DO 20 I=1,5
          READ(2,*) (V1(I,J), J=1,5)
20      CONTINUE
```

```
      REWIND 2
      DO 21 I=1,5
      DO 22 J=1,5
        V2(I,J) = V1(I,J)
22    CONTINUE
21    CONTINUE
      I=Ø
      DO 23 J=1,5
      DO 24 K=1,5
      I=I+1
        DO 25 L=1,5
        DO 26 M=1,5
          B(I,L,M)=((2 * V1(J,K)) - 1) * ((2 * V2(L,M)) - 1)
26      CONTINUE
25      CONTINUE
24    CONTINUE
23    CONTINUE
      I=1
      DO 27 J=1,5
      DO 28 K=1,5
        B(I,J,K)=Ø
        I=I+1
28    CONTINUE
27    CONTINUE
      DO 29 I=1,5
        READ(3,*) (V1(I,J), J=1,5)
29    CONTINUE
      REWIND 3
      DO 3Ø I=1,5
      DO 31 J=1,5
        V2(I,J) + V1(I,J)
31    CONTINUE
3Ø    CONTINUE
      I=Ø
      DO 32 J=1,5
      DO 33 K=1,5
      I=I+1
        DO 34 L=1,5
        DO 35 M=1,5
          C(I,L,M)=((2 * V1(J,K)) - 1) * ((2 * V2(L,M)) - 1)
35      CONTINUE
34      CONTINUE
33    CONTINUE
32    CONTINUE
      I=1
      DO 36 J=1,5
      DO 37 K=1,5
        C(I,J,K)=Ø
        I=I+1
37    CONTINUE
36    CONTINUE
      DO 38 I=1,5
        READ(4,*) (V1(I,J), J=1,5)
38    CONTINUE
      REWIND 4
      DO 39 I=1,5
      DO 4Ø J=1,5
        V2(I,J) = V1(I,J)
4Ø    CONTINUE
39    CONTINUE
      I=Ø
      DO 41 J=1,5
      DO 42 K=1,5
      I=I+1
        DO 43 L=1,5
        DO 44 M=1,5
          D(I,L,M) = ((2 * V1(J,K)) - 1) * ((2 * V2(L,M)) - 1)
44      CONTINUE
43      CONTINUE
42    CONTINUE
41    CONTINUE
```

```
      I=1
      DO 45 J=1,5
      DO 46 K=1,5
        D(I,J,K)=0
        I=I+1
46    CONTINUE
45    CONTINUE
      DO 47 I=1,25
      DO 48 J=1,5
      DO 49 K=1,5
        T(I,J,K) = A(I,J,K) + B(I,J,K) + C(I,J,K) + D(I,J,K)
49    CONTINUE
48    CONTINUE
47    CONTINUE
      RETURN
      END
C
C
C***************PATTERN RECOGNITION PHASE***************************
C
      SUBROUTINE PATRECOG (T)
C
      INTEGER A, B, D, E, F, G, I, J, K, L, M
      INTEGER T(25,5,5),PROD(25,5,5),PATCROSS(5,5),PAT(5,5)
      INTEGER PATSQ(5,5),PATX(5,5),HOP(5,5),V(5,5),S(5,5)
      INTEGER SUM(25),PATDIA(5,5)
C
      DO 10 I=1,5
        READ(1,*) (PATCROSS(I,J), J=1,5)
10    CONTINUE
      REWIND 1
      DO 11 I=1,5
        READ(2,*) (PATSQ(I,J), J=1,5)
11    CONTINUE
      REWIND 2
      DO 12 I=1,5
        READ(3,*) (PATX(I,J), J=1,5)
12    CONTINUE
      REWIND 3
      DO 13 I=1,5
        READ(4,*) (PATDIA(I,J), J=1,5)
13    CONTINUE
      REWIND 4
C
      DO 999 A=1,8
C
      DO 14 I=1,5
        READ((A),*) (PAT(I,J), J=1,5)
14    CONTINUE
      WRITE(10,*)'%%%%%%%%%%%%%%%%%%%%%% INPUT PATTERN %%%%%%%'
      DO 15 I=1,5
        WRITE(10,202) (PAT(I,J), J=1,5)
15    CONTINUE
      WRITE (10,*)'%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%'
      DO 16 I=1,5
      DO 17 J=1,5
        V(I,J)=0
        HOP(I,J)=0
17    CONTINUE
16    CONTINUE
      B=0
C
301   B=B+1
      IF(B.EQ.3) THEN
          WRITE(10,*)' '
          WRITE(10,*)' S(I,J) IS AS FOLLOWS AT ITERATIONS = 3'
          DO 18 I=1,5
            WRITE(10,202) (S(I,J), J=1,5)
18        CONTINUE
        ELSEIF(B.EQ.50) THEN
          WRITE(10,*) 'B=50 ; HOPFIELD OUTPUT IS AS FOLLOWS'
```

```
            DO 19 I=1,5
              WRITE(10,202) (HOP(I,J), J=1,5)
 19         CONTINUE
          ELSEIF(B.EQ.51) THEN
            WRITE(10,*)
            WRITE(10,*) 'B=51 ; HOPFIELD OUTPUT IS AS FOLLOWS'
            DO 20 I=1,5
              WRITE(10,202) (HOP(I,J), J=1,5)
 20         CONTINUE
          ELSEIF(B.EQ.52) THEN
            WRITE(10,*)
            WRITE(10,*) 'B=52 ; HOPFIELD OUTPUT IS AS FOLLOWS'
            DO 21 I=1,5
              WRITE(10,202) (HOP(I,J), J=1,5)
 21         CONTINUE
          ELSEIF(B.EQ.53) THEN
            WRITE(10,*)
            WRITE(10,*) 'B=53 ; HOPFIELD OUTPUT IS AS FOLLOWS'
            DO 22 I=1,5
              WRITE(10,202) (HOP(I,J), J=1,5)
 22         CONTINUE
            WRITE(10,*)'%%%%%%%%%%%%%% GOING TO NEXT PAT %%%%%%%
            GOTO 999
          ELSE
        ENDIF
C
        DO 23 I=1,25
          SUM(I)=0
 23     CONTINUE
        DO 24 I=1,25
        DO 25 J=1,5
        DO 26 K=1,5
          PROD(I,J,K)=(T(I,J,K) * V(J,K))
 26     CONTINUE
 25     CONTINUE
 24     CONTINUE
        DO 27 I=1,25
        DO 28 J=1,5
        DO 29 K=1,5
          SUM(I) = SUM(I) + PROD(I,J,K)
 29     CONTINUE
 28     CONTINUE
 27     CONTINUE
        K=1
        DO 30 I=1,5
        DO 31 J=1,5
          S(I,J) = SUM(K) + PAT(I,J)
          K=K+1
 31     CONTINUE
 30     CONTINUE
        DO 32 I=1,5
        DO 33 J=1,5
          HOP(I,J)=V(I,J)
 33     CONTINUE
 32     CONTINUE
        DO 34 I=1,5
        DO 35 J=1,5
          IF (S(I,J).GT.0) THEN
            V(I,J)=1
          ELSE
            V(I,J)=0
          ENDIF
 35     CONTINUE
 34     CONTINUE
        DO 36 I=1,5
        DO 37 J=1,5
          IF (V(I,J).EQ.HOP(I,J)) THEN
            GOTO 37
          ELSE
            GOTO 301
          ENDIF
 37     CONTINUE
```

```
C 36      CONTINUE
C
          DO 38 I=1,5
          DO 39 J=1,5
            IF (HOP(I,J).EQ.PATCROSS(I,J)) THEN
                GOTO 39
              ELSE
                WRITE(10,*) ' '
                WRITE(10,*) 'CROSS NOT RECOGNIZED'
                GOTO 995
            ENDIF
39        CONTINUE
38        CONTINUE
          WRITE(10,*) ' '
          WRITE(10,*) 'CROSS RECOGNIZED'
C
995       DO 40 D=1,5
          DO 41 E=1,5
            IF(HOP(D,E).EQ.PATSQ(D,E)) THEN
                GOTO 41
              ELSE
                WRITE(10,*) ' SQUARE NOT RECOGNIZED'
                GOTO 996
            ENDIF
41        CONTINUE
40        CONTINUE
          WRITE(10,*) 'SQUARE RECOGNIZED'
C
996       DO 42 F=1,5
          DO 43 G=1,5
            IF(HOP(F,G).EQ.PATX(F,G)) THEN
                GOTO 43
              ELSE
                WRITE(10,*) 'X NOT RECOGNIZED'
                GOTO 997
            ENDIF
43        CONTINUE
42        CONTINUE
          WRITE(10,*) 'X RECOGNIZED'
C
997       DO 44 L=1,5
          DO 45 M=1,5
            IF(HOP(L,M).EQ.PATDIA(L,M)) THEN
                GOTO 45
              ELSE
                WRITE(10,*) 'DIAMOND NOT RECOGNIZED'
                GOTO 998
            ENDIF
45        CONTINUE
44        CONTINUE
          WRITE(10,*) 'DIAMOND RECOGNIZED'
C
998       WRITE(10,*) '%%%%%%%%%%%%%%%%%%% OUTPUT %%%%%%%%%%%%%%%%%%'
          WRITE(10,201) B
201       FORMAT('0','NUMBER OF ITERATIONS TO STEADY STATE =',I3)
          WRITE(10,*) 'HOPFIELD MODEL NEURON OUTPUTS ARE:'
          DO 46 I=1,5
          WRITE(10,202) (HOP(I,J), J=1,5)
202       FORMAT('0',5I2)
46        CONTINUE
C
999       CONTINUE
          RETURN
          END
```

```
C************************HOPFIELD COTINUOUS MODEL**************
C************************25 NEURON - 4 PATTERN***************
C
C                           PROGRAM MAIN
C
        INTEGER T(25,5,5)
C
        OPEN (UNIT=1, FILE='PATC', STATUS='OLD')
        OPEN (UNIT=2, FILE='PATSQ', STATUS='OLD')
        OPEN (UNIT=3, FILE='PATX', STATUS='OLD')
        OPEN (UNIT=4, FILE='PATD', STATUS='OLD')
        OPEN (UNIT=5, FILE='ADATA', STATUS='OLD')
        OPEN (UNIT=6, FILE='BDATA', STATUS='OLD')
        OPEN (UNIT=7, FILE='CDATA', STATUS='OLD')
        OPEN (UNIT=8, FILE='DDATA', STATUS='OLD')
        OPEN (UNIT=10, FILE='LPT1', STATUS='OLD')
C
        CALL LEARN (T)
C
        CALL PATRECOG (T)
C
        END
C
C
C************COMPUTING LEARNED SYNAPTIC WEIGHTS PHASE*************
C
        SUBROUTINE LEARN (T)
C
        INTEGER I, J, K, L, M
        INTEGER V1(5,5),V2(5,5),T(25,5,5),A(25,5,5),B(25,5,5)
        INTEGER C(25,5,5),D(25,5,5)
C
        DO 11 I=1,5
          READ(1,*) (V1(I,J), J=1,5)
11      CONTINUE
        REWIND 1
        DO 12 I=1,5
        DO 13 J=1,5
          V2(I,J)=V1(I,J)
13      CONTINUE
12      CONTINUE
        I=0
        DO 14 J=1,5
        DO 15 K=1,5
        I=I+1
          DO 16 L=1,5
          DO 17 M=1,5
            A(I,L,M)=((2 * V1(J,K)) - 1) * ((2 * V2(L,M)) - 1)
17        CONTINUE
16        CONTINUE
15      CONTINUE
14      CONTINUE
        I=1
        DO 18 J=1,5
        DO 19 K=1,5
          A(I,J,K)=0
          I=I+1
19      CONTINUE
18      CONTINUE
        DO 20 I=1,5
          READ(2,*) (V1(I,J), J=1,5)
20      CONTINUE
        REWIND2
        DO 21 I=1,5
        DO 22 J=1,5
          V2(I,J) = V1(I,J)
22      CONTINUE
21      CONTINUE
        I=0
        DO 23 J=1,5
```

```
      DO 24 K=1,5
      I=I+1
        DO 25 L=1,5
        DO 26 M=1,5
          B(I,L,M)=((2 * V1(J,K)) - 1) * ((2 * V2(L,M)) - 1)
26      CONTINUE
25      CONTINUE
24    CONTINUE
23    CONTINUE
      I=1
      DO 27 J=1,5
      DO 28 K=1,5
        B(I,J,K)=Ø
        I=I+1
28    CONTINUE
27    CONTINUE
      DO 29 I=1,5
        READ(3,*) (V1(I,J), J=1,5)
29    CONTINUE
      REWIND 3
      DO 3Ø I=1,5
      DO 31 J=1,5
        V2(I,J) + V1(I,J)
31    CONTINUE
3Ø    CONTINUE
      I=Ø
      DO 32 J=1,5
      DO 33 K=1,5
      I=I+1
        DO 34 L=1,5
        DO 35 M=1,5
          C(I,L,M)=((2 * V1(J,K)) - 1) * ((2 * V2(L,M)) - 1)
35      CONTINUE
34      CONTINUE
33    CONTINUE
32    CONTINUE
      I=1
      DO 36 J=1,5
      DO 37 K=1,5
        C(I,J,K)=Ø
        I=I+1
37    CONTINUE
36    CONTINUE
      DO 38 I=1,5
        READ(4,*) (V1(I,J), J=1,5)
38    CONTINUE
      REWIND 4
      DO 39 I=1,5
      DO 4Ø J=1,5
        V2(I,J) = V1(I,J)
4Ø    CONTINUE
39    CONTINUE
      I=Ø
      DO 41 J=1,5
      DO 42 K=1,5
      I=I+1
        DO 43 L=1,5
        DO 44 M=1,5
          D(I,L,M) = ((2 * V1(J,K)) - 1) * ((2 * V2(L,M)) - 1)
44      CONTINUE
43      CONTINUE
42    CONTINUE
41    CONTINUE
      I=1
      DO 45 J=1,5
      DO 46 K=1,5
        D(I,J,K)=Ø
        I=I+1
46    CONTINUE
45    CONTINUE
      DO 47 I=1,25
      DO 48 J=1,5
```

```
          DO 49 K=1,5
            T(I,J,K) = A(I,J,K) + B(I,J,K) + C(I,J,K) + D(I,J,K)
 49       CONTINUE
 48       CONTINUE
 47       CONTINUE
          RETURN
          END
C
C
C***************PATTERN RECOGNITION PHASE**************************
C
          SUBROUTINE PATRECOG (T)
C
          INTEGER A, B, D, E, F, G, I, J, K, L, M
          INTEGER T(25,5,5),PROD(25,5,5),PATCROSS(5,5),PAT(5,5)
          INTEGER PATSQ(5,5),PATX(5,5),HOP(5,5),V(5,5),PATDIA(5,5)
          REAL    SUM(25),S(5,5),PS(5,5)
C
          DO 10 I=1,5
            READ(1,*) (PATCROSS(I,J), J=1,5)
 10       CONTINUE
          REWIND 1
          DO 11 I=1,5
            READ(2,*) (PATSQ(I,J), J=1,5)
 11       CONTINUE
          REWIND 2
          DO 12 I=1,5
            READ(3,*) (PATX(I,J), J=1,5)
 12       CONTINUE
          REWIND 3
          DO 13 I=1,5
            READ(4,*) (PATDIA(I,J), J=1,5)
 13       CONTINUE
          REWIND 4
C
          DO 999 A=1,8
C
          DO 14 I=1,5
            READ((A),*) (PAT(I,J), J=1,5)
 14       CONTINUE
          WRITE(10,*)'%%%%%%%%%%%%%%%%%%%% INPUT PATTERN %%%%%%'
          DO 15 I=1,5
            WRITE(10,202) (PAT(I,J), J=1,5)
 15       CONTINUE
          WRITE (10,*)'%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%'
          DO 16 I=1,5
          DO 17 J=1,5
            V(I,J)=0
            HOP(I,J)=0
            PS(I,J)=0
 17       CONTINUE
 16       CONTINUE
          B=0
C
 301      B=B+1
          IF(B.EQ.3) THEN
              WRITE(10,*)' '
              WRITE(10,*)' S(I,J) IS AS FOLLOWS AT ITERATIONS = 3'
              DO 18 I=1,5
                WRITE(10,203) (S(I,J), J=1,5)
 18           CONTINUE
            ELSEIF(B.EQ.50) THEN
              WRITE(10,*) 'B=50 ; HOPFIELD OUTPUT IS AS FOLLOWS'
              DO 19 I=1,5
                WRITE(10,202) (HOP(I,J), J=1,5)
 19           CONTINUE
            ELSEIF(B.EQ.51) THEN
              WRITE(10,*)
              WRITE(10,*) 'B=51 ; HOPFIELD OUTPUT IS AS FOLLOWS'
              DO 20 I=1,5
                WRITE(10,202) (HOP(I,J), J=1,5)
```

```
20          CONTINUE
          ELSEIF(B.EQ.52) THEN
            WRITE(10,*)
            WRITE(10,*) 'B=52 ; HOPFIELD OUTPUT IS AS FOLLOWS'
            DO 21 I=1,5
              WRITE(10,202) (HOP(I,J), J=1,5)
21          CONTINUE
          ELSEIF(B.EQ.53) THEN
            WRITE(10,*)
            WRITE(10,*) 'B=53 ; HOPFIELD OUTPUT IS AS FOLLOWS'
            DO 22 I=1,5
              WRITE(10,202) (HOP(I,J), J=1,5)
22          CONTINUE
            WRITE(10,*)'%%%%%%%%%%%%%% GOING TO NEXT PAT %%%%%%%
            GOTO 999
          ELSE
        ENDIF
C
        DO 23 I=1,25
          SUM(I)=0
23      CONTINUE
        DO 24 I=1,25
        DO 25 J=1,5
        DO 26 K=1,5
          PROD(I,J,K)=(T(I,J,K) * V(J,K))
26      CONTINUE
25      CONTINUE
24      CONTINUE
        DO 27 I=1,25
        DO 28 J=1,5
        DO 29 K=1,5
          SUM(I) =(SUM(I) + REAL(PROD(I,J,K)))
29      CONTINUE
28      CONTINUE
27      CONTINUE
        K=1
        DO 30 I=1,5
        DO 31 J=1,5
          S(I,J) = (RC/RC+1 * (SUM(K) + PS(I,J) + REAL(PAT(I,J))))
          K=K+1
31      CONTINUE
30      CONTINUE
        DO 32 I=1,5
        DO 33 J=1,5
          PS(I,J) = S(I,J)
33      CONTINUE
32      CONTINUE
        DO 34 I=1,5
        DO 35 J=1,5
          HOP(I,J)=V(I,J)
35      CONTINUE
34      CONTINUE
        DO 36 I=1,5
        DO 37 J=1,5
          IF (S(I,J).GT.0) THEN
              V(I,J)=1
            ELSE
              V(I,J)=0
          ENDIF
37      CONTINUE
36      CONTINUE
        DO 38 I=1,5
        DO 39 J=1,5
          IF (V(I,J).EQ.HOP(I,J)) THEN
              GOTO 39
            ELSE
              GOTO 301
          ENDIF
39      CONTINUE
38      CONTINUE
        DO 40 I=1,5
```

```
        DO 41 J=1,5
          IF (HOP(I,J).EQ.PATCROSS(I,J)) THEN
              GOTO 41
            ELSE
              WRITE(1Ø,*) ' '
              WRITE(1Ø,*) 'CROSS NOT RECOGNIZED'
              GOTO 995
          ENDIF
41      CONTINUE
4Ø      CONTINUE
        WRITE(1Ø,*) ' '
        WRITE(1Ø,*) 'CROSS RECOGNIZED'
C
995     DO 42 D=1,5
        DO 43 E=1,5
          IF(HOP(D,E).EQ.PATSQ(D,E)) THEN
              GOTO 43
            ELSE
              WRITE(1Ø,*) ' SQUARE NOT RECOGNIZED'
              GOTO 996
          ENDIF
43      CONTINUE
42      CONTINUE
        WRITE(1Ø,*) 'SQUARE RECOGNIZED'
C
996     DO 44 F=1,5
        DO 45 G=1,5
          IF(HOP(F,G).EQ.PATX(F,G)) THEN
              GOTO 45
            ELSE
              WRITE(1Ø,*) 'X NOT RECOGNIZED'
              GOTO 997
          ENDIF
45      CONTINUE
44      CONTINUE
        WRITE(1Ø,*) 'X RECOGNIZED'
C
997     DO 46 L=1,5
        DO 47 M=1,5
          IF(HOP(L,M).EQ.PATDIA(L,M)) THEN
              GOTO 47
            ELSE
              WRITE(1Ø,*) 'DIAMOND NOT RECOGNIZED'
              GOTO 998
          ENDIF
47      CONTINUE
46      CONTINUE
        WRITE(1Ø,*) 'DIAMOND RECOGNIZED'
C
998     WRITE(1Ø,*) '%%%%%%%%%%%%%%%%%% OUTPUT %%%%%%%%%%%%%%%%%%'
        WRITE(1Ø,2Ø1) B
2Ø1     FORMAT('Ø', 'NUMBER OF ITERATIONS TO STEADY STATE =',I3)
        WRITE(1Ø,*) 'HOPFIELD MODEL NEURON OUTPUTS ARE:'
        DO 48 I=1,5
        WRITE(1Ø,2Ø2) (HOP(I,J), J=1,5)
2Ø2     FORMAT('Ø',5I2)
48      CONTINUE
2Ø3     FORRMAT('Ø',5F7.2)
C
999     CONTINUE
        RETURN
        END
```

END
DATE
FILMED
DTIC
JULY 88