NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

THESIS

HYPERSPECTRAL IMAGERY ANALYSIS
USING NEURAL NETWORK TECHNIQUES

by

Mark M. Gautreaux
June, 1995

Thesis Advisor: R.C.Olsen
Co-Advisor: D. Walters

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE June 1995	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE HYPERSPECTRAL IMAGERY ANALYSIS USING NEURAL NETWORK TECHNIQUES	5. FUNDING NUMBERS
6. AUTHOR Mark M. Gautreaux	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)* Every material has a unique electromagnetic reflectance/emission signature which can be used to identify it. Hyperspectral imagers, by collecting high spectral resolution data, provide the ability to identify these spectral signatures. Utilization and exploitation of hyperspectral data is challenging because of the enormous data volume produced by these imagers. Most current processing and analyzation techniques involve dimensionality reduction, during which some information is lost. This thesis demonstrates the ability of neural networks and the Kohonen Self-Organizing Map to classify hyperspectral data. The possibility of real time processing is addressed.

14. SUBJECT TERMS Hyperspectral Imagery Analysis	15. NUMBER OF PAGES 135
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18 298-102

Approved for public release; distribution is unlimited.

HYPERSPECTRAL IMAGERY ANALYSIS USING NEURAL NETWORK TECHNIQUES

Mark M. Gautreaux
Lieutenant, United States Navy
B.S., Louisiana State University

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN APPLIED PHYSICS

from the

NAVAL POSTGRADUATE SCHOOL
June 1995

Author: _____
Mark M. Gautreaux

Approved by: _____
R.C. Olsen, Thesis Advisor

_____
D. Walters , Co-Advisor

_____ for.
W.B. Colson, Chairman
Department of Physics

ABSTRACT

Every material has a unique electromagnetic reflectance/emission signature which can be used to identify it. Hyperspectral imagers, by collecting high spectral resolution data, provide the ability to identify these spectral signatures. Utilization and exploitation of hyperspectral data is challenging because of the enormous data volume produced by these imagers. Most current processing and analyzation techniques involve dimensionality reduction, during which some information is lost. This thesis demonstrates the ability of neural networks and the Kohonen Self-Organizing Map to classify hyperspectral data. The possibility of real time processing is addressed.

TABLE OF CONTENTS

I. INTRODUCTION 1

II. NEURAL NETWORKS 3

A. HISTORY OF NEURAL NETWORKS 3

B. PRINCIPLES OF NEURAL NETS 10

1. Firing Rules 10

2. Learning Process 12

3. Error-Correction Learning: The Delta Rule 14

C. KOHONEN'S SELF-ORGANIZING MAP 17

1. Motivation 17

2. Biological Foundation 17

3. Network Description 18

a. Step 1 19

b. Step 2 20

c. Step 3 20

d. Step 4 20

e. Step 5 21

f. Step 6 22

g. Step 7 23

h. Step 8 23

III. HYPERSPECTRAL IMAGERY 25

A. OVERVIEW 25

B. SENSORS 26

1. Advanced Airborne Hyperspectral Imaging System (AAHIS) 27

2. Spatially Modulated Imaging Fourier Transform Spectrometer (SMIFTS) 28

a. Fourier Transform Spectroscopy 28
b. FTVHSI Attributes 33
IV. ANALYSIS AND RESULTS 35
A. SIMPLE, TEST SPECTRA 35
1. Two Node Network 36
2. Four Node Network 37
3. Arrays of Larger Size 39
B. SIMPLE IMAGES 40
1. Black and White Image 41
2. Variable Grey Image 41
C. VARIABLE INTENSITIES OF SIMPLE TEST SPECTRA 42
D. REAL DATA 43
1. Camouflage Image 43
a. Sum of Absolute Differences 43
b. Correlation Coefficient 44
2. Operation Desert Radiance Image 45
3. Run Times 46
V. CONCLUSIONS 47
APPENDIX A 49
APPENDIX B 89
REFERENCES 97
INITIAL DISTRIBUTION LIST 99

I. INTRODUCTION

Remote sensing with optical sensors has evolved from panchromatic to multispectral systems, such as LANDSAT, over the last two decades. These systems collect electromagnetic data over a spectral range of interest, typically in the visible and near infrared. Such data have been used for earth resource studies, land use, and agriculture since the 1970's. Such data allow for broad categorization of remotely sensed regions. Higher resolution data offers a promise of more detailed information. Hyperspectral imaging utilizes a set of images each of which covers a spectral band of 10 nanometers or less. This narrow band collection process makes it possible to resolve minerals, types of plants, and the spectra of various "cultural" artifacts.

Inherent with the multiple spectral band data collection process is an increase in the data volume associated with each pixel and, hence, each image. An image produced by AVIRIS (a 224 band instrument), contains over thirty times the data contained in a similar image produced by LANDSAT (a 7 band instrument). This creates a problem for storing and processing. Most current hyperspectral data processing methods begin with some form of dimensionality reduction, such as principle components, and none, as far as the author knows, are done real time. Any time data reduction is done some of the information is lost and non real time processing implies that all data must be stored. A process which utilizes all of the data for material identification and then performs a massive data reduction would be ideal.

Concurrent with the progress in hyperspectral imaging, computer technology and our understanding of the biological processes involved in learning and intelligence have made significant advances. These advancements have allowed neural networks to progress into a mature technology. They are useful in the following situations:

1. Capturing associations or discovering regularities within a set of patterns

2. Where the volume, number of variables or diversity of the data is very great
3. The relationships between variables are vaguely understood
4. The relationships are difficult to describe adequately with conventional approaches. (Westervelt, Krzysik, and Seel, 1994)

Hyperspectral data fit all four of these criterion.

This thesis tries to attack the problem of analyzing hyperspectral images by employing neural networks with the Kohonen Self-Organizing Map, which has been described by Stan Openshaw as being simple, flexible, and capable of handling immense quantities of data (Openshaw, 1994). It is broken down into five chapters and two appendixes. Chapter I, the INTRODUCTION, is followed by Chapter II, NEURAL NETWORKS, which addresses the history of neural networks, presents an introduction to their workings and concludes with a discussion of Teuvo Kohonen's neural network. Chapter III, HYPERSPECTRAL IMAGERY, defines the term and presents a description of the two instruments used to collect the data analyzed in Chapter IV, ANALYSIS AND RESULTS. Chapter V contains the CONCLUSIONS drawn from Chapter IV. Appendix A contains all figures and Appendix B contains an IDL version of Kohonen's network.

II. NEURAL NETWORKS

A. HISTORY OF NEURAL NETWORKS

Since the late 1980's, the field of neural networks has garnered significant attention from both the commercial and research communities. This method of analyzing problems has gained renewed interest as computer processing speed has exploded. This work will outline the history of Neural Networks, describe the basic concepts upon which all Neural Networks are constructed and give a step by step description of the Kohonen Self Organizing Map. In this report the terms 'neural networks' and 'neural computing' are synonymous and are defined as:

> The study of networks of adaptable nodes which, through a process of learning from task examples, store experiential knowledge and make it available for use. (Aleksander and Morton, 1990)

The concept is not new. It owes its origins to the study of the fundamental cell of the living brain: the neuron (Harvey, 1994). An elementary model of a neuron consists of the body of the cell, called the soma, and the axon, which links the neurons together. The point at which the soma and the axon connect is known as the synapse or synaptic connection. When properly stimulated by the synaptic connections, the soma fires, sending out a small electrical pulse, which travels down the axon to other neurons (McCulloch and Pitts, 1943). In a rudimentary sense, the brain learns by setting a criterion for the soma to fire and adjusting the "weights" of the synaptic connections through experience. Both the criterion and the weights are adjusted continuously. In 1943 neurophysiologist Warren McCulloch and logician Walter Pitts published *A Logical Calculus of the Ideas Immanent in Nervous Activity* which described the neural process and developed a simple model of the neuron using variable resistors, which represented the variable synaptic connections, and summing amplifiers, which represented the operation of the neuron body. This model was the product of over five years of research

conducted by a neural modeling community centered at the University of Chicago (Haykin, 1994). The model was adopted by the pioneers of neural computing and provides the basis of most nets being discussed today (Aleksander and Morton, 1990).

In 1949 neurophysiologist Donald Hebb published *The Organization of Behavior* in that an explicit statement of a physiological learning rule for synaptic modification was presented for the first time (Haykin, 1994). He put forward the idea that a group of neurons could reverberate in different patterns, each being related to a different experience (Hebb, 1949). Specifically, Hebb proposed that the connectivity of the brain is continually changing as an organism learns differing functional tasks, and that neural assemblies are created by such changes (Haykin, 1994). He introduced his "postulate of learning", which states that the strength or weight of the synaptic connection between two neurons is increased by the repeated activation of one neuron by the other across that synapse. This laid the seeds for a model of dynamic memory and increased the possible applications of neural computing.

A major milestone was reached by Frank Rosenblatt in 1958 with the creation of the perceptron, a word he coined to refer to a class of simple neuron-like learning networks that were capable of recognizing images. His creation was based on the McCulloch and Pitts model discussed above. Rosenblatt pioneered two techniques of fundamental importance to the study of learning in neural computing: digital computer simulation and formal mathematical analysis (Rumelhart and McClelland, 1986). In his 1962 book *Principles of Neural-dynamics*, Rosenblatt describes what he thought he was doing as follows:

> Perceptrons are not intended to serve as detailed copies of any actual nervous system. They're simplified networks, designed to permit the study of lawful relationships between the organization of a nerve net, the organization of its environment , and the "psychological" performances of which it is capable. Perceptrons might actually correspond to parts of more extended networks and biological systems; in this case, the results obtained will be directly applicable. More likely they represent extreme simplification of the central nervous system

> in which some properties are exaggerated and others suppressed. In this case, successive perturbations and refinements of the system may yield a closer approximation.
>
> The main strength of this approach is that it permits meaningful questions to be asked and answered about particular types of organizations, hypothetical memory mechanisms, and neural models. When exact analytical answers are unobtainable, experimental methods, either with digital simulation or hardware models, are employed. The model is not the terminal result, but a starting point for exploratory analysis of its behavior. (Rosenblatt, 1962)

Rosenblatt's perceptron created much excitement in the world of neural computing and is a fundamental building block for many of today's functioning neural nets.

Progress in neural computing continued throughout the 1960's. Widrow and Hoff introduced the "least mean-square (LMS)" algorithm in 1960 and used it to train an adaptive pattern classification machine (called Adaline for adaptive linear element). This machine was constructed for the purpose of illustrating adaptive behavior and artificial learning (Widrow and Hoff, 1960) Except for the training procedure, it was identical to the perceptron. Two years later, Widrow and his students introduced one of the earliest attempts at a trainable, layered neural network with multiple adaptive elements. They called their creation Madaline (multiple-Adaline) (Haykin, 1994). As the decade progressed it became apparent that neural nets could be simulated on contemporary computers.

Neural computing seemed to have unlimited potential until 1969 when Marvin Minsky and Seymour Papert published their book *'Perceptrons: An Introduction to Computational Geometry'*. This book questioned the ability of single layer perceptrons to perform some simple image recognition tasks. The book described two problems, called *parity* and *connectedness*, which perceptrons could not perform (Minsky and Papert, 1969). Parity refers to whether an image contains an odd or even number of distinct isolated parts. Connectedness is best defined by an example. The letter *w* is connected where as the letter *i* is not connected. Both of these tasks can be easily solved

by conventional computing methods but Minsky and Papert rigorously showed that in order for a single layer perceptron network to solve these problems the size of the network had to grow as the size of the image grew. This is unacceptable as it would lead to a different architecture for images of different sizes. These and related problems became known as hard learning problems.

Several factors combined to cause advancements in neural computing to slow to a trickle throughout the seventies. First and foremost was the elegance with which Minsky and Papert defined the problems. This made gathering interest and financial support very difficult. Another problem was that technology had yet to produce personal computers and workstations for experimentation. During this time many of the researchers deserted the field in search of more promising areas. This left only a handful of early pioneers continuing to do research in neural computing, most of which were from the psychology and neuroscience communities. From a physics and engineering perspective, we may look back on the 1970s as a "decade of dormancy" for neural networks (Haykin, 1994).

In spite of this lack of interest, one important concept did emerge during the 1970s. This being competitive learning and "self-organizing maps". The idea is to feed several processing nodes with the same inputs. Each time an input is presented, the node which most closely represents that input is declared the winner. It and its neighbors are updated to more closely resemble that input. Eventually, only the winner is updated. The result is a network which classifies the inputs into groups (this discussion is expanded in Section C of this chapter). The first computer demonstration of this process was probably done by von der Malsburg in 1973 (Haykin, 1994). In 1976, Willshaw and von der Malsburg published the first paper on self-organizing maps.

John Hopfield of the California Institute of Technology was responsible for the resurgence of interest, especially in the physics and engineering communities, in the analysis of neural computing. His 1982 paper entitled '*Neural Networks and Physical Systems with Emergent Collective Properties*' drew attention to two properties of

interconnected cells of simple non-linear devices: first, that such a system has stable states which will always be entered if the net is started in similar states and, second, the fact that such states can be created by changing the strength of the interconnections between the cells (Haykin, 1994). Much of the fascination of this paper came from the realization that the properties he identified are inherent to fully interconnected neural networks. These properties are known as associative memory to computer engineers (Aleksander and Morton, 1990). His analysis is derived from the physical concept of energy. This energy is represented by successive firings of the net and is determined by the strength of the connections and the thresholds of the neurons. He offers a proof that the network will operate by minimizing this energy when settling into stable patterns of operation, or 'energy wells' (Aleksander and Morton, 1990). Each successive firing of the net will decrease its overall energy and eventually the net would arrive at a stable minimum at which time the firing pattern would remain constant. He also demonstrated that the connection strengths and thresholds can be calculated so as to create these stable energy states. This class of fully interconnected neural networks with feedback attracted a great deal of attention in the 1980s and in the course of time became known as *Hopfield networks* (Haykin, 1994).

Another significant development occurred during 1982. Teuvo Kohonen published a paper furthering the concept and usefulness of self-organizing maps. His paper has received much greater attention than the earlier work of Willshaw and von der Malsburg (Haykin, 1994). A detailed explanation of self-organization and the Kohonen Network is presented in Section C.

Subsequent investigation of the Hopfield Model revealed that it was possible to train a network. Rules such as the Widrow-Hoff (LMS) procedure could be used to make gradual adjustments to the net parameters until the wells were created. This eliminated the need to solve massive sets of simultaneous equations by conventional methods and implied that neural networks could be used as a tool to solve such sets of equations

(Aleksander and Morton, 1990).

The Hopfield Model did not solve all problems associated with neural computing. It quickly became apparent that false, or local, minima occurred. It also did not solve the problem of hard learning.

In 1986 Geoffrey Hinton and Terry Sejnowski introduced methods of solving both of these problems using a fully interconnected network. In order to solve the local minima problem they introduced 'noise' to the Hopfield model and called their net the Boltzman machine (Aleksander and Morton, 1990). This can be viewed as in Figure 2.1. If enough noise is added to the ball, it will move freely between the two minima. If the level of noise is decreased slowly enough the ball eventually come to rest in the deeper of the two minima. This process is known as simulated annealing. It is based on the proof by Hopfield that the global energy function representing the network can be minimized through a process of asynchronously updating the nodes (Rumelhart, Hinton, and McClelland, 1986). As for the problem of hard learning, Hinton and Sejnowski developed learning rules which allowed hidden nodes in a Boltzman machine to be trained. Boltzman machines of any complexity require enormous processing time and/or speed to be practical.

Minsky and Papert had stated that if a method of training a multiple layer, feed forward, perceptron based network could be found, hard learning could be accomplished. In *Parallel Distributed Processing*, published in 1986, Rummelhart, Hinton and Williams derived a learning algorithm which accomplished this task. A similar generalization of the algorithm was derived independently by Parker in 1985, and a roughly similar learning algorithm was studied by LeCun, also in 1985. This process became known as 'back-propagation' and is now the most popular learning algorithm for the training of multi layer perceptrons (Haykin, 1994).

In 1988, Linsker described a new principle for self-organization in a perceptual network. The principle is designed to preserve maximum information about input

activity patterns, subject to such constraints as synaptic connections and synapse dynamic range. A similar suggestion had been made independently by several vision researchers. However it was Linsker who used abstract concepts rooted in information theory to formulate the *principle of maximum information preservation.* (Haykin, 1994)

In the 1990s, much of the research involving neural networks has centered around applications. This paper will attempt to contribute in this area by applying neural networks to hyperspectral imagery for the purpose of classification. At least two previous attempts have been made. Brown and DeRouin have published several papers including *Comparing Neural Network Classifiers and Feature Selection for Target Detection in Hyperspectral Imagery* (1992), which investigated neural network applicability to target detection and feature selection in an automatic target detection scenario. Also in 1992 Shen and Horblit published *Application of Neural Networks to Hyper-Spectral Image Analysis and Interpretation,* which examined the possibility of classifying AVIRIS data. These attempts are similar in that both use some method of data reduction to decrease the dimensionality (each used the method of principle components at some point), and then apply an error back propagation, feed forward network to the reduced data set. Brown and DeRouin used their network to produce a yes or no determination for each pixel in the image. A "yes" indicating that the pixel of interest belonged to the category of interest, in this case camouflaged targets. Shen and Horblit attempted to design a network which classified the scene according to spectral similarity. Both of these papers demonstrate, to varying degrees, the applicability of neural networks to hyperspectral imagery.

The resurgence of research in neural net technology throughout the 1980s can be directly traced to Hopfield's 1982 paper and was given a boost by the publication of Rummelhart and McLelland's book *Parallel Distributed Processing.* As computer processing speed continues to increase, possible applications of neural computing in the science and engineering communities appear to be limited only by one's imagination.

B. PRINCIPLES OF NEURAL NETS

Typically neural nets are organized in layers, with each layer consisting of n processing nodes. Figure 2.2 illustrates properties which are common to most neural nodes. Each node has an input (x_1-x_n) which is modulated by an adjustable weight (w_1-w_n). The bias input a_o is fixed at a value of 1 and, when combined with its adjustable weight w_o, acts as a threshold for the firing rule. The significance of this will be explained in Section 3.a of this chapter. The node performs a summation *v(k)*, where *k* stands for the node number, on the weight modulated inputs according to:

$$v(k) = x_o w_o + x_1 w_1 + x_2 w_2 + \ldots + x_n w_n \qquad (2.1)$$

The node operates in either the Teach or Use mode. This is determined by the position of a switch. While in the Teach, or training mode, the node will use some type of 'learning rule' which modifies the weights of the connections according to a comparison of the Teaching Input and the Output. This process is illustrated in Figure 2.3.

Training continues for either a defined period of time or until the output is within a specified delta of the Teaching Input. Once training is complete the node is ready to enter the Use mode. In the Use mode the neuron is presented with input patterns. For any input resembling the Teach Input to a specified degree, the node will respond with the 'learned' output.

1. Firing Rules

The word 'firing' is borrowed from the world of biology, where it describes the act of a neuron emitting a series of electrical pulses (Aleksander and Morton, 1990). A 'firing rule', also known as an activation function and represented by φ(k), determines how a node responds to a given input pattern. In the following example, which is slightly modified from Aleksander and Morton, 1990, let a 1 represent a node which is firing and a 0 one which is not firing. Allow the rule to be as follows:

Take a collection of training patterns for a node, some of which cause it to fire (the 1-taught set of patterns) and others which prevent it from doing so (the 0-taught set). Then the patterns not in the collection cause the node to fire if, on comparison, they have more input elements in common with the 'nearest' pattern in the 1-taught set than with the 0-taught set. If there is a tie do not fire.

Given a three input node with a 1-taught set (x_1, x_2, x_3) of 111 and 101, and 0-taught set of 000 and 001, the following truth table is produced prior to the application of the firing rule:

x_1	0	0	0	0	1	1	1	1
x_2	0	0	1	1	0	0	1	1
x_3	0	1	1	0	0	1	0	1
F	0	0	0/1	0/1	0/1	1	0/1	1

Table 2.1 Truth table prior to firing rule application.

Ambiguities exist in all input patterns which were not part of the training set. As an example of the way the firing rule is applied, take pattern 010. It differs from 000 only in the second element, from 001 in the second and third elements, from 101 in all three elements, and from 111 in the first and third element. Pattern 010 is closest to pattern 000 which is part of the 0-taught set. Therefore the node will not fire when presented with the 010 pattern. After the application of the firing rule the truth table becomes:

x_1	0	0	0	0	1	1	1	1
x_2	0	0	1	1	0	0	1	1
x_3	0	1	1	0	0	1	0	1
F	0	0	0	0	0	1	1	1

Table 2.2 Truth table after firing rule application.

The firing rule allows the node to classify input sets, which have not been seen during training and are not identical to the training inputs, by comparing the degree of similarity or distance between sets. This gives the neural net one of its most powerful and intriguing abilities.

Listed below are three commonly used firing rules (Haykin, 1994):

1. Threshold Function.

$$\varphi(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{if } v < 0 \end{cases} \tag{2.2}$$

2. Piecewise-Linear Function.

$$\varphi(v) = \begin{cases} 1 & v \geq 1/2 \\ v & -1/2 \leq v < 1/2 \\ 0 & v < -1/2 \end{cases} \tag{2.3}$$

3. Sigmoid Function.

$$\varphi(v) = \frac{1}{1 + e^{-av}} \tag{2.4}$$

The sigmoid and related functions are by far the most common form of activation function used in the construction of neural networks (Haykin, 1994). It is non-linear, which is required to solve non-linearly separable problems. It offers a continuous range of values from 0 to 1, is smooth, and is differentiable (the importance of this will become apparent during the discussion of error correction learning). A plot of this function is given in Figure 2.4.

2. Learning Process

Fundamental to the performance and effectiveness of a neural net is its ability to

learn. Learning in the context of neural networks has been defined as:

> Learning is the process by which the free parameters of a neural network are adapted through a continuing process of stimulation by the environment in which the network is embedded. The type of learning is determined by the manner in which the parameter changes take place. (Haykin, 1994)

This definition of the learning process implies the following sequence of events:

1. The neural network is stimulated by an environment.
2. The neural network undergoes changes as a result of this stimulation.
3. The neural network responds in a new way to the environment, because of the changes that have occurred in its structure. (Haykin, 1994)

In the above example, the learning process had already taken place. It was used to teach the network to fire appropriately for the training patterns. If, during training, the output of the network had been 1 for the input 000, then the weights associated with each input would have been adjusted such that the output would become 1. This process is demonstrated in Section B.3.a of this chapter.

Many algorithms, or 'learning rules', have been developed to train neural nets. These include Error-Correction, Hebbian, Competitive, and Boltzman learning (Haykin, 1994). All learning rules can be classified as either supervised or unsupervised. These are defined as follows:

> Supervised training requires the pairing of each input vector with a target vector representing the desired output. An input vector is applied and the output of the network is calculated which is compared with the corresponding target vector. The difference (error) is fed back through the network and its weights are changed, according to an algorithm that tends to minimize the error.
>
> Unsupervised learning requires no target vector for the outputs. The training set consists solely of input vectors. The training algorithm modifies network weights to produce output vectors that are consistent. The training process extracts the statistical properties of the training set and groups similar input vectors into classes. Unsupervised training is a for more plausible model of learning in the biological system. (Nasrabadi, 1994)

Supervised learning will be presented through a discussion of the Delta Rule, which is a form of Error-Correction. Unsupervised learning will be examined in Chapter 2 Sec C as part of the discussion on Kohonen Self Organizing Maps.

3. Error-Correction Learning: The Delta Rule

The ability of a neural net to learn lies in its variable connection weights. The initial weights are generated in a random fashion and contain no useful information. An effective method of updating these weights must be incorporated. This was a large stumbling block in the development of neural nets, especially nets containing hidden layers as depicted in Figure 5. Error-correction is one means of accomplishing this.

The Delta Rule, also known as the Widrow-Hoff Rule, was first suggested by Bernard Widrow in 1962 (Widrow, 1962). It has been widely used and is well understood. The concept of the Delta Rule is to take the difference (ϵ) between the threshold of the target output (t) plus a desired overshoot (δ) and the node summation v(k), multiply by a constant (η) known as the 'learning rate', and adjust (Δw) the active weights accordingly. In equation form this looks like:

$$\varepsilon = (t + \delta) - v(k) \qquad (2.5)$$

$$\Delta w_{ki}(n) = \eta \varepsilon_{ki}(n) \qquad (2.6)$$

Where k represents the node of interest, i indicates the weight associated with node k, and n denotes the epoch or learning iteration. The rule should be applied to the desired truth table using the following steps:

1. Select a truth table column.
2. If an error is detected, determine the distance between $v(k)$ and the desired firing value.
3. Adjust the weights that have firing inputs and the threshold to remove a portion of

the error.

4. Go back to step 1, until none of the columns cause errors. (Aleksander and Morton, 1990)

The best way to describe the Delta Rule is through an example. A simple system consisting of a discrete node with two inputs and a bias (a_o which is held constant at 1) is chosen to solve a linearly separable problem. Figure 2.6 depicts this system. The problem is defined as follows: Classify the four points A, B, C, and D into two groups as depicted in Figure 2.7 Table 2.3 depicts the truth table solution illustrated in Figure 2.7.

	A	B	C	D
x_1	0	0	1	1
x_2	0	1	0	1
F	0	1	0	1

Table 2.3 Example problem truth table.

Initial conditions are set as follows:

$$w_o = +0.2 \qquad w_1 = -0.25 \qquad w_2 = 0.2$$

$$\delta = \pm 0.1 \qquad \eta = +0.4$$

The threshold function is used as the firing rule.

$$\varphi(v) = \begin{cases} 1 & if\ v \geq 0 \\ 0 & if\ v < 0 \end{cases} \tag{2.2}$$

Training is started by presenting point A to the node with the desired output of 0. The summation v is calculated according to Eq. 2.4 to be 0.2 which would result in an incorrect output of 1. The error ε is calculated using Eq. 2.5 to be -0.3 and Δw (by Eq. 2.6) to be -0.12. Δw is applied only to w_o because it is the sole active weight. Once Δw is applied, w_o becomes 0.08. The complete process is depicted in Table 2.4.

The final solution yields the following equation:

$$v = -0.176x_1 + 0.274x_2 - 0.017 \tag{2.7}$$

Setting v equal to the threshold (0) and rearranging produces Eq. 2.8.

$$x_1 = 1.56x_2 - 0.10 \tag{2.8}$$

This line is superimposed on the original problem (Figure 2.7) to illustrate that it does in fact separate the two groups appropriately (see Figure 2.8). While this figure makes it obvious that Eq. 2.8 is a solution, it also makes it obvious that it is not a unique solution. Because neural nets present non unique solutions, proper training for each environment is imperative to their success.

EPOCH	INPUT	w_o	w_1	w_2	v	COMMENT
1	A	0.200	-0.250	0.200	0.200	Δw = -0.120
2	A	0.080	-0.250	0.200	0.080	Δw = -0.072
3	A	0.008	-0.250	0.200	0.008	Δw = -0.043
4	A	-0.035	-0.250	0.200	-0.035	Column 1 Satisfied
5	B	-0.035	-0.250	0.200	-0.285	Column 2 Satisfied
6	C	-0.035	-0.250	0.200	0.165	Column 3 Satisfied
7	D	-0.035	-0.250	0.200	-0.085	Δw = 0.074
8	D	0.039	-0.176	0.274	0.137	Column 1 Satisfied
9	A	0.039	-0.176	0.274	0.039	Δw = -0.056
10	A	-0.017	-0.176	0.274	-0.017	Column 1 Satisfied
11	B	-0.017	-0.176	0.274	-0.193	Column 2 Satisfied
12	C	-0.017	-0.176	0.274	0.257	Column 3 Satisfied
13	D	-0.017	-0.176	0.274	0.082	Column 4 Satisfied

Table 2.4 Iterations of network.

C. KOHONEN'S SELF-ORGANIZING MAP

1. Motivation

> Economic representation of data with all their interrelationships is one of the most central problems in information sciences, and such an ability is obviously characteristic of the operation of the brain, too. In thinking, and in the subconscious information processing, there is a general tendency to compress information by forming *reduced representations* of the most relevant facts, without loss of knowledge about their interrelationships. The purpose of intelligent information processing seems in general to be creation of simplified images of the observable world at various levels of abstraction, in relation to a particular subset of received data. (Kohonen, 1987)

From this perspective Teuvo Kohonen has investigated and tried to duplicate the processes that the brain uses to recognize, categorize and identify.

2. Biological Foundation

Evidence generated in the late 1970's indicates that the brain forms neural representations of the various sensory inputs, adjusting these representations over time to form an image of that input. This was followed by the discovery that certain areas of the cortex, when stimulated by a sensory input, produce a response which preserves the topographical order of the input. The sensory responses of the auditory cortex exhibit this behavior. In the auditory cortex there exists a "tonotopic map" in which the spatial order of cell responses correspond to the pitch or acoustic frequencies of tones perceived (Kohonen, 1987). Sounds received by the ear produce a response from a defined area on the cortex. The lowest perceptible acoustic frequencies induce a response at one edge of this area while the highest frequencies induce a response at the opposite edge. The map of acoustic frequency response across this area is perfectly ordered and almost logarithmic with respect to frequency.

The possibility that the representation of knowledge in a particular category of things in general might assume the form of a feature map that is geometrically organized over the corresponding piece of the brain motivated Kohonen to do a series of theoretical

investigations into this process. The results of these investigations (presented in Chapter 5 of *Self-Organization and Associative Memory)* led him to believe that one and the same functional principle might be responsible for self-organization of widely different representations of information. Further investigation led to the realization that many functional properties inherent to self-organizing systems are exhibited by neural networks. His networks attempt to preserve the topological relations of the data while performing a dimensionality reduction of the representation space (Kohonen, 1987). Hyperspectral data, being of many dimensions, appears to be a prime candidate for analysis by such networks.

3. Network Description

Stan Openshaw describes Kohonen's self-organizing map as one of the most interesting of all the competitive neural nets. Its fascination results from the realization that self-organization is a very powerful neural process and that parts of the brain certainly seem to operate in a similar fashion (Openshaw, 1994). It is designed to classify data consisting of many variables. As to its ability to perform this task, Openshaw attributes the following characteristics: (1) simplicity in algorithmic design; (2) ability to handle immense complexity; (3) nice mathematical properties; (4) user induced flexibility; and (5) a plausible degree of biological inspiration. Simplicity and flexibility make this approach very attractive.

Many variations of Kohonen's original design are in existence. Openshaw offers an excellent description of the basic algorithm which, with appropriate modifications, is reproduced here.

A basic Kohonen network algorithm can be outlined in the following steps:

Step 1. Initialization. Define geometry, dimensionality, and size of neuron array.

Step 2. Each neuron has a vector of M weights. Set these weights to some initial value, usually random values.

Step 3. Select a data case that also has variable values and apply any relevant measurement noise to the data.

Step 4. Find whichever neuron is 'nearest' to the data case under consideration.

Step 5. 'Update' the vectors of M weights for all the neurons in the topological neighborhood of the winning neuron, otherwise leave alone.

Step 6. Reduce learning parameter and neighborhood weights by a very, very small amount.

Step 7. Repeat steps 3 to step 6 until convergence, typically a large number of times.

Step 8. Once training is complete, classify data.

This algorithm is computational simple, produces reliable maps, can be modified to accept vectors of any dimensionality and is suitable for parallel processing. The particulars of the above steps are as follows:

a. Step 1

The process is started by defining the geometry of the map. This geometry consists of the number of variables contained in the input data and the size of the array of processing nodes. The processing nodes are similar to that depicted in Figure 2.1. The number of adjustable weights associated with each node is set equal to the number of input variables. Generally the array is sized into either one or two dimensions. Choosing the number of nodes present in the array is based on the expected number of categories the data is to be divided into and experience using the network. Each processing node has the potential to define a category, but need not. As an example take a 5 x 5 array of nodes. When presented with a data set, this array can separate the data into between 1 and 25 categories. The number of categories into which any data set is divided depends on the number of processing nodes and the degree of similarity throughout that data set. Allowing arrays of different sizes to process the same data set will provide the user with valuable information about the workings of the network and the data set. Figure 2.9 depicts the simplest of Kohonen networks, a 2 x 1 array, where

I_n represents an input vector of x_n components. The output of each node equates to whether or not that input vector is a member of its category.

b. Step 2

In order to prevent a bias, the weights are initialized randomly. Most high level programming languages contain some type of random number generator which can be modified to initialize the weights to numbers comparable to the input values. While initialization of the weights to values which are comparable to the inputs is not critical, it will decrease the number of iterations required for training.

c. Step 3

This step begins to separate the neural network from traditional classification schemes. Normally data would be presented in sequential order, each having the same weighting. Some data sets lend themselves to sequential, even weighting analysis (the data sets analyzed in this thesis fall into this category), others are analyzed more accurately by equalizing noise levels and sampling some data cases more often, to reflect the reliability of the data (for applicable examples see Openshaw, 1994). Analysis done in this thesis was based on random selection of the data cases.

d. Step 4

Determining the best matched neuron to represent the input case is fairly simple and extremely important. Various mathematical methods exist for determining 'nearness' or 'similarity', and depending on the form of data measurement and desired results, different measures will be appropriate. Trying different methods offers insight into the network and the data set. The two measures used in this analysis are known as the sum of absolute differences (Eq. 2.9) and the correlation function (Eq. 2.10). In Eq. 2.9 D stands for difference and the smallest D will be declared the winner. In Eq. 2.10 C stands for correlation and the C which is closest to 1 (1 being an exact correlation, 0 being uncorrelated, and -1 being anti-correlated) will be declared the winner. In both Equations x_i is the input vector and w_i is the weight vector. In Eq. 2.10 $x \bullet w$ is the dot,

or inner, product of the two vectors.

$$D=\sum_{i=1}^{n} |x_i - w_i| \tag{2.9}$$

$$C=\frac{x \bullet w}{|x|\,|w|} \tag{2.10}$$

e. Step 5

The adaptive behavior of the self-organizing map results from this part of the algorithm. Once the winner has been declared, the weights associated with that neuron and its neighbors are adjusted such that the similarity between the input and the nodes is increased. The size of the 'neighborhood' of the winning neuron is initially set to some 'distance' and all neurons inside this distance are updated. The neighborhood gradually becomes more and more exclusive over time, such that, at the end of the training cycle, only one neuron is being updated for each input. This process causes the network to gradually become tuned to different inputs in an orderly fashion, almost as if a continuous mapping of the input space was formed over the network (Openshaw, 1994). This results in parts of the network closely resembling the different input patterns. This ordering and smoothing process is extremely subtle.

Two updating algorithms were used in the data analysis portion of this thesis. The first and simplest is known as a block party, and is defined by Kohonen as follows:

$$w_i(t+1)=w_i(t)+\alpha(t)[x_i - w_i(t)] \quad for\ i \in N_c(t) \tag{2.11}$$

else

$$w_i(t+1)=w_i(t) \tag{2.12}$$

where w_i(t) is the weight vector for any neuron i which lies within the neighborhood set

$N_c(t)$ of the winning neuron, $w_i(t+1)$ is the updated weights for this neuron i, x_i is the vector of values for the input data case, and $\alpha(t)$ is a training constant (also known as the 'learning rate') which establishes the size of the update. The training constant is typically started at some value between 0 and 1, and decreases with time. This training process is illustrated in figures 2.10 and 2.11. This portion of the analysis (which will be discussed in more detail in Chapter 4) was set up to demonstrate the network's ability to correctly classify four 'made up' spectra having dramatically differing characteristics. Figure 2.10 depicts the four input spectra. Figure 2.11 is a time lapsed plot of the four sets of weights. Immediately noticeable is that the final, trained weights very closely resemble the inputs. Closer investigation of this plot reveals the approximate time at which the updating process transitioned to single node updating per input. This is evident by the constancy of the weights after the transition.

The second, and slightly more sophisticated, updating algorithm is one which makes the learning rate dependent on the distance (d) from the winning neuron. In effect $\alpha(t)$ becomes $\alpha(d,t)$. A useful, simple method of accomplishing this is a Gaussian function of distance from the winning neuron. Muller and Reinhardt use the following function (Openshaw, 1994):

$$\alpha(d,t)=\exp[-d^2/(2\beta(t))] \tag{2.13}$$

Where $\beta(t)$ is the size of neighborhood at time t.

f. Step 6

In order to achieve stability, the training parameters (ie. learning rate and neighborhood size) must be decreased slowly with time. This process, known as simulated annealing, and its importance, has already been described in section A of this chapter. The algorithm used in this thesis to accomplish this process employed a linear decrease according to iteration number. These Equations are as follows:

$$\alpha(t)=\alpha_o*(1-I_n/I_{max}) \tag{2.14}$$

$$d(t)=d_o*(1-I_n/I_{max}) \tag{2.15}$$

where α_o is the initial training constant, I_n is the iteration number, I_{max} is the total number of iterations, and d_o is the initial neighborhood defining distance. Selecting initial values for α and d_o is best gained through experience. The training constant is normally constrained to $0 < \alpha_o < 1$.

g. Step 7

The optimum number of iterations, which defines the number of training cycles, depends on the number of neurons, the size of the data set, and the noise level of the data. No empirical relationships exist to define this number. Again experience with the network offers the best insight. For small data sets and few neurons (ie. 16 inputs, 50 variables per input, little noise, 4 neurons) 1000 iterations may be enough, while for large data sets and many neurons (ie. 50,000 inputs, 70 variables, some noise, 50 neurons) 2,000,000 iterations may not be enough.

h. Step 8

Finally, once all training is complete, the inputs are classified. Each input is compared to each neuron according to some measure of similarity, normally the same measure used in training. The input is declared a member of the winning neuron's category.

Appendix A of this paper contains a Kohonen network written in IDL (Interactive Data Language). It is based on the FORTRAN program given in appendix II of Openshaw, 1994. Chapter 4 will analyze the applicability of this network to classifying hyperspectral imagery.

III. HYPERSPECTRAL IMAGERY

This Chapter will begin with a look at hyperspectral imagery, its attributes and impediments, and will conclude with a discussion of the two instruments used to collect the data analyzed in Chapter IV.

A. OVERVIEW

Remote sensing is defined as the acquisition of information about an object without being in physical contact (Elachi, 1987). Information is acquired by detecting and measuring changes that the object imposed on the surrounding environment. Areas of study in which measurable changes occur include acoustics, the electromagnetic spectrum and perturbations of the gravity field. The term 'remote sensing' is most commonly used in connection with electromagnetic techniques of information acquisition (Elachi, 1987). These techniques cover the entire electromagnetic spectrum.

Because all materials reflect, absorb, or emit photons in ways characteristic of their molecular makeup, a high resolution trace of the intensity of the transmitted, reflected or emitted radiation versus wavelength forms a graphical record unique to a given material (Rinker, 1990). Hyperspectral imagery, also known as imaging spectrometry, is a form of remote sensing that attempts to reproduce this unique graphical record. It refers to the imaging of an area or "scene" over a large number of discrete, contiguous spectral bands such that the image contains a complete reflectance spectrum.

Hyperspectral imaging is a follow on to multispectral imaging, the difference being the spectral resolution of the collected data. The primary limitation associated with multispectral sensors is an inability to distinguish between certain materials because of poor spectral resolution. The spectral reflectance and emittance characteristics for surface materials, which are determined by electronic and vibrational energy states within the materials, are usually too highly structured to be resolved at coarse spectral resolutions (Vane, 1985). Many of these identifying features occur over bandwidths on

the order of 20-40 nm. Hyperspectral sensors nominally collect data in wavelength bands of 10 nm or less, thus allowing for the identification of such features and in turn identification of the material. Over the last two decades parallel advancements in optics, microelectronics and computer technology have allowed hyperspectral collection of imagery data to mature to the point where it can provide detailed information in many remote sensing environments.

Figure 3.1 illustrates the potential of hyperspectral imagery to distinguish materials. It is a plot of the percent reflectance vs. wavelength of two fabrics (A and C) and a green leaf (B). It is probable that a broadband instrument would be able to identify fabric C from the green leaf but would have great difficulty with fabric A. The spectral reflectance of Fabric A and the green leaf are very similar, but, when the two spectra are over-laid a noticeable difference is present. This difference could only be distinguished by a narrow band instrument. The value of hyperspectral imaging systems lies in their ability to collect complete reflectance spectrum for each picture element (pixel) in the image. This should allow for accurate identification of each pixel and comparison between pixels, thus producing a more accurate depiction of whatever is being imaged.

Increased resolution does not come without a price. When compared to current broadband multispectral sensor systems, hyperspectral sensors produce on the order of 10 to 30 times the amount of data per pixel. Storing and processing this massive amount of data is difficult. Most of today's attempts at analyzing hyperspectral imagery involve reducing the dimensionality of the data. Such reduction methods inevitably lose some of the information. This paper attempts to develop a method of hyperspectral data analysis that is performed without reduction of the dimensionality of the data.

B. SENSORS

This paper analyzes two data sets, collected using different sensors. A brief examination of these instruments follows.

1. Advanced Airborne Hyperspectral Imaging System (AAHIS)

The AAHIS sensor was originally developed by Science Applications International Corporation (SAIC), San Diego, California. The sensor was upgraded in a cooperative effort with SETS Technology Inc., Honolulu, Hawaii. The basic block diagram depicting the AAHIS major components is shown in Figure 3.2. The AAHIS sensor was designed for high signal-to-noise ratio performance based on it's intended use in maritime applications which involve low reflectance environments (e.g. 5% reflectance). Table 3.1 lists the AAHIS performance characteristics. The table also lists the sensor's signal-to-noise ratio performance over a land-based (20% reflectance) environment.

Sensor Parameters	Nominal Value
Useful Spectral Range (nm)	440-870
Number of Spectral Channels	72
Nominal Bandwidth (nm) * 4 channels summed on chip, 2 in software	12.4
Cross Track IFOV (mradians) * 2 channels summed in software	1.1
Along Track IFOV (mradians)	1.0
Spacial pixel number	190
Swath Width (degrees)	11.4
Signal-to-Noise (5 % reflectance target) * 50 Hz frame rate	200-400
Signal-to-Noise (20 % reflectance target) * 50 Hz frame rate	500-850

Table 3.1 AAHIS Sensor Characteristics

2. Spatially Modulated Imaging Fourier Transform Spectrometer (SMIFTS)

The second instrument used to collect data is a second generation SMIFTS type spectrometer known as FTVHSI (Fourier Transform Visible Hyperspectral Imager). SMIFTS is a cryogenically cooled, imaging , spatially modulated Fourier transform interferometer spectrometer for spatially resolved spectral imaging (Lucey, et al., 1993). It offers several positive characteristics which include a wide field of view, simultaneous measurement of all spectral channels, broad wavelength range and moderate spectral resolution. The FTVHSI was built by the Florida Institute of Technology and offers better spatial resolution, spectral resolution and contrast (Otten, et al., 1995). This section will begin with a discussion of the theory behind Fourier transform spectroscopy and the Michelson interferometer and will conclude with a description of the FTVHSI instrument and it attributes.

a. Fourier Transform Spectroscopy

Fourier transformation of a sampled interference pattern to obtain the spectrum of the input source is the basis for Fourier transform spectroscopy, a decades old discipline which principally utilizes Michelson interferometers to make spectral measurements. All of the principles which apply to Michelson interferometers also apply to SMIFTS. (Lucey, et al., 1993)

The Michelson interferometer operates by splitting an electromagnetic wave into two optical paths, one of which can be adjusted in length. Constructive and destructive interference occurs when the electric fields recombine at the detector. The sequence of intensity measurements for different path distances includes the autocorrelation function of the electric field. The Fourier transform of this autocorrelation function is the spectrum of the source, interferometer and detector system. Figure 3.2 depicts the standard model of a Michelson interferometer, with S representing the input source. A given ray is split in two by the half-silvered mirror; the two halves are reflected at mirrors M_1 and M_2 and then recombined at the half-silvered mirror. Mirrors M_1 and

M_2 are positioned at distances which differ by d. This creates the phase variation. The compensator plate is often inserted to make the optical paths symmetrical, so that each ray passes through the same thickness of glass (Klein, 1970).

The ensuing derivation of formulas for interference spectroscopy closely follows that which is presented in Klein, 1970.

An interference spectrometer is a Michelson interferometer modified to use collimated light as shown in Figure 3.3. An image of the aperture in front of the source S is formed in the focal plane of Lens L_2 at the detector D. The lenses ensure that the light is approximately collimated and, therefore the equation for the phase shift for light of wavelength λ or frequency ν reflected from the two mirrors is given by:

$$\Delta\nu=\frac{4\pi d}{\lambda}=\frac{4\pi\nu d}{c} \tag{3.1}$$

where d is the effective separation of M_1 and M_2.

For a single frequency input ν_1, the time-averaged intensity at the detector can be written as (Klein, 1970):

$$I_d=I_1+I_2+2\sqrt{I_1I_2}\cos\frac{4\pi\nu_1 d}{c} \tag{3.2}$$

letting $x = 2\sqrt{I_1I_2} / (I_1+I_2)$

$$=(I_1+I_2)(1+x\cos\frac{4\pi\nu_1 d}{c}) \tag{3.3}$$

$$= I_o \left(1 + x\cos\frac{4\pi\nu_1 d}{c} \right) \tag{3.4}$$

where I_1 and I_2 are the intensities in each beam at the detector and $I_o = I_1 + I_2$. An ideal beam splitter will give $I_1 = I_2$ and $x=1$.

If this concept is extended to the general case, where the spectral intensity is described by a continuous function $I(\nu)$ and $I(\nu)\ d\nu$ represents the intensity in the beam in the frequency range ν to $\nu + d\nu$, then Eq. 3.4 becomes:

$$I_d(\tau) = \int_0^\infty I(\nu)\ (1 + \cos 2\pi\nu\tau)\ d\nu \tag{3.5}$$

where the time constant τ is given by

$$\tau = \frac{2d}{c} \tag{3.6}$$

This result consists of an average term

$$I_o = \int_0^\infty I(\nu)\ d\nu \tag{3.7}$$

plus an oscillatory term

$$I_{osc} = \int_0^\infty I(\nu)\ \cos\ 2\pi\nu\tau\ d\nu \tag{3.8}$$

and can be written in the form

$$I_d(\tau) = I_o\ [1 + \gamma(\tau)] \tag{3.9}$$

where

$$\gamma(\tau) = \int_0^\infty P(\nu) \cos 2\pi\nu\tau \, d\nu \tag{3.10}$$

is the normalized oscillatory term and where

$$P(\nu) = \frac{I(\nu)}{I_o} \tag{3.11}$$

is the normalized spectral distribution function. $\gamma(\tau)$ is the output of the interferometer and is known as an interferogram. Eq. 3.10 expresses $\gamma(\tau)$ as the Fourier transform of $P(\nu)$; we can then recover P as a function of frequency if we can determine γ as a function of τ for all τ, by Fourier inversion (Klein, 1970).

Much of the literature on Fourier Transform Spectroscopy talks of the relationship between the interferogram and the autocorrelation function. What follows is a derivation given by Klein, 1970, which illustrates this relationship.

Suppose that the beam splitter in the interferometer performs ideally. Then, the detector responds to the square of the electric field $E(t)$ from M_1, and the delayed return from M_2, $E(t-\tau)$. The detector signal is then proportional to the time average of the total field squared and, neglecting constants, is

$$I_d = [\, E(t) + E(t-\tau)]^2 = E(t)^2 + E(t-\tau)^2 + 2[E(t)\, E(t-\tau)] \tag{3.12}$$

The time averages are defined as follows:

$$\langle E(t)^2 \rangle = \frac{1}{T} \int_{t_o - T/2}^{t_o + T/2} E(t)^2 dt \tag{3.13}$$

b. FTVHSI Attributes.

A SMIFTS type instrument employs three major optical subsystems: a Sagnac interferometer that produces the spatially modulated interferogram; a Fourier transform lens that frees the spectral properties of dependence on aperture geometry and allows the wide field of view; and a cylindrical lens that reimages one axis of the input aperture onto the detector array providing the one dimension of imaging. Figure 3.5 is schematic depiction of a SMIFTS type instrument. (Lucey, et al. 1993)

The SAGNAC interferometer is known as a triangle path or common path interferometer. The similarities between it and the Michelson interferometer are obvious. Again, the phase change is induced by displacement of one of the mirrors.

The spectral range of any interferometer is based on the spectral response of the detector array and the transmission/reflection characteristics of the optics. In this case, a detector made of silicon is used to sample the resultant interference pattern. The spectral response range of this detector is between 270 nm and 1140 nm. The instrument's response range is limited to between 450 nm and 1040 nm due to instrument configuration (Otten, et al., 1995).

The following table compares SMIFTS to other hyperspectral instruments.

	Resolution $\lambda/\Delta\lambda$	Wavelength Range	Moving Parts	Simultaneous Acquisition	Throughput
SMIFTS	10^2 - 10^3	Broad	NO	YES	very high
GRATING	10^2 - 10^5	Narrow	NO	YES	low
PRISM	10^2 - 10^3	Narrow	NO	YES	low
Michelson	10 - 10^6	Broad	YES	NO	very high
Electronic Filter	10^2	Narrow	NO	NO	very high
Mechanical Filter	10 - 10^3	Broad	YES	NO	very high
Mask Filter	10^2	Narrow	NO	YES	very high

Table 3.2 Hyperspectral Sensor Technology Characteristics

IV. ANALYSIS AND RESULTS

In analyzing the applicability of the Kohonen Self-Organizing Map to hyperspectral imagery, a systematic increase in data complexity was followed. Several times during the analysis the map's ability to classify hyperspectral data seemed questionable. Understanding, gained at lower levels of data complexity, provided knowledge of what the network was doing, why it was doing it and what changes could be made to correct the problem. What follows is an outline of this process and the results and conclusions drawn from it.

A version of the program used in this analysis is presented in Appendix B. It is written in IDL and based on Openshaw's FORTRAN code (Openshaw, 1994). The code is easily adaptable to almost any computer language. All analysis was done on a Silicon Graphics work station.

A. SIMPLE, TEST SPECTRA

To begin the process, four line spectra were created to represent the spectral reflectivity of four fictitious materials. These spectra, depicted in Figure 2.10, contain fifty data points. Each spectra was purposely created with a large degree of dissimilarity. The idea was that if the network could not discern between these spectra there would be no reason to continue the analysis. Each spectra, which represent a category of material, was then replicated four times so that each of these four categories contained four members for a total of sixteen inputs. The inputs were then fed into networks of varying sizes and dimensions, beginning with small number of nodes and then increasing, in an attempt to correctly categorize the spectra. Noise levels imposed on the spectra, learning rate, neighborhood size and number of iterations were varied to observe their effects on the outcome. Noise was generated by the addition, and in some cases subtraction, of uniform random numbers ranging between 0 and 1 multiplied by a constant to produce

the desired noise level.

1. Two Node Network

The two node network is the smallest non trivial network. The parameters used in each run are presented in tabular form in Table 4.1, where d is the neighborhood size parameter, α is the learning constant, N.C. is the number of categories into which the network divided the inputs, noise level is given in percentage and number correct is subjectively determined by the author. The network consistently divided the inputs into two categories each containing eight members. Figure 2.10 also depicts how this network grouped the inputs. The spectra represented by continuous lines were grouped together as were the spectra pictured as diamonds. Initially, the basis for this grouping scheme was not known (it was latter determined to be based on average spectral intensity).

MATRIX SIZE	ITERATIONS	d	α	N.C.	MEM	NOISE APPLIED	NUMBER CORRECT
2 × 1	1000	.4	.5	2	8	0	16
2 × 1	1000	.4	.4	2	8	10	16
2 × 1	10000	.4	.4	2	8	10	16
2 × 1	10000	.4	.4	2	8	10 to 20	16

Table 4.1 Results of the Two Node Network

During the training process the network attempts to replicate the inputs which are assigned to each node, hence it is interesting and insightful to look at a plot of the final weights associated with each node. Figure 4.1 is such a plot. In this network there are two nodes and fifty weights per node. The x-axis in divided into 100 increments, the first fifty of which belong to node # 1, and so forth (all subsequent plots of this type

follow this format). Since there are four categories and only two nodes, an accurate replication of each category is not possible. The network combines the shape of two of the spectra into a single, 'average' spectra. Anytime too few nodes are used, some type of averaging must take place. The results of the two node network were encouraging enough to proceed to the next logical network size, four nodes.

2. Four Node Network

Knowing that the data set consisted of four categories, high expectations were held for the four node network. If this network could not produce a four group, four member result, then the applicability of the Kohonen network would appear doubtful. Because of the anticipated importance of this network size, hundreds of runs were executed. Table 4.2 is a summary of the most interesting.

RUN #	MATRIX SIZE	ITERATIONS	d	α	N.C.	MEM	NOISE	NUMBER CORRECT
1	4 × 1	10000	.4	.4	3	448	0	12
2	4 × 1	10000	.6	.5	4	4444	0	16
3	4 × 1	10000	.4	.5	3	448	10	8
4	4 × 1	30000	.6	.3	4	4444	10	16
5	2 × 2	10000	.4	.4	3	448	10	8
6	2 × 2	30000	.5	.2	4	4444	10	16
7	4 × 1	10000	.4	.4	4	4444	10 to 20	16
8	1 × 4	10000	.4	.3	4	4444	up to 50	16
9	1 × 4	10000	.4	.4	4	5344	up to 70	15

Table 4.2 Results of the Four Node Network

The first configuration selected was a linear 4 × 1 matrix of nodes. Its first run, run

#1 in Table 4.2, did not produce the expected four groups of four result. This run resulted in three categories with eight members in the third category. Four inputs were miss-classified (miss-classification being defined as placing an input in the wrong category). Figure 4.2 is a plot of the final weights. Inspection of this plot reveals that only two of the input spectra had been replicated accurately. At this point, the assumption was made that the network was stuck in a local minima. Several additional runs were made, each with some modification to the original parameters. The problem was corrected by a simultaneous change to a larger neighborhood size (an increase in the learning parameter from 0.4 to 0.6 was sufficient) and an increase in the learning constant (from 0.4 to 0.5). In terms of the simulated annealing process, this amounted to shaking the network harder and allowing it to get out of the local minima. As can be seen in Figure 4.3, run #2 identified the four spectra correctly.

As has been mentioned previously, the training process is an attempt by each node to take on the characteristics of one category of inputs. Figure 2.11 illustrates this process. It is a time stacked plot of the weights associated with run #2. Two characteristics of the network are readily apparent in this plot:

1) As the energy of the network decreases, it will stabilize in some constant configuration.
2) Each node is capable of accurate replication of an input group.

Once positive results were obtained, the effects of higher levels of noise were examined. Figure 4.4 is a plot of the four spectra with 20% noise added. The network continued to produce an accurate classification up to 50% noise (see runs 7 - 10). Figure 4.5 is a plot of the final weight values of run # 7. Notice that some of the noise remains. Noise levels above 50% consistently caused miss-classification of one or more of the inputs.

The next step was to examine the other two possible four node configurations, 1×4 and 2×2. The 1×4 node array produced results which were not noticeably different

from the 4 × 1. The 2 × 2 array, while again producing similar results, required repeated adjusting of the algorithm. On several runs, for reasons which are yet to be understood, one of the nodes was quickly eliminated from the updating process (see Figure 4.6 weights 150-200). Node number 4 was dropped from the training process and eight inputs were placed into category 3. The network often appeared to be stuck in what has been previously described as a local minima. In all cases, once the appropriate parameters had been manipulated, the network performed a correct classification. Figure 4.7 is the plot of final weights for the same 2 × 2 array. In this case the problem was solved by increasing the number of training iterations from 10000 to 30000 and decreasing the learning constant from 0.4 to 0.2. Often several, and sometimes many, adjustments had to be performed prior to network convergence on the correct classification.

By the end of the four node network examination, a better feel for what is meant by the terms 'local minima' and 'simulated annealing' had been gained and what effects adjusting key parameters, such as learning rate and iterations, had on these terms.

Experience with the four node network produced two contradictory opinions.

1. Proper 'tweaking' of the network could lead to correct classification of spectral data.
2. Any tool which has to be continuously adjusted in order to obtain useful results will have limited practical applications.

With this in mind, testing of higher dimension arrays commenced.

3. Arrays of Larger Size

Arrays of larger size were considered and, as with the previous size arrays, many runs were conducted. An illustrative batch is presented in Table 4.3. As a group these networks performed admirably. With no noise added to the inputs, all configurations correctly classified the data. As noise levels increased, the networks over-classified the data (as opposed to a miss-classification, over-classification is defined as creating a

separate category where one is not desired). Over-classification increases as the number of nodes and noise levels increase, such that a 5 × 5 matrix classified the sixteen inputs, containing 10% to 20% noise, into sixteen separate categories. A positive note was that miss-classifications did not occur. Figure 4.8 depicts the final weights of run #7. The replication process is quite evident.

For these networks increasing the number of iterations above 10,000 had almost no effect on the outcome. Small variations in the other parameters seemed to have little effect either.

RUN #	MATRIX SIZE	ITERATIONS	d	α	N.C.	MEM	NOISE	NUMBER CORRECT
1	4 × 2	10000	.4	.4	4	4444	10	16
2	4 × 2	20000	.4	.4	5	43144	up to 40	15
3	8 × 1	10000	.4	.4	5	44431	10	15
4	8 × 1	10000	.4	.4	4	4444	0	16
5	2 × 4	10000	.4	.4	4	4444	10	16
6	4 × 4	10000	.4	.4	4	4444	0	16
7	4 × 4	10000	.4	.4	10	----	10	?
8	5 × 5	10000	.4	.2	16	1	10 to 20	?

Table 4.3 Results of Larger Size Arrays

B. SIMPLE IMAGES

Having obtained some confidence in the network's ability to classify spectra, the next step was to determine its ability to work with an image. For this purpose two simple variable images were created.

1. Black and White Image

The first test image, Image 1, is depicted in Figure 4.9. It consists of four small light grey squares imposed on a large black square background. The image was created by setting the background equal to a small number when compared to the inner square fields (in this case 6 and 200 respectively). The image is displayed in a grey scale ranging from 0 for black to 255 for white. This image was then read into networks consisting of between two and sixteen nodes. Each run produced the same result, an accurate classification of the image into two groups (see Figure 4.10). Changes in iterations, learning rate and neighborhood size had no effect on the output.

Noise was then added to the data which changed the image to look like Figure 4.11. The two node array correctly classified the image, for noise levels up to 50%, into two categories (see Figure 4.12). This result was expected, given that the network was forcing the image into two categories. At noise levels above 50% members of the inner squares could end up being closer in magnitude to the black exterior. Figure 4.13 exhibits the result of 55% noise added to the upper right square. As before, when more nodes were used to classify a noisy image, an over classification occurred.

2. Variable Grey Image

Image 2, depicted in Figure 4.14, closely resembles Image 1. It was created by changing the values in three of the black squares to 50, 100, and 150, respectively. Classification of Image 2 by networks consisting of five or more nodes correctly categorized the image into five separate groups. A five node network output appears in Figure 4.15. Close analysis of these two figures reveals that the inner squares have not kept their same relative brightness. This reveals that the network is assigning values to groups according to their relative placement in the nodal array and not according to any specific characteristic of the inputs associated with that group.

A two node array produced the image depicted in Figure 4.16. Adding noise to Image 2 produced results consistent with those described in the analysis of Image 1.

C. VARIABLE INTENSITIES OF SIMPLE TEST SPECTRA

Spectral reflectivity curves, while maintaining a given shape, vary as the intensity of the illuminating source changes. It is important that a classification scheme be able to group two identical objects, one in shade and one in sunlight, as a member of the same category. In order to test the Kohonen network's ability to classify spectra according to curve shape and not intensity, the spectra described in part A of this chapter have been modified such that two members of each group exist, with each being offset by some constant from the other. Figure 4.17 plots the resulting spectra. Ideal classification would result in four categories of two members, each being of the same shape.

Many network configurations were put forward in an attempt to classify this data set correctly, with none yielding the desired results. Furthermore, manipulation of iterations, learning rate, and neighborhood size proved to be futile. Unless the offset was comparatively small, the network grouped the spectra according to an average intensity. A change in how the winning neuron was selected was in order.

All runs up to this point in the analysis had been conducted using the sum of absolute differences (Eq. 2.9) as the method of determining the most similar neuron. It seems to reason that adding up the absolute distances between each corresponding point would equate to a comparison of average intensity. A method of determining curve shape similarity was needed.

The degree of similarity of any two vectors in space is determined by taking the dot product between such vectors. With this in mind, Eq. 2.10 was selected for the described purpose. The results of this change were impressive. Not only were the spectra properly classified according to curve shape, but the time to complete a training iteration and classify inputs was decreased dramatically. With these results in hand, real data were selected for analysis.

D. REAL DATA

1. Camouflage Image

The first set of data set was taken by Bruce Raffert and Glen Sellars as part of work being done by Kestrel Corp. in conjunction with Phillips Laboratory. A SMIFTS type hyperspectral instrument, built by Raffert, was used to collect the spectral reflectivity of a piece of cloth camouflage. The cloth was placed fifty feet from the instrument, draped over a piece of wooden board and leaned against a metal pole. The background consisted of short grass. The image was collected sequentially in 99 vertical swaths, starting at the left edge and ending at the right edge. The data were collected into 100 bandwidths which progress linearly in wavenumber, such that bandwidth number 34 corresponds to 6328 Å (angstroms). The data form what is known as a hypercube with the x and y dimensions corresponding to the spatial, and z being frequency. Figure 4.18 depicts the data in its hypercube form, Figure 4.19 is a plot of band 37.

The objective of this analysis was to determine the network's ability to distinguish between the camouflage and the background. To show that significant differences do exist in the spectral response of the camouflage and the background grass, spectra taken from each are plotted in Figure 4.20. The line with the highest intensity value is the background grass, the middle line is from the light portion of the camouflage and the dashed line is from the dark.

Both of the previously described methods of determining the winning neuron were used (ie. the sum of absolute differences and the dot product). The process and results of each are given below.

a. Sum of Absolute Differences

An ideal classifying scheme would separate the image into three categories, the background, the camouflage, and the pole, and, most importantly, all background pixels would be classified as different from the camouflage. Because the camouflage is made of two different colors, it is unlikely that all pixels in the camouflage would all be

classified the same. A more realistic hope would be to distinguish the camouflage as two different groups, each being separate from the background.

In the pursuit of this goal, the image was read into networks of varying sizes (two to eight nodes). Regardless of the network size, the algorithm classified the image according to what the author believes is intensity. This conclusion is based on the similarity in shape of each set of the final weights. To illustrate this effect, Figure 4.21 is presented. It is a plot of the final weights associated with a 3×1 array. The network appears to be replicating the average spectra of pixels of like intensity. Figure 4.22 shows the categories produced by the same network. Notice that, while the camouflage is distinguishable from the background, many pixels in both the camouflage and the background are classified as members of the same group.

Several attempts were made to preprocess the data in an attempt to get the network to classify by spectral curve shape. These are listed below:

1. Normalization of the maximum intensity value in each pixel's spectra to the overall maximum intensity value of the image.
2. Normalization of the average intensity value in each pixel's spectra to the average intensity value of the entire image.
3. Retransformation of the first three principle components back into frequency space. This acts as a low pass filter.

These attempts produced results that were no better than the initial runs.

b. Correlation Coefficient

Using the correlation coefficient to obtain the 'distance' to the winning neuron worked much better. Again, the image was read into networks of varying sizes, this time with dramatically different results. Figure 4.23 is a plot of the final weights associated with a 3×1 array. By comparing Figures 4.20, 4.21, and 4.23 it is obvious that the network is no longer distinguishing between groups according to intensity, and is now looking at spectral curve shape. Figure 4.24 is the classification produced by this same

network. Notice that very few pixels in the background are classified as being in the same group as the camouflage, and only a small number of camouflage pixels are miss-classified as background pixels. Also note that the camouflage is broken into two distinct categories. The fact that the pole is not placed into a separate category is a problem that is not completely understood. One possibilityis that the pole is a small portion of the image and is not dramatically different spectrally from the darker portion of the camouflage.

2. Operation Desert Radiance Image

The second data set is the result of airborne hyperspectral remote sensing collections conducted at White Sands, New Mexico in October 1994, as part of an experiment called Operation DESERT RADIANCE. The primary objective of this experiment was to determine the ability of remotely sensed data to detect and identify a target under camouflage. The data set used in this analysis was collected by an airborne AAHIS instrument, flown at 2000 feet. In the scene are several target panels and an M-60 tank covered in desert camouflage. The background mostly consists of brown sandy dirt. Small bushes and shrubs are present throughout the scene.

The first objective was to see if the network could identify the tank. Many attempts were made at accomplishing this with no success. In a last ditch effort, a 4 × 2 network was trained on an area of pixels of which most (approximately 75%) involved the tank. The tank was still divided into several different categories. Because the data were taken in the visible wavelengths, and camouflage is designed to inhibit visible detection, this task proved to be too difficult for the network to accomplish. What the network did show was that the spectral signature of the target was not significantly different from its background. In other words, the camouflage did its job. Based on the results obtained in from the first data set, the author believes that, had the data included the near IR wavelengths, the network would have been able to identify the target.

The network did isolate several interesting categories. These are presented in Figures

4.25, 4.26, 4.27, and 4.28. Figure 4.25 shows an instrument calibration panel, which was singled out as a separate category. The author believes everything in Figure 4.26 to be vegetation of some kind, mostly small bushes and shrubs. Figures 4.27 and 4.28 all appear to be members of the target panels, set out as part of the experiment. Lack of adequate ground truth prevents further analysis of these groups.

3. Run Times

The correlation coefficient method of measuring similarity resulted in a much quicker algorithm. Test runs were done just for this comparison. (Typical training for images took much longer, on the order of one to three hours.) Consecutive runs were executed on the first data set, holding all parameters constant. Table 4.4 lists several network configurations and their run times.

Size	Time (sec)	
	Dot Product	Summation
2 × 1	72	206
4 × 1	105	390
4 × 2	176	755

Table 4.4 Run Time Comparison

Another time measurement of interest is the time it takes to categorize an image. For single category classification, looking for one item of interest in an image, the first camouflage image averaged 19 seconds, while the DESERT RADIANCE image averaged 31 seconds. These numbers are significant when trying to determine this process's ability to perform realtime data processing. The DESERT RADIANCE image is 192 pixels wide by 512 pixels long. If each pixel is 10 feet across, the image is then 5120 feet long. A processing time of 31 seconds allows the collecting device to travel at almost 120 miles per hour. This appears to have potential for real time classification (target identification) once a network has been trained.

V. CONCLUSIONS

Hyperspectral imagery was analyzed using the Kohonen Self-Organizing Map neural network. Hyperspectral data was successfully read into the network, organized and categorized.

The first hyperspectral image to be analyzed was one consisting of a piece of camouflage cloth surrounded by a grassy background. The data contained 100 spectral bands that extended from approximately 450 nm to 1040 nm. The network, using the correlation coefficient to determine the nearest neuron, was able to accurately distinguish between the camouflage and the background. The absolute difference algorithm converged on intensity.

The second data set was taken as part of operation DESERT RADIANCE. It consisted of an arid landscape separated into 70 spectral bands ranging from 440 nm to 870 nm. Placed in the image were several target panels, an M-60 tank covered by camouflage and a calibration panel. The data were collected only in the visible frequencies. The network did not identify the tank. However, it did single out other objects as distinct groups in the scene. Major Mat Fay analyzed this same scene using coordinate rotation methods with some success at isolating the tank (Fay, 1995). The contrast between the two approaches is instructive. The neural net technique is effective at identifying major categories in a hyperspectral image. Rotations in spectral space seem to offer more promise in isolating minority elements.

This analysis has led the author to believe that the best approach to using this neural net technique is to:

1. Take a representative segment of whatever environment is to be classified containing the item or items of interest.
2. Train the network with this data set.
3. Determine if the item of interest is singled out as a separate category.

If so, then a pretrained network operating in similar environments should be able to identify the item of interest. Real time processing is a definite possibility.

In order to demonstrate the applicability of this network to conclusively analysis of hyperspectral imagery, several data sets with well defined, highly accurate ground truths, are needed.

APPENDIX A

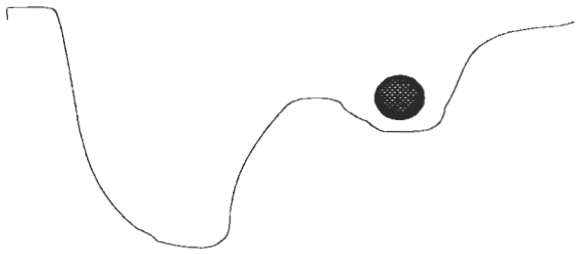

Figure 2.1 Energy surface representation of neural network.

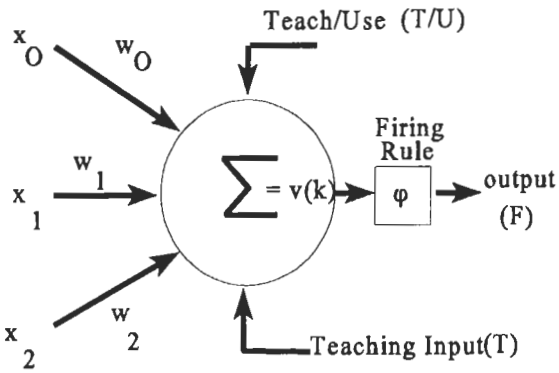

Figure 2.2 Typical neural node.

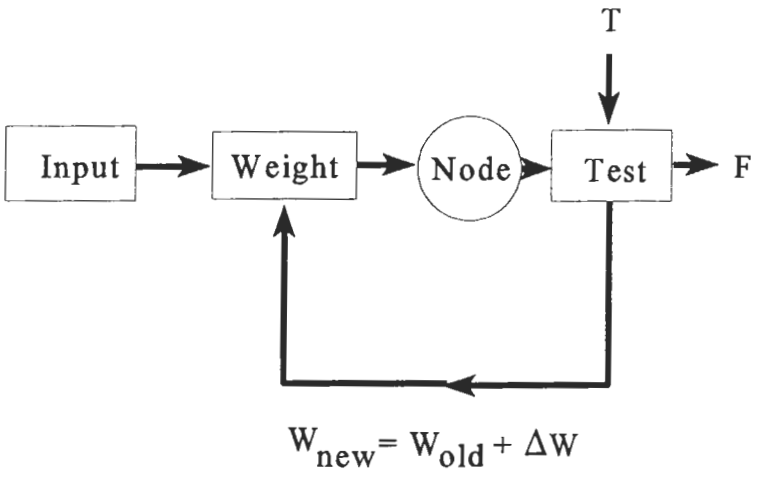

Figure 2.3 Learning mode block diagram.

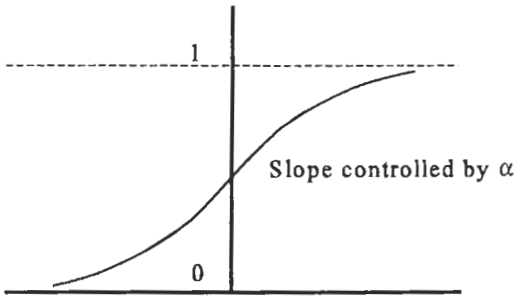

Figure 2.4 Sigmoid Function

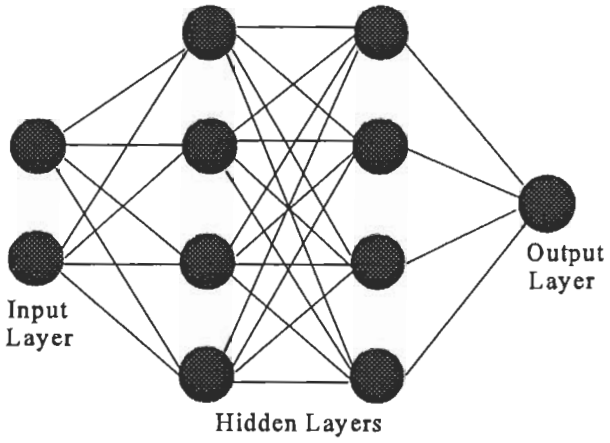

Figure 2.5 Feed forward neural network containing hidden layers.

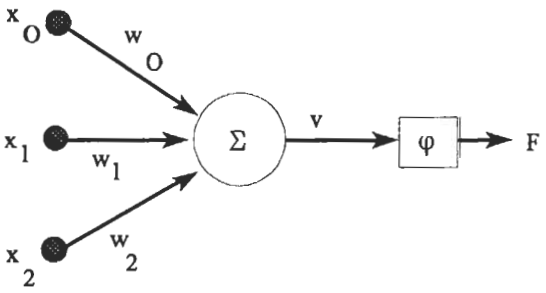

Figure 2.6 Example problem network.

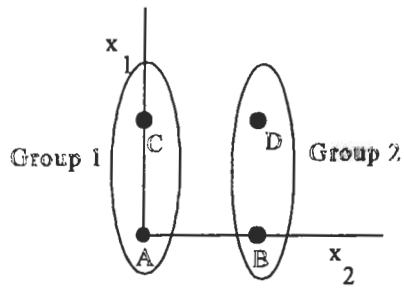

Figure 2.7 Problem definition.

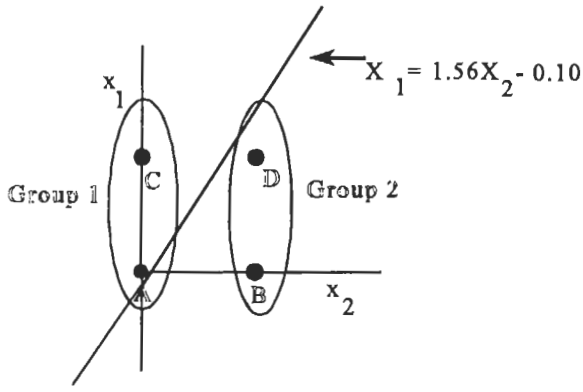

Figure 2.8 Solution to example problem.

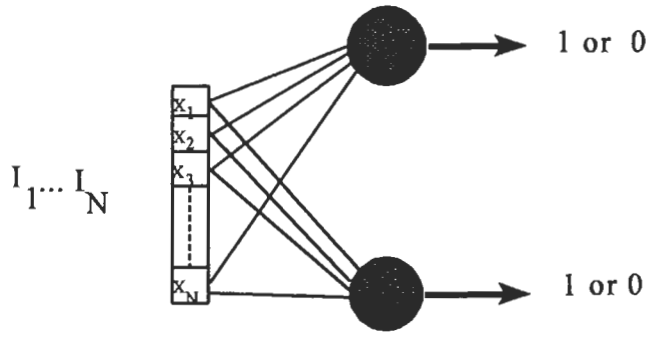

Figure 2.9 Simple Kohonen Network

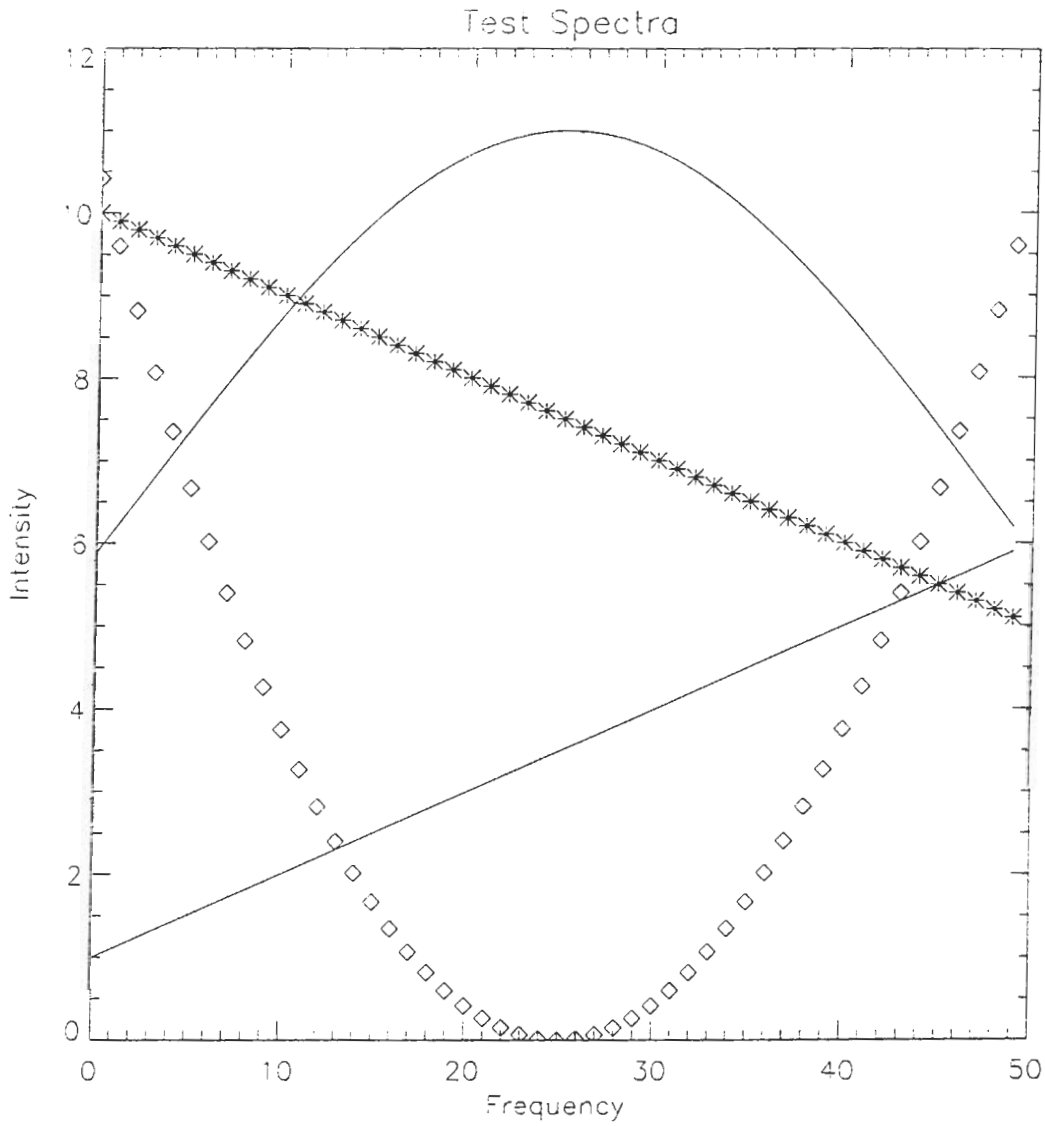

Figure 2.10 Computer Generated Test Spectra

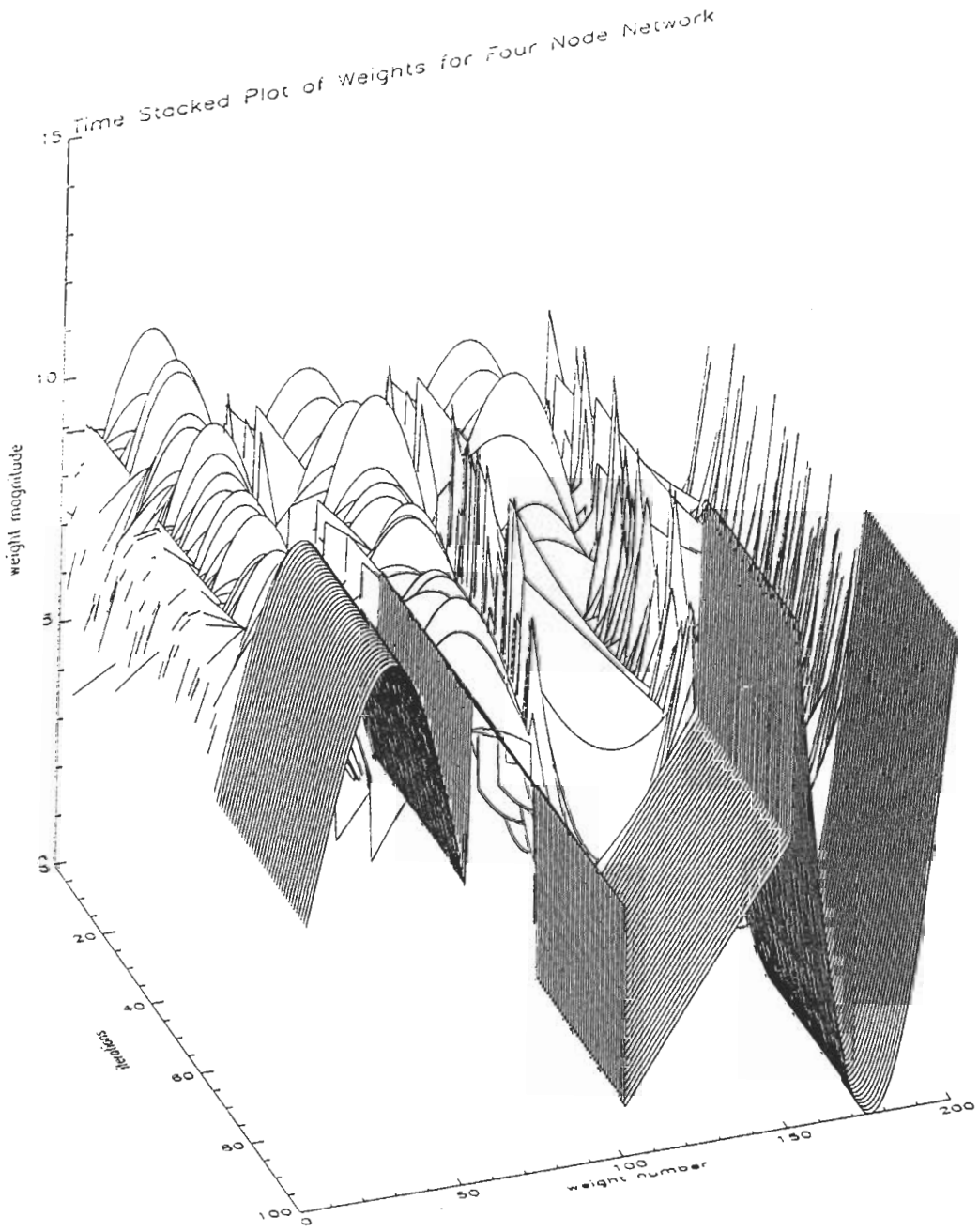

Figure 2.11 Replication Process of Weights for Four Node Network

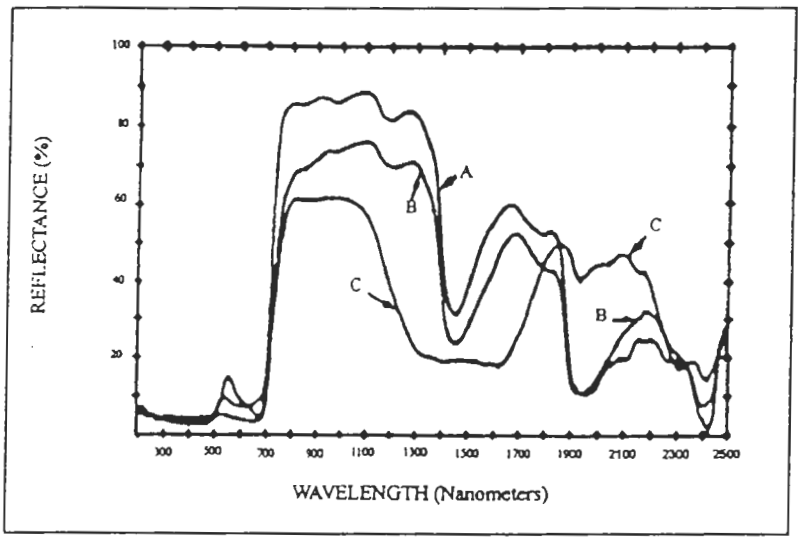

Figure 3.1 Spectral Reflectance Plot (Rinker, 1990)

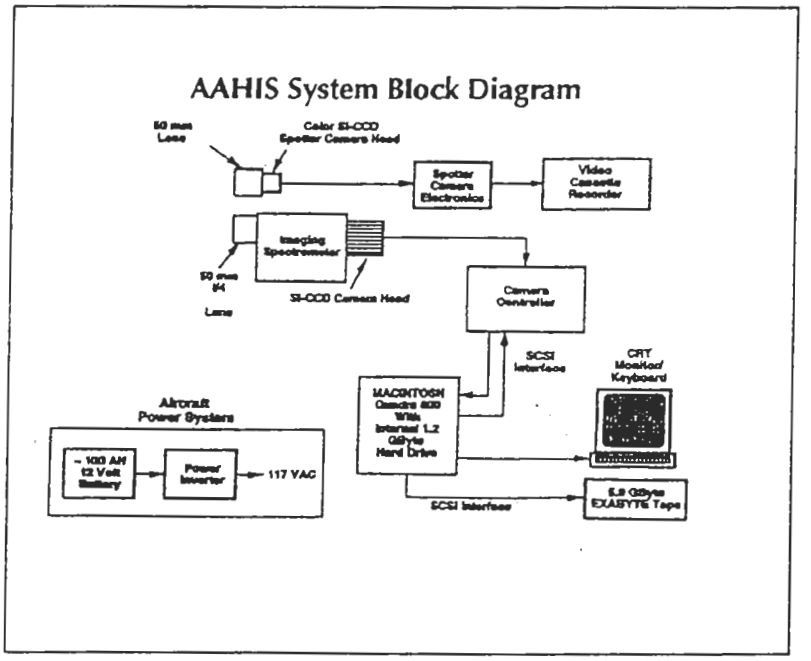

Figure 3.2 AAHIS Block Diagram

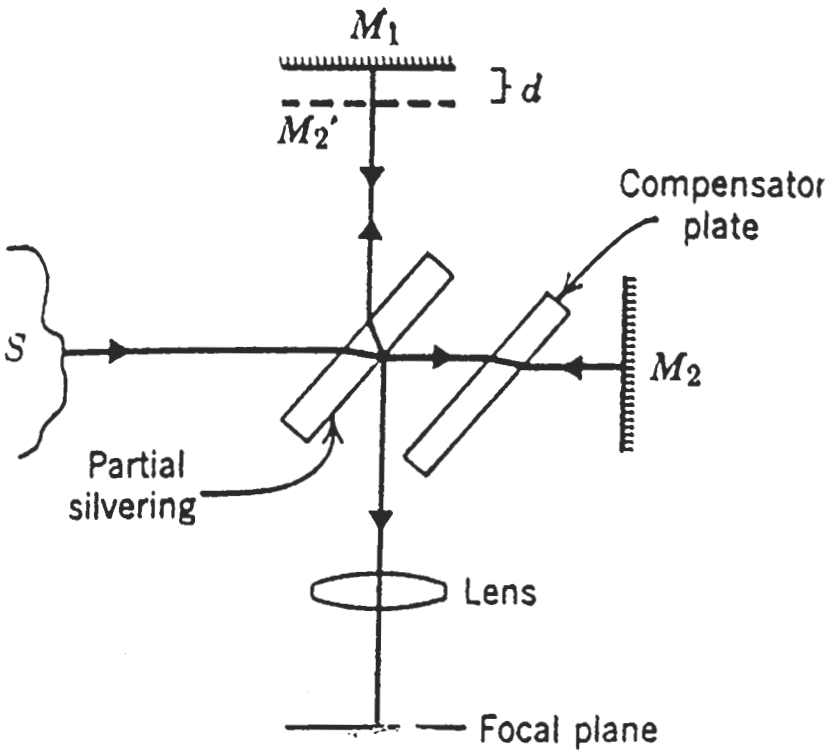

Figure 3.3 Michelson Interferometer (Klein, 1970)

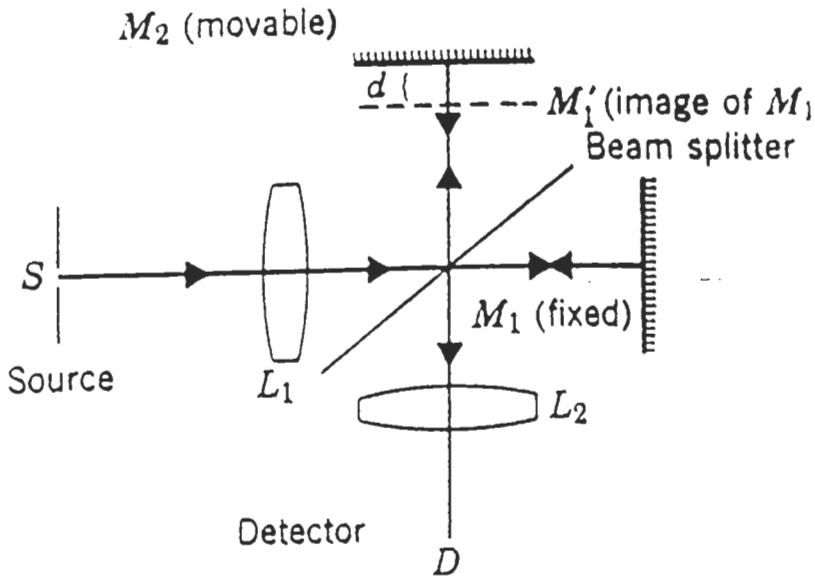

Figure 3.4 Michelson Interferometer for use in Spectroscopy (Klein, 1970)

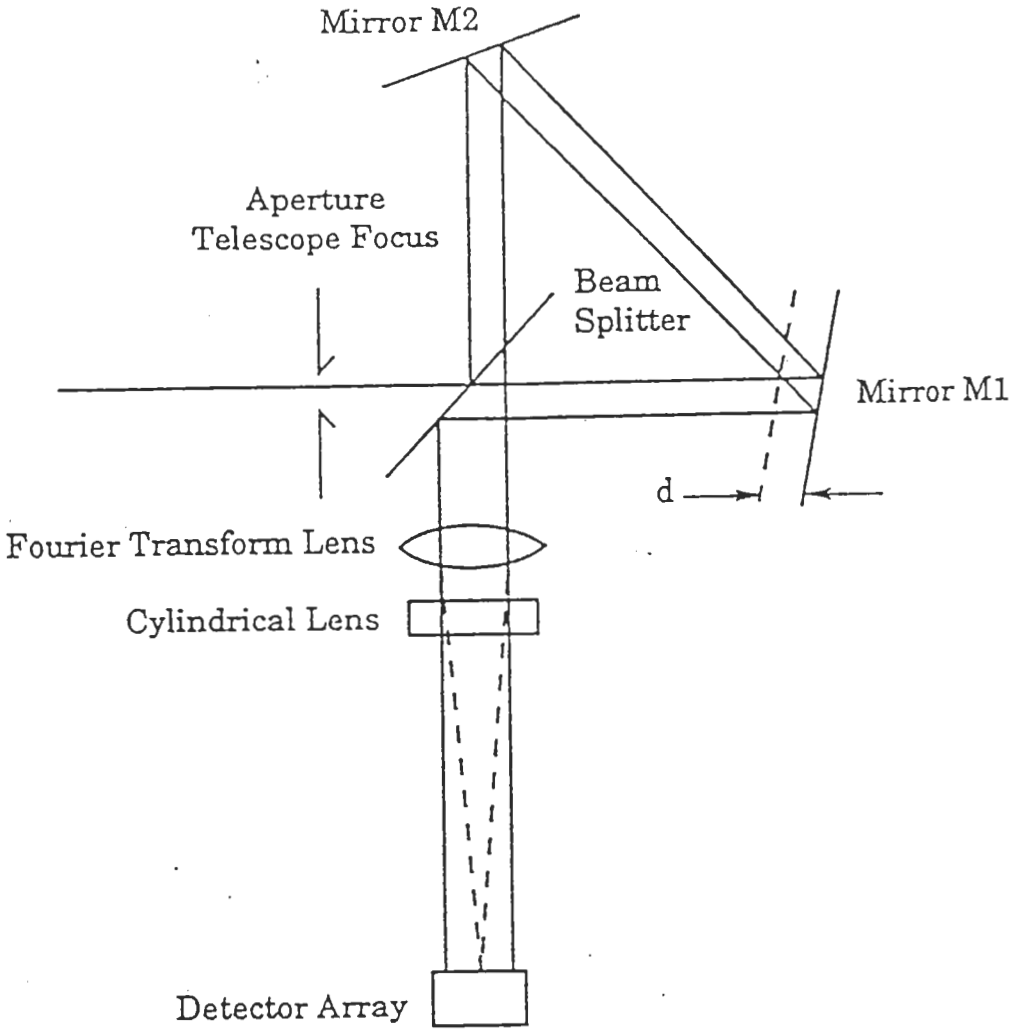

Figure 3.5 Schematic of SMIFTS Instrument (Lucey, et al., 1993)

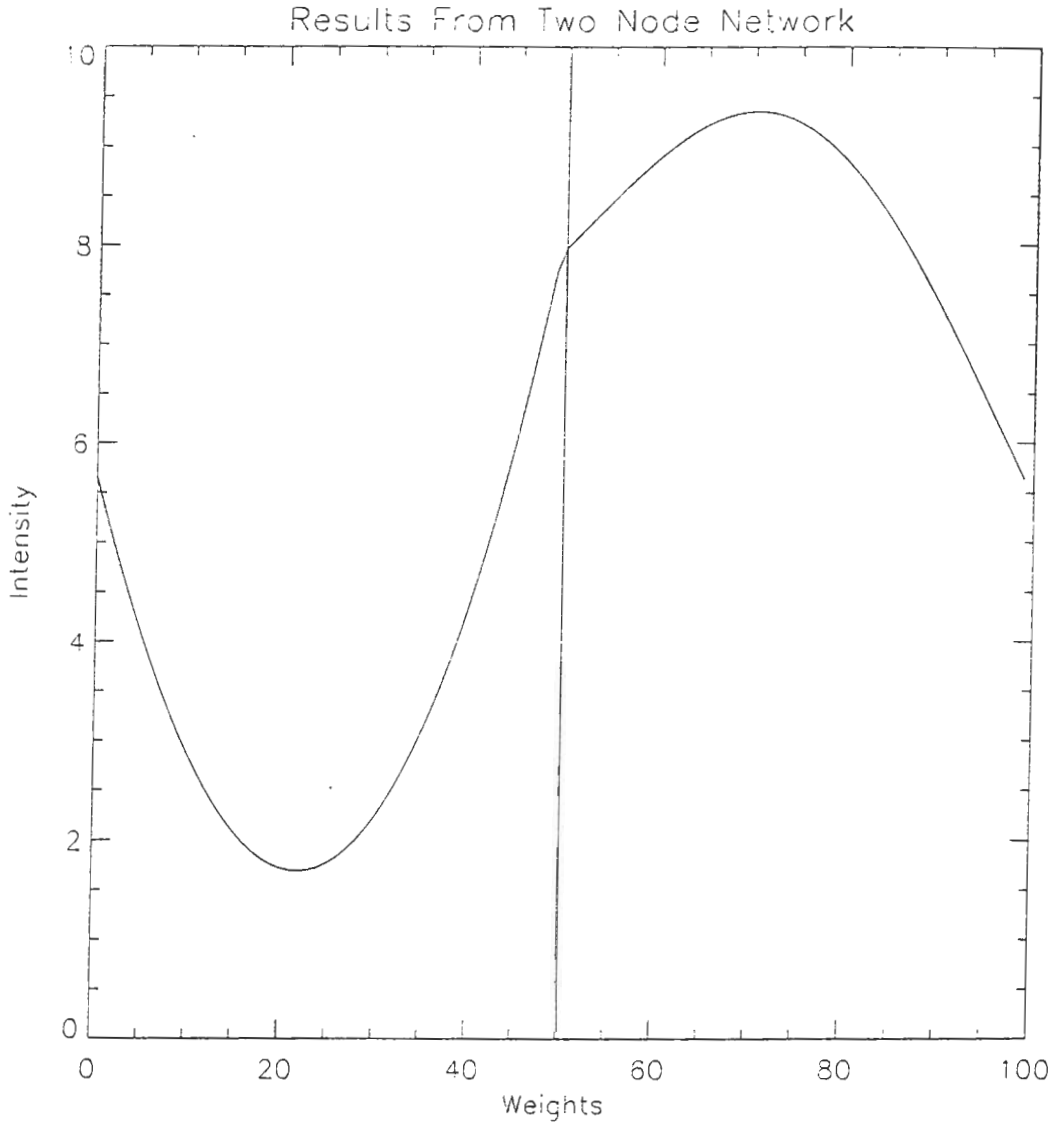

Figure 4.1 Plot of Final Weights for Two Node Network

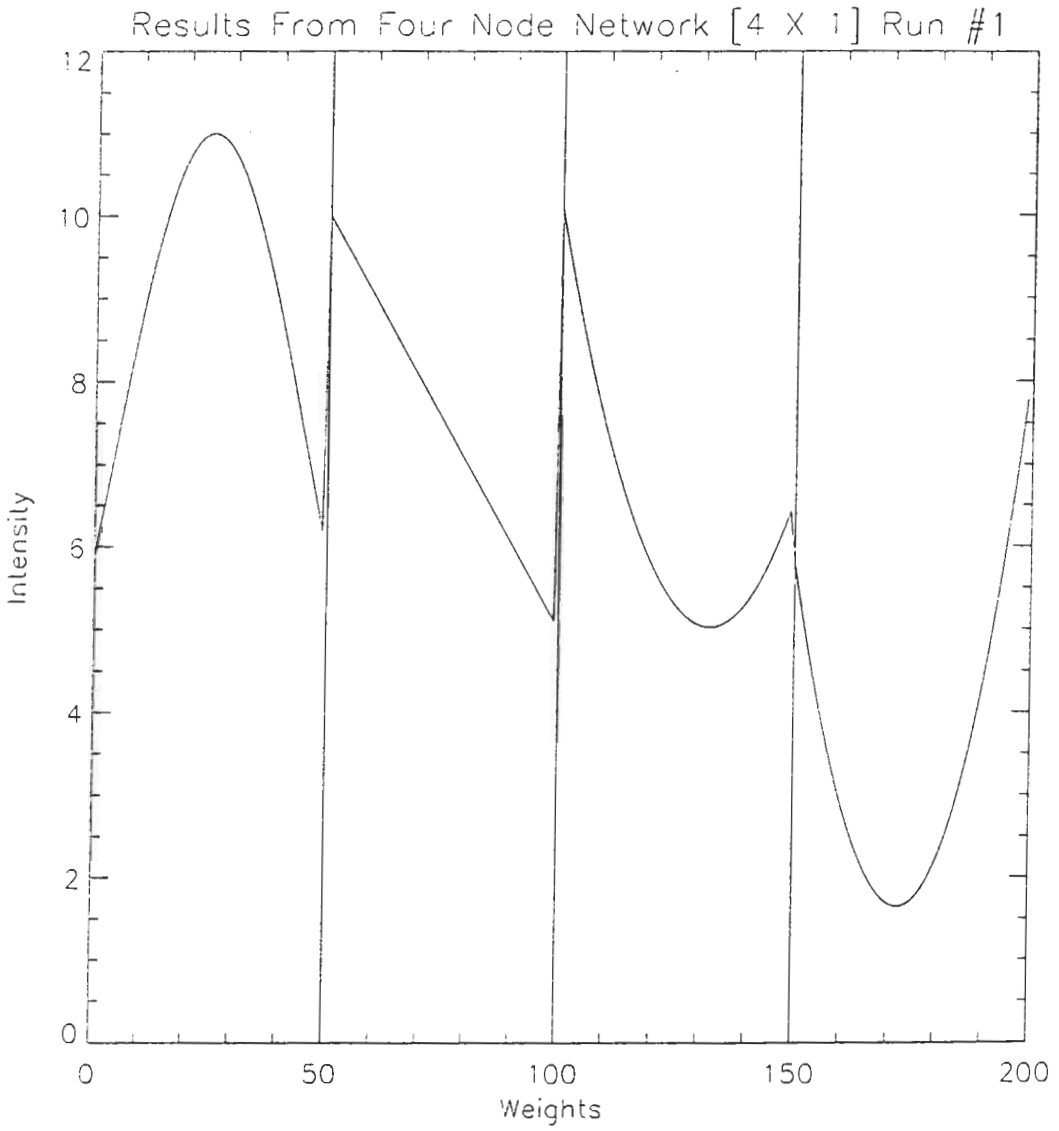

Figure 4.2 Plot of Final Weights for Four Node Network, Run # 1

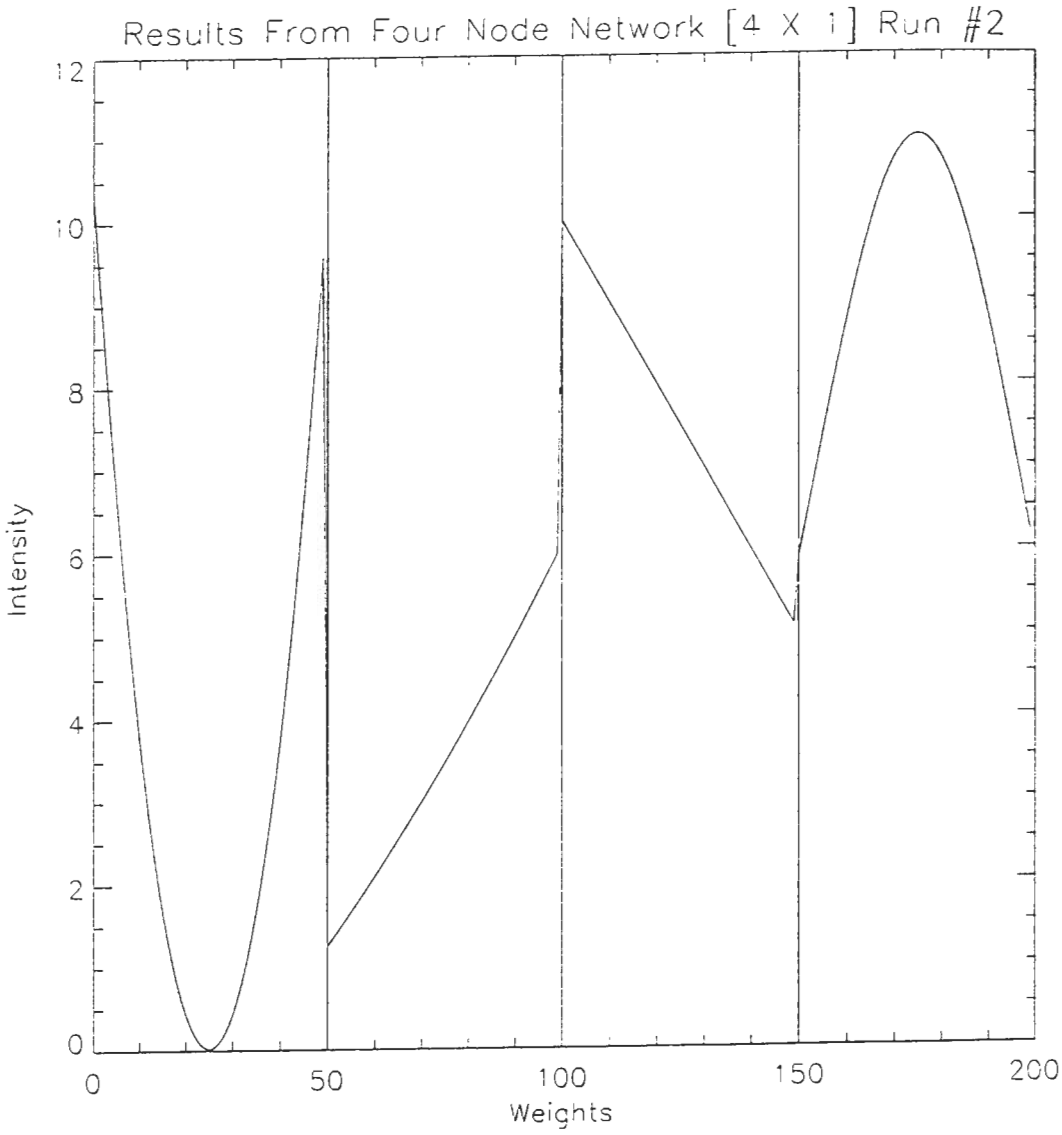

Figure 4.3 Plot of Final Weights for Four Node Network, Run # 2

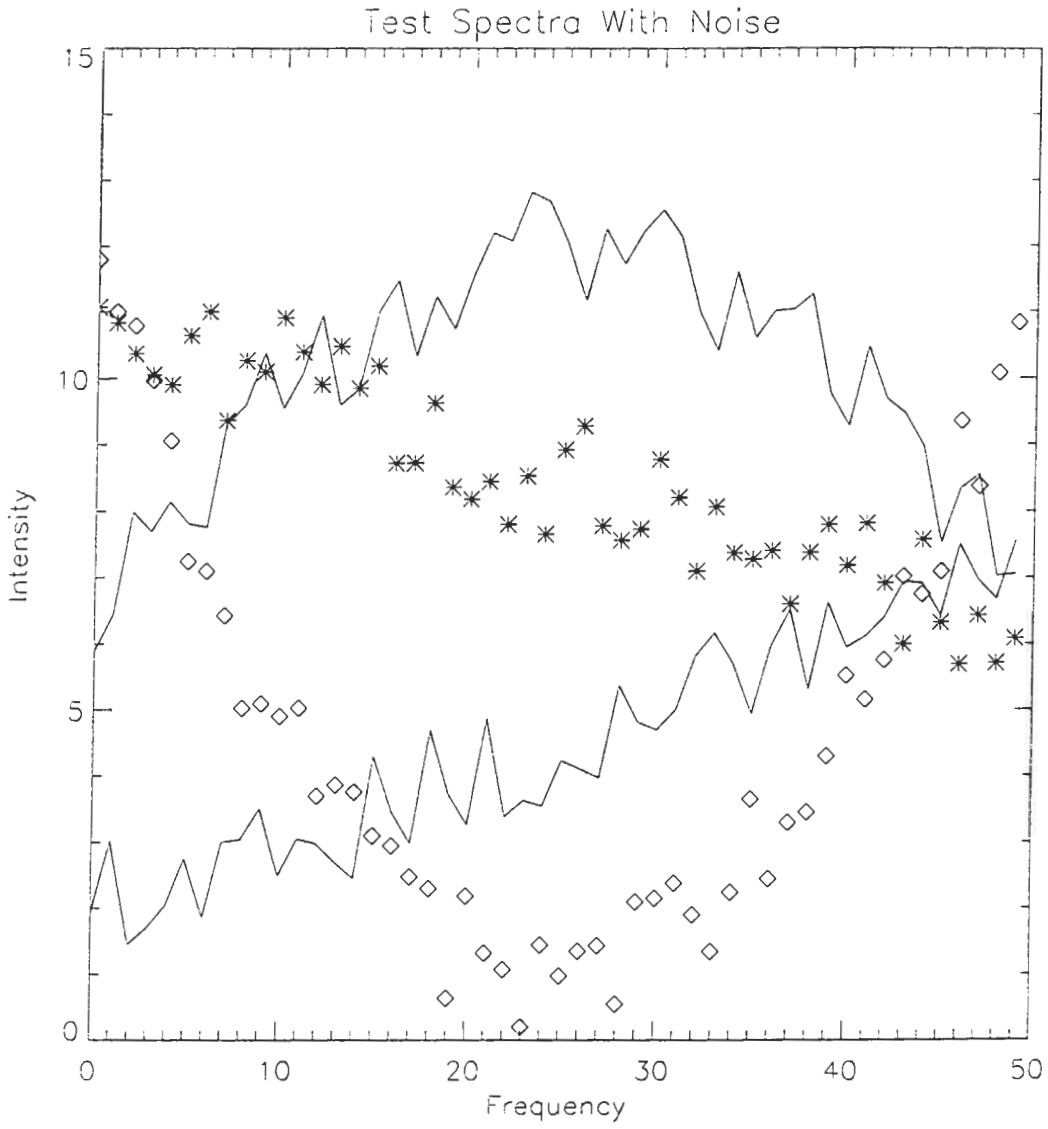

Figure 4.4 Computer Generated Test Spectra with Noise Added

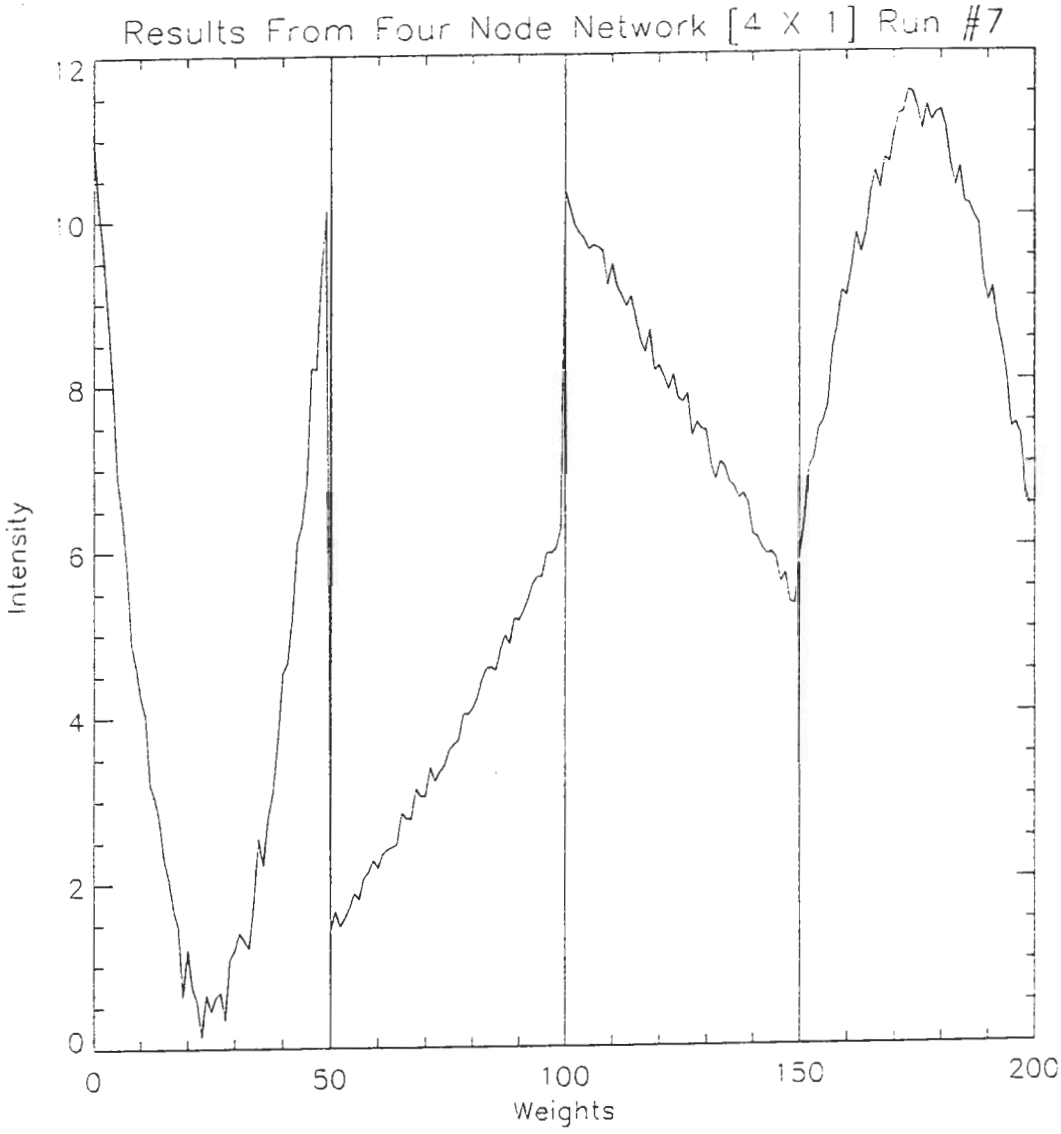

Figure 4.5 Plot of Final Weights for Four Node Network, Run # 7

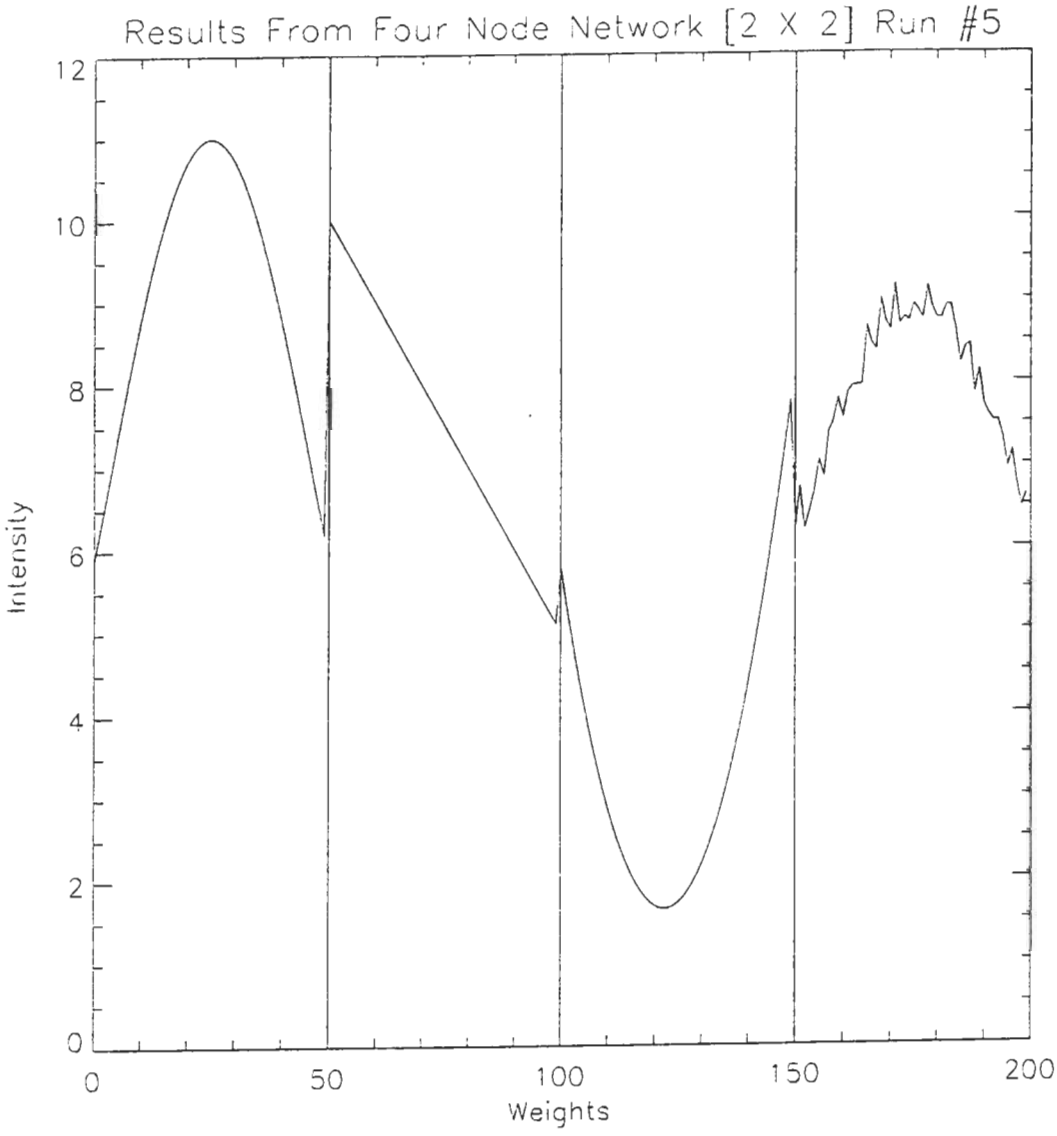

Figure 4.6 Plot of Final Weights for Four Node Network, 2 X 2, Run # 5

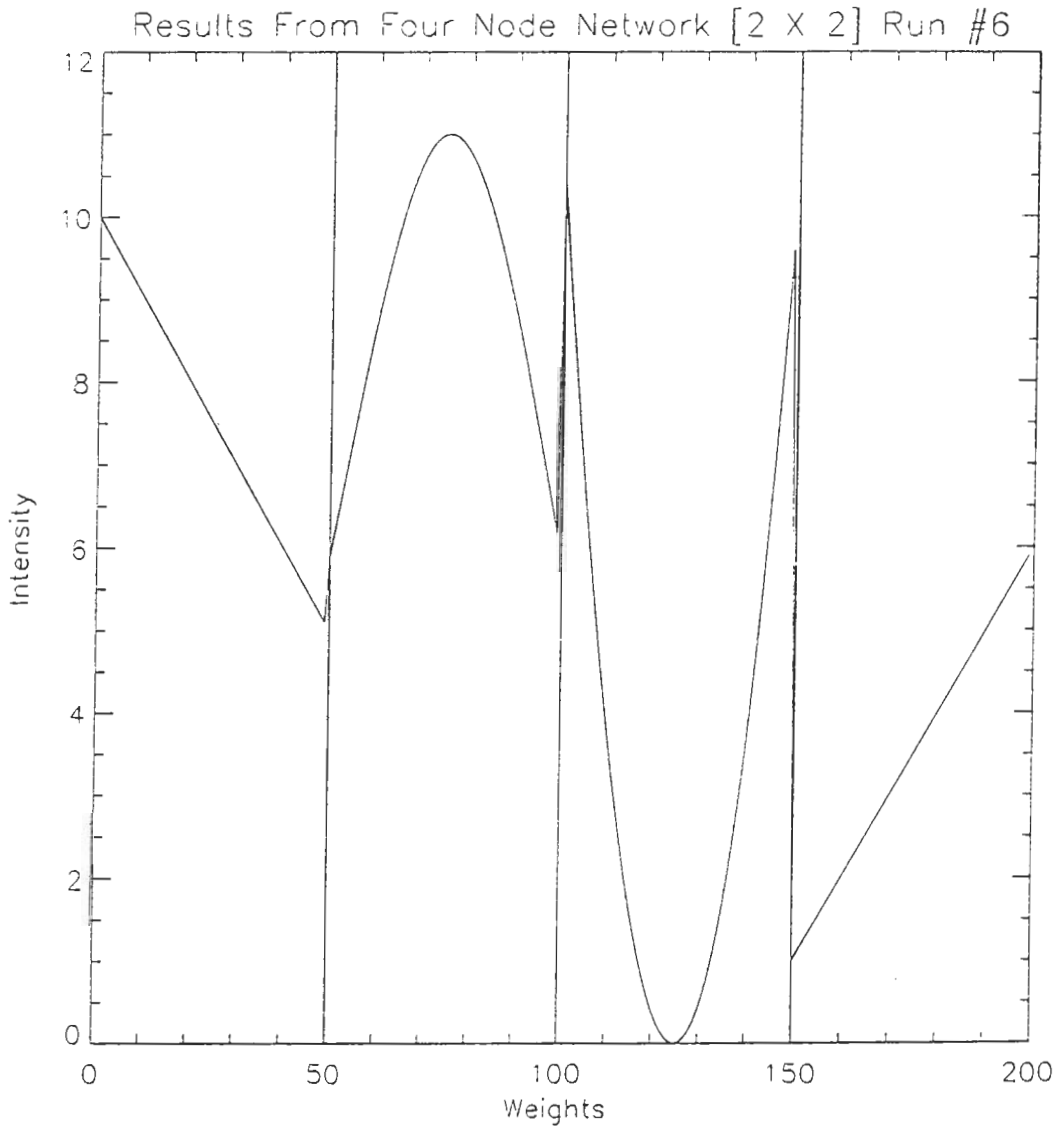

Figure 4.7 Plot of Final Weights for Four Node Network, 2 X 2, Run # 6

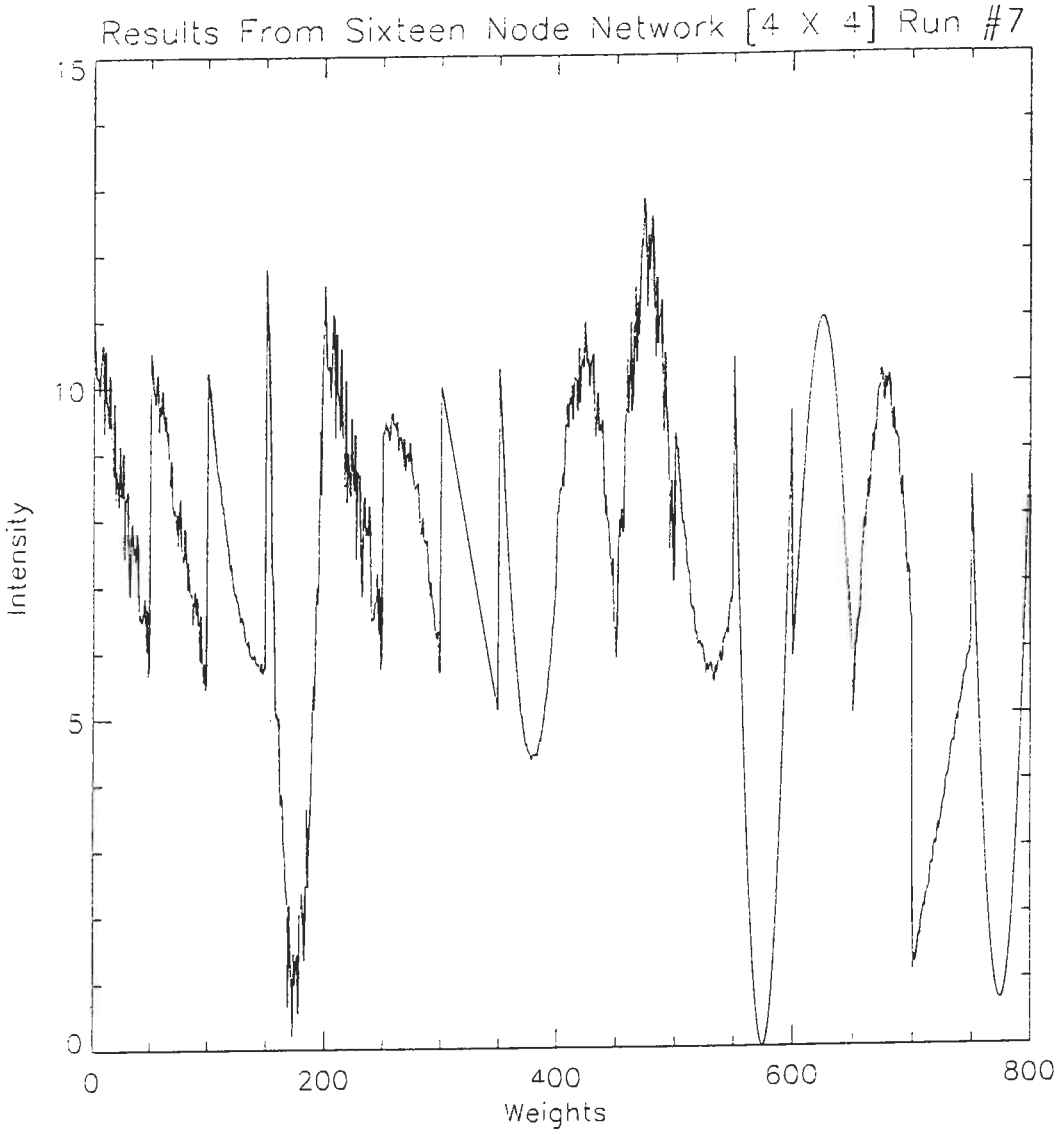

Figure 4.8 Plot of Final Weights for Sixteen Node Network, Run # 7

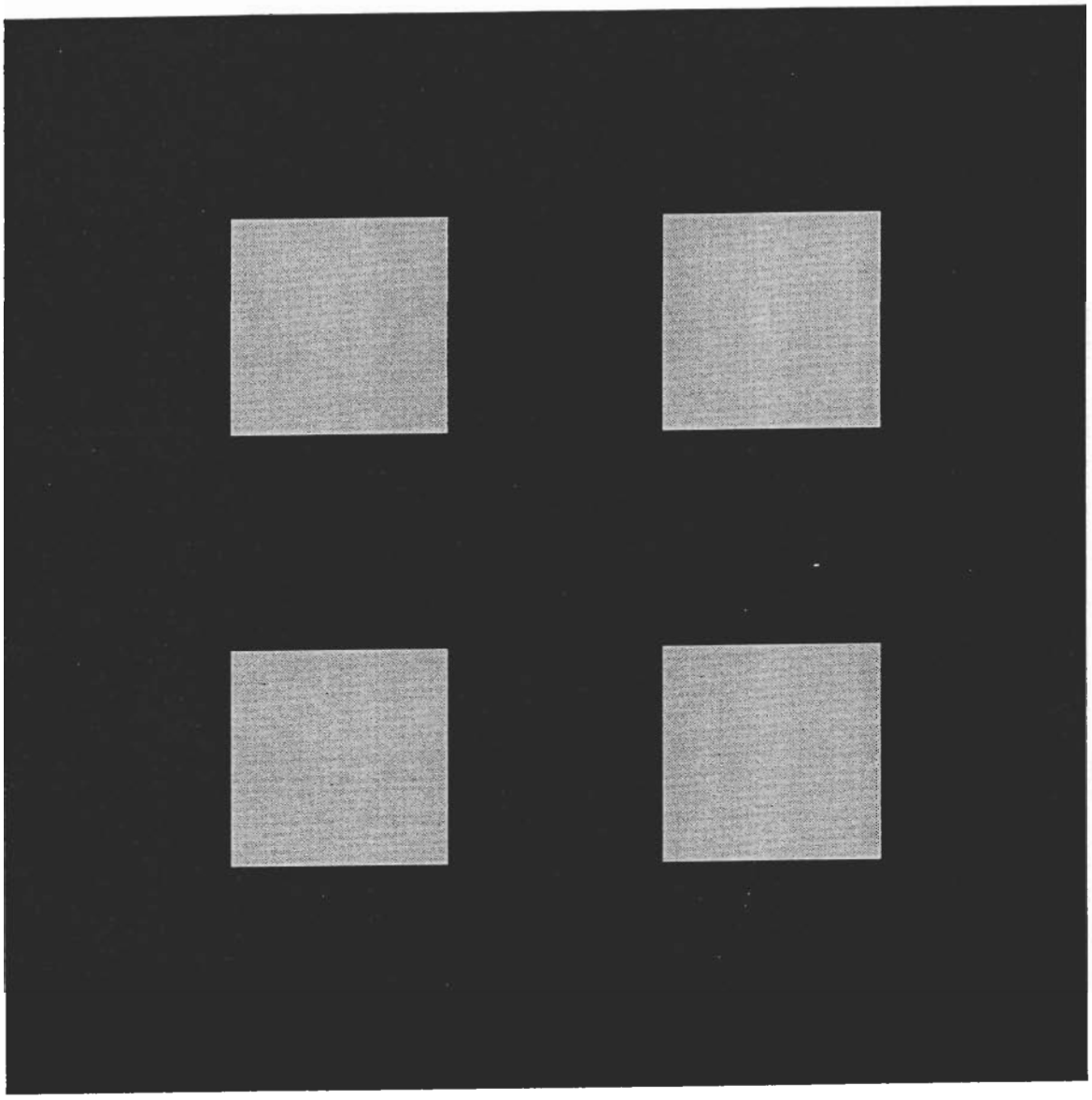

Figure 4.9 Test Image

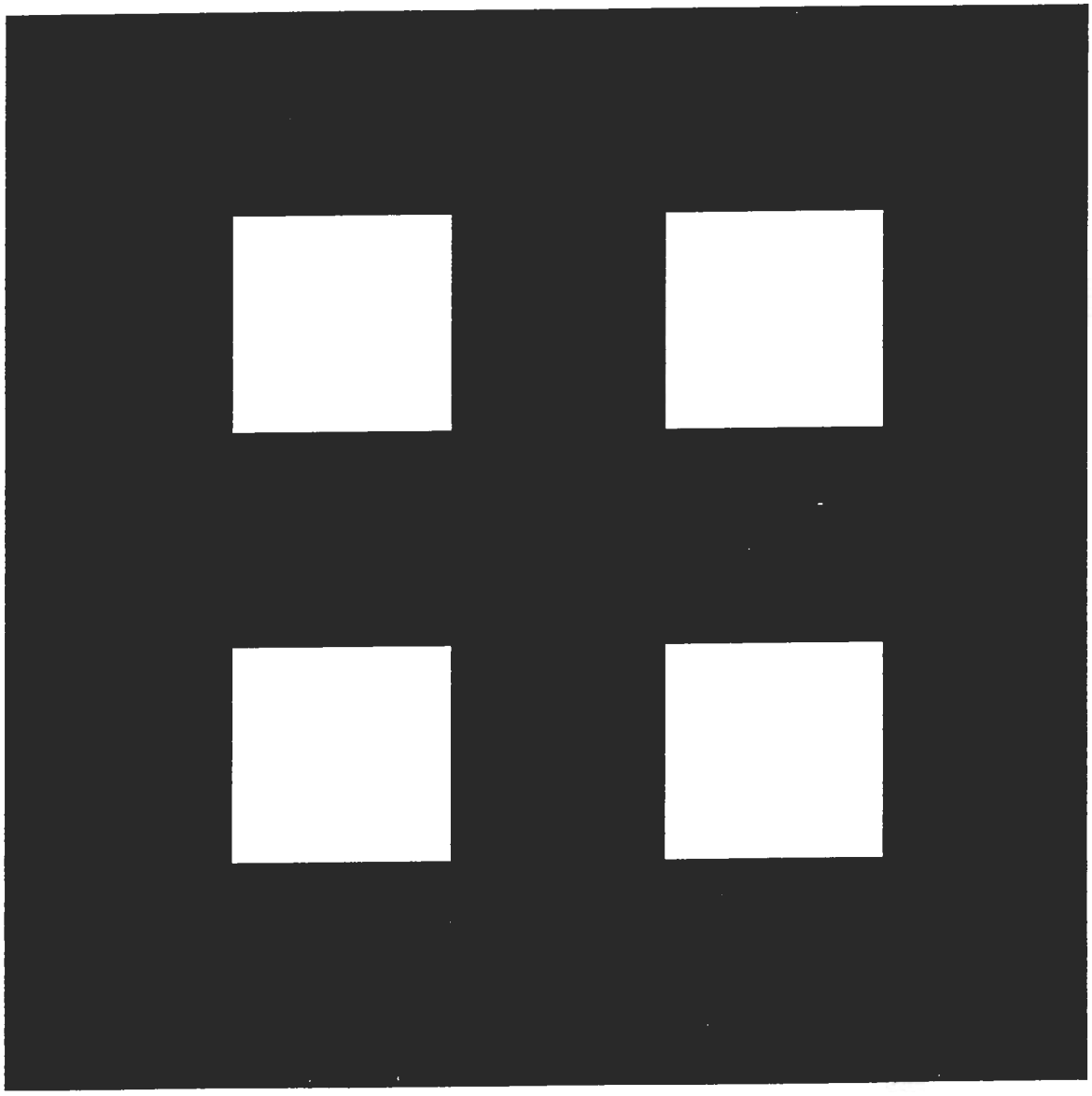

Figure 4.10 Network Categorization of Test Image

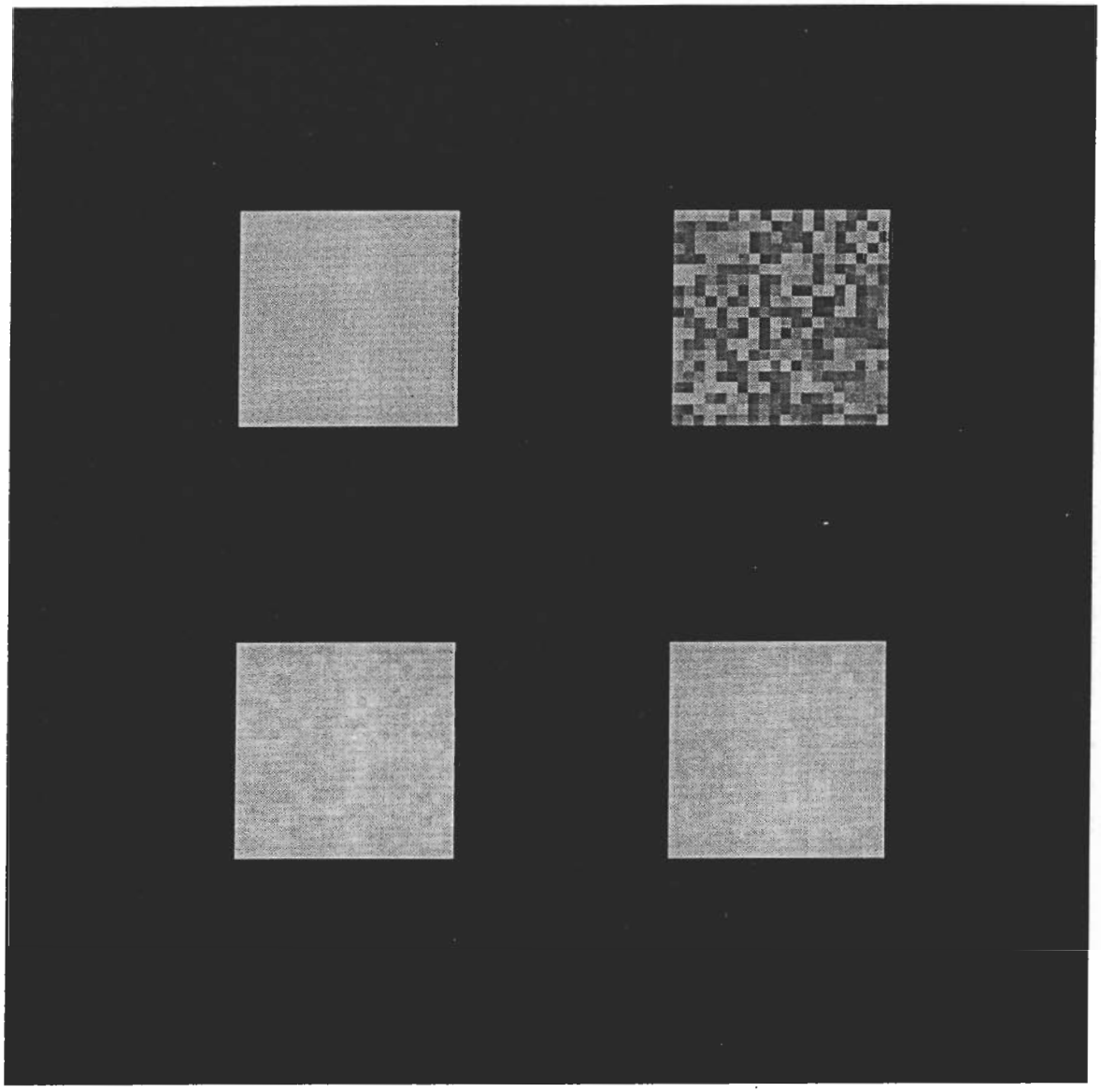

Figure 4.11 Test Image with Noise

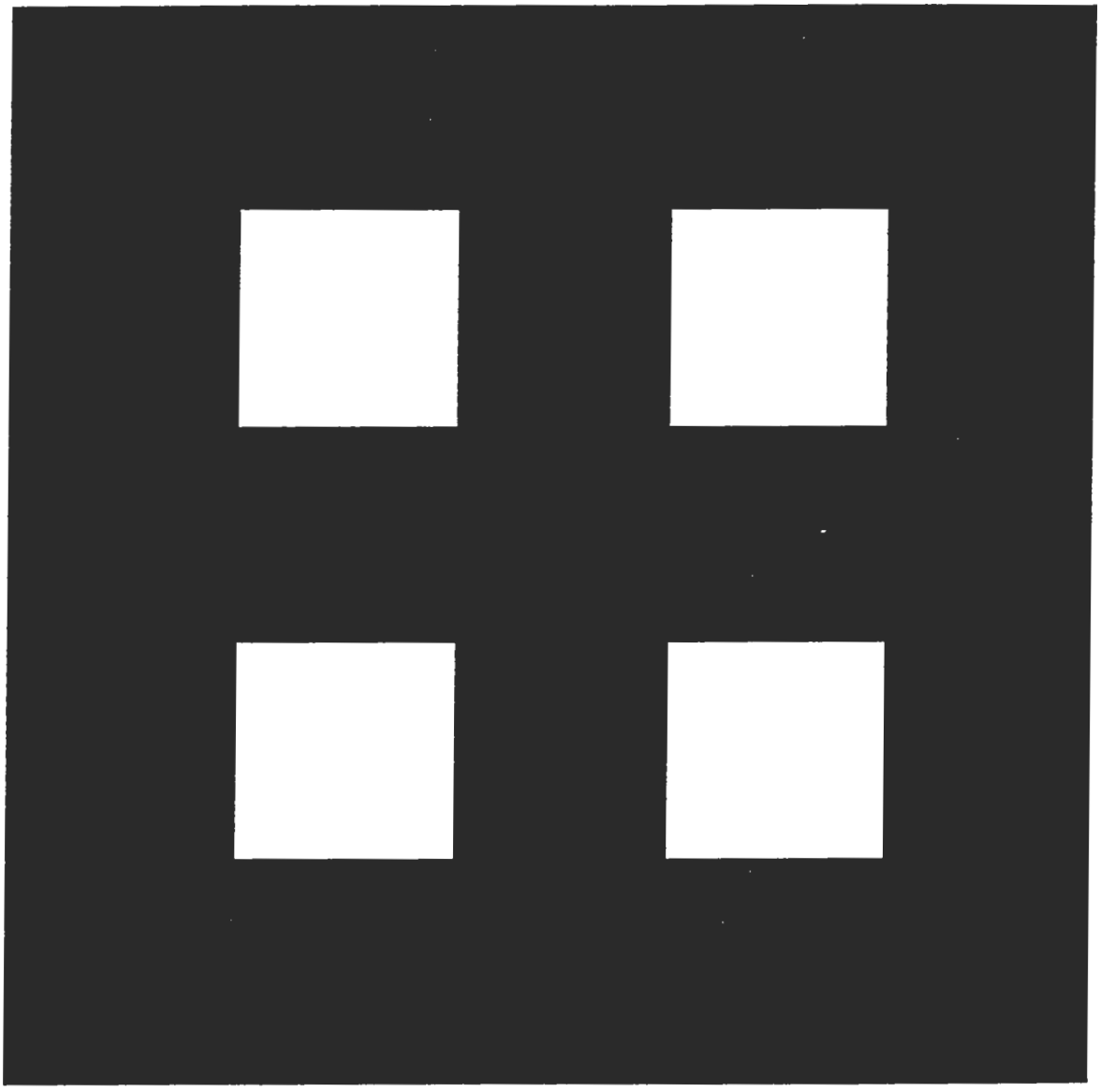

Figure 4.12 Two Node Network Categorization: Noise < 50%

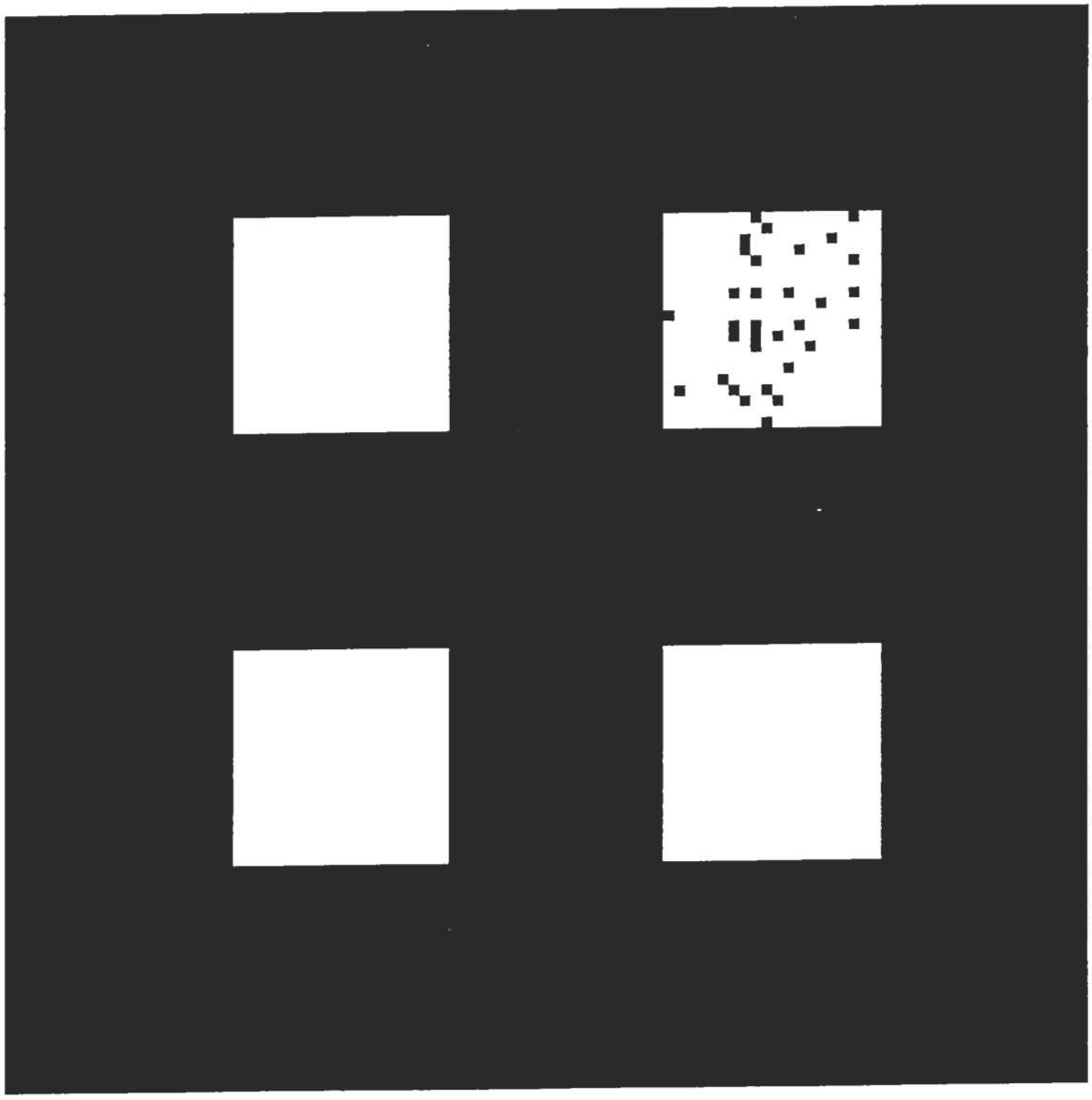

Figure 4.13 Two Node Network Categorization: Noise = 55%

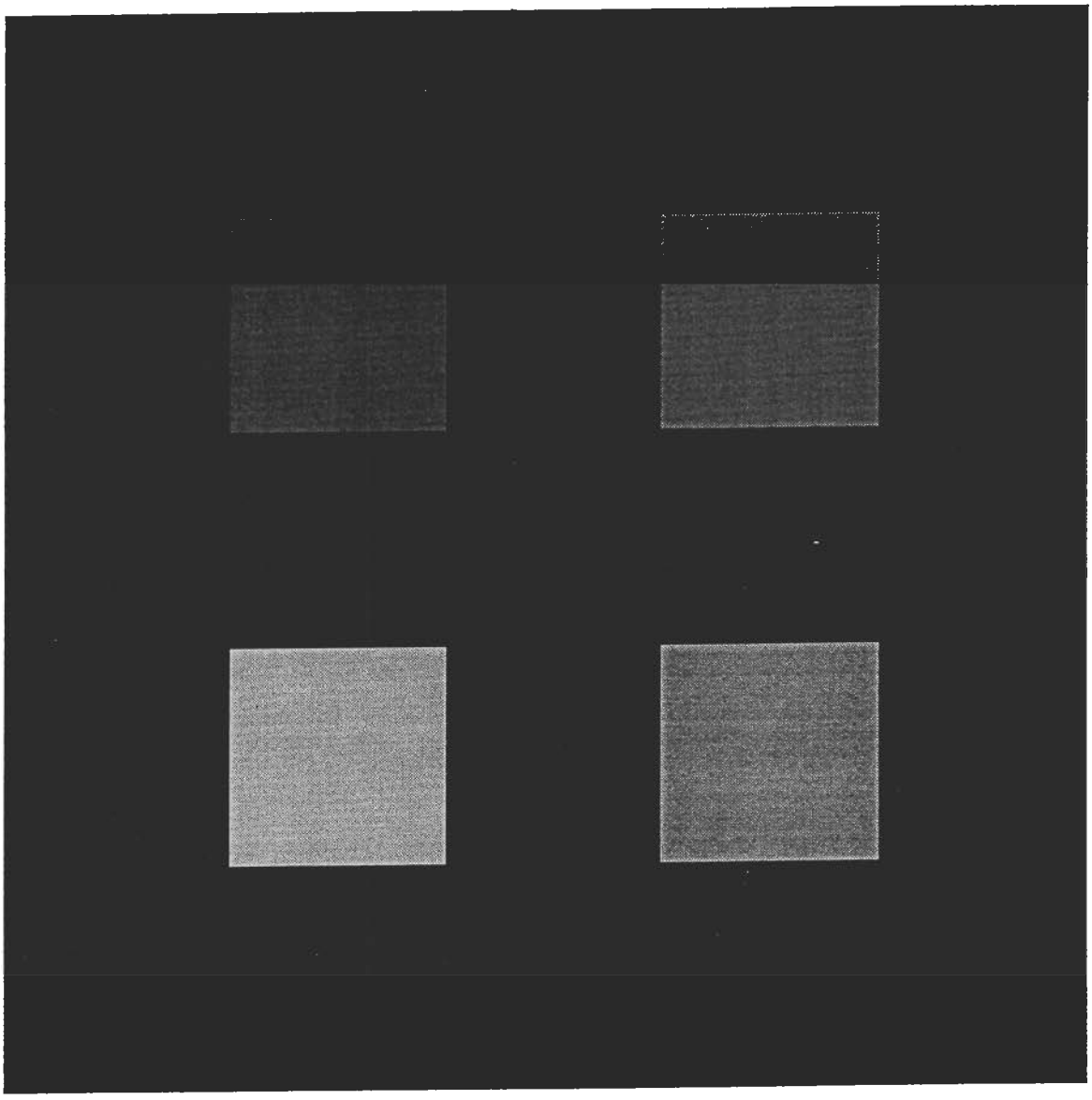

Figure 4.14 Variable Grey Test Image

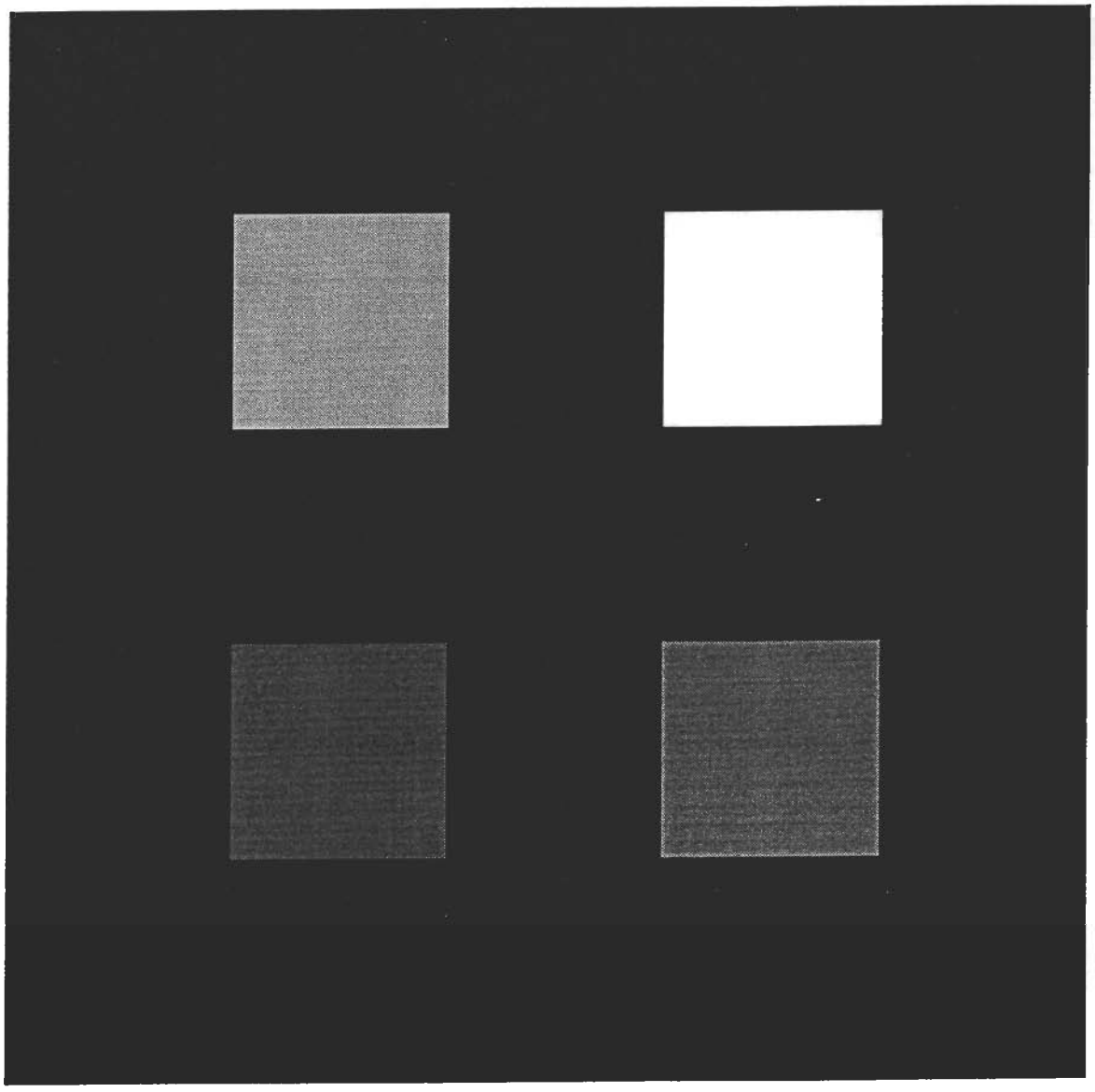

Figure 4.15 Five Node Network Categorization

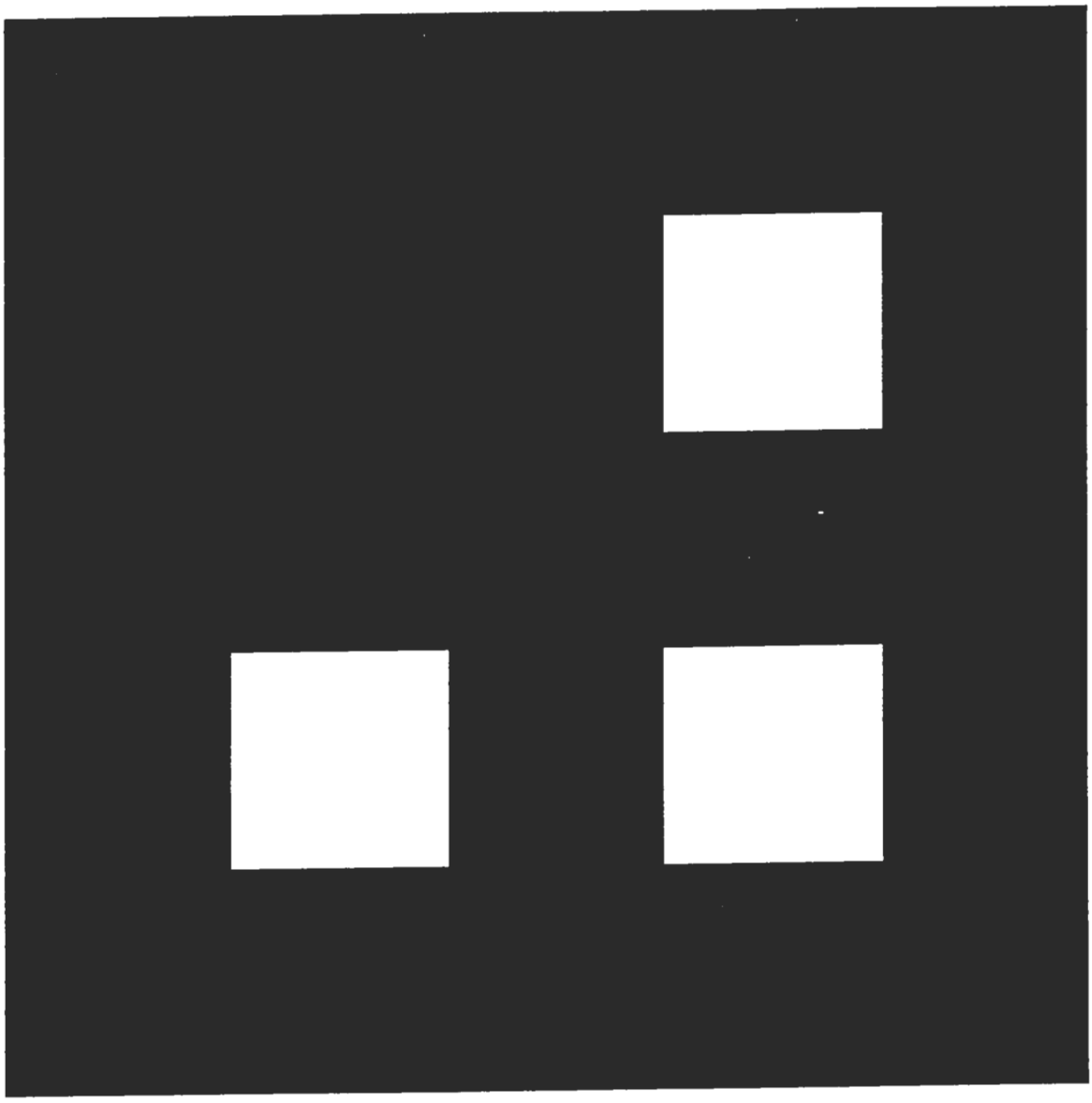

Figure 4.16 Two Node Network Categorization of Variable Grey Test Image

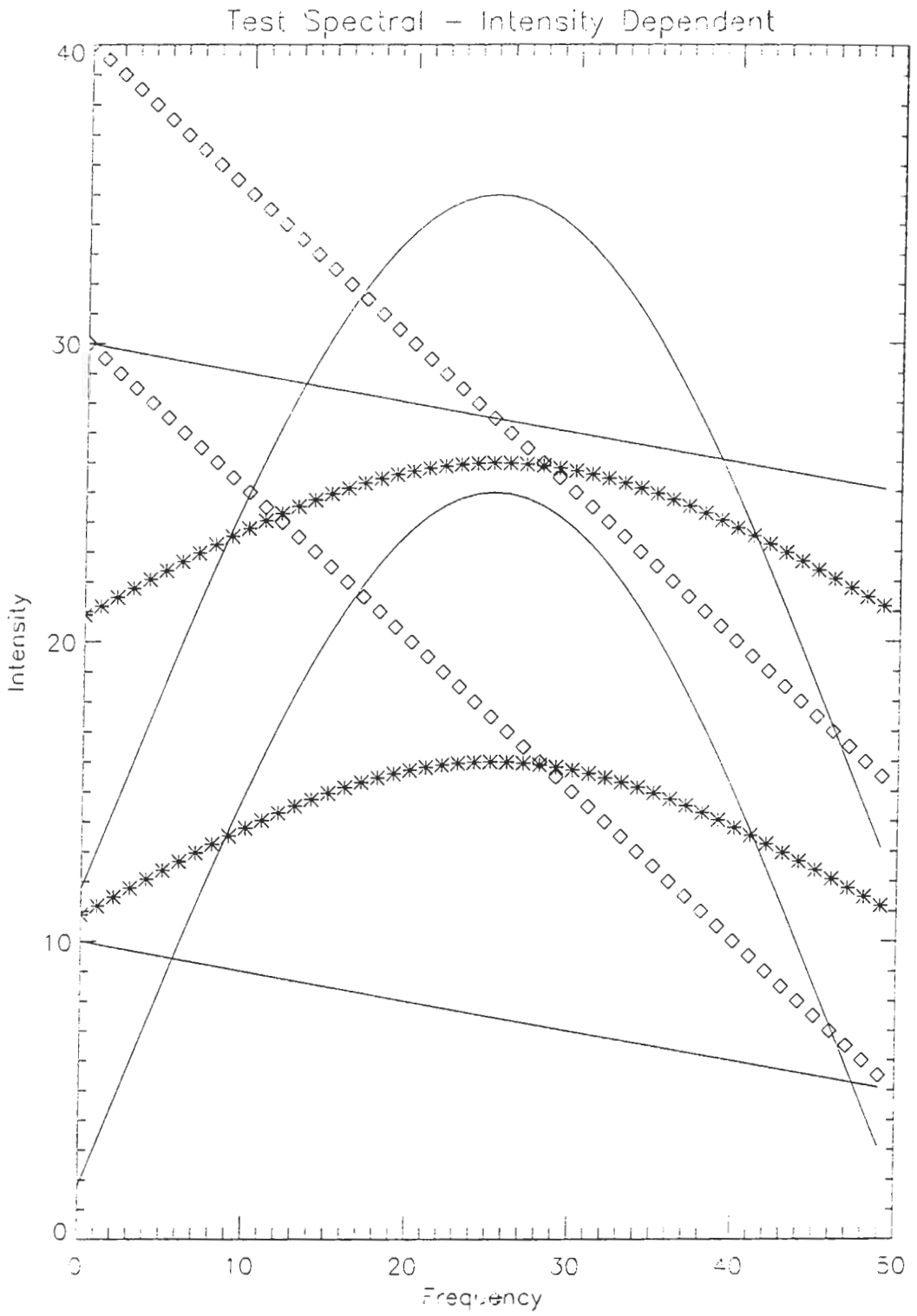

Figure 4.17 Computer Generated Test Spectra with Intensity Difference

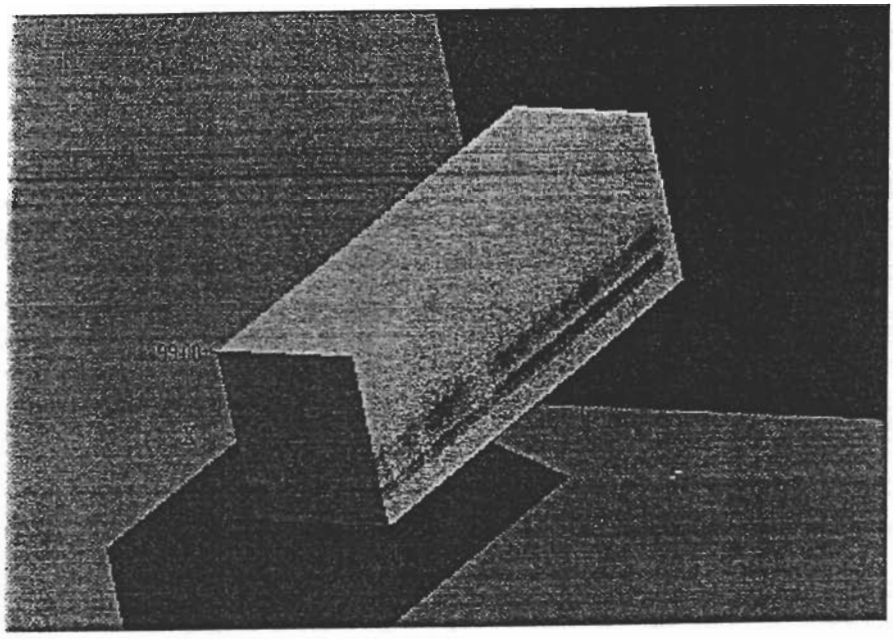

Figure 4.18 Hypercube of Camouflage Data

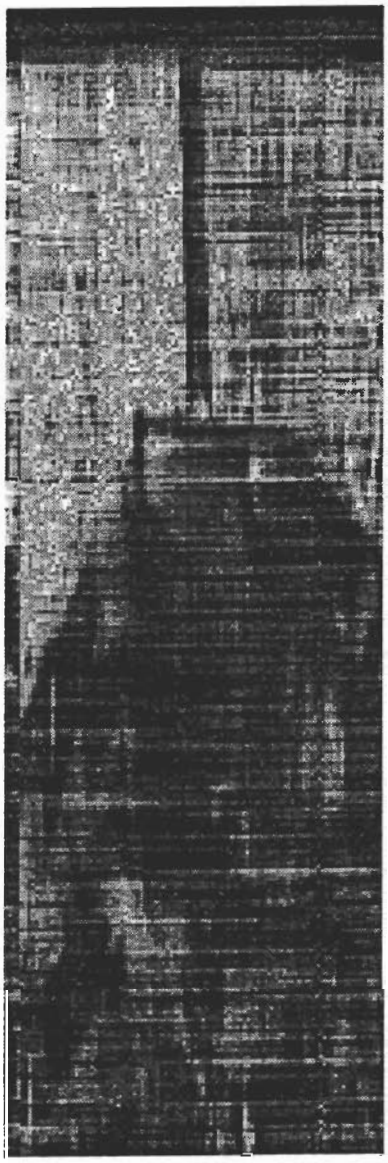

Figure 4.19 Band 37 of Camouflage Data

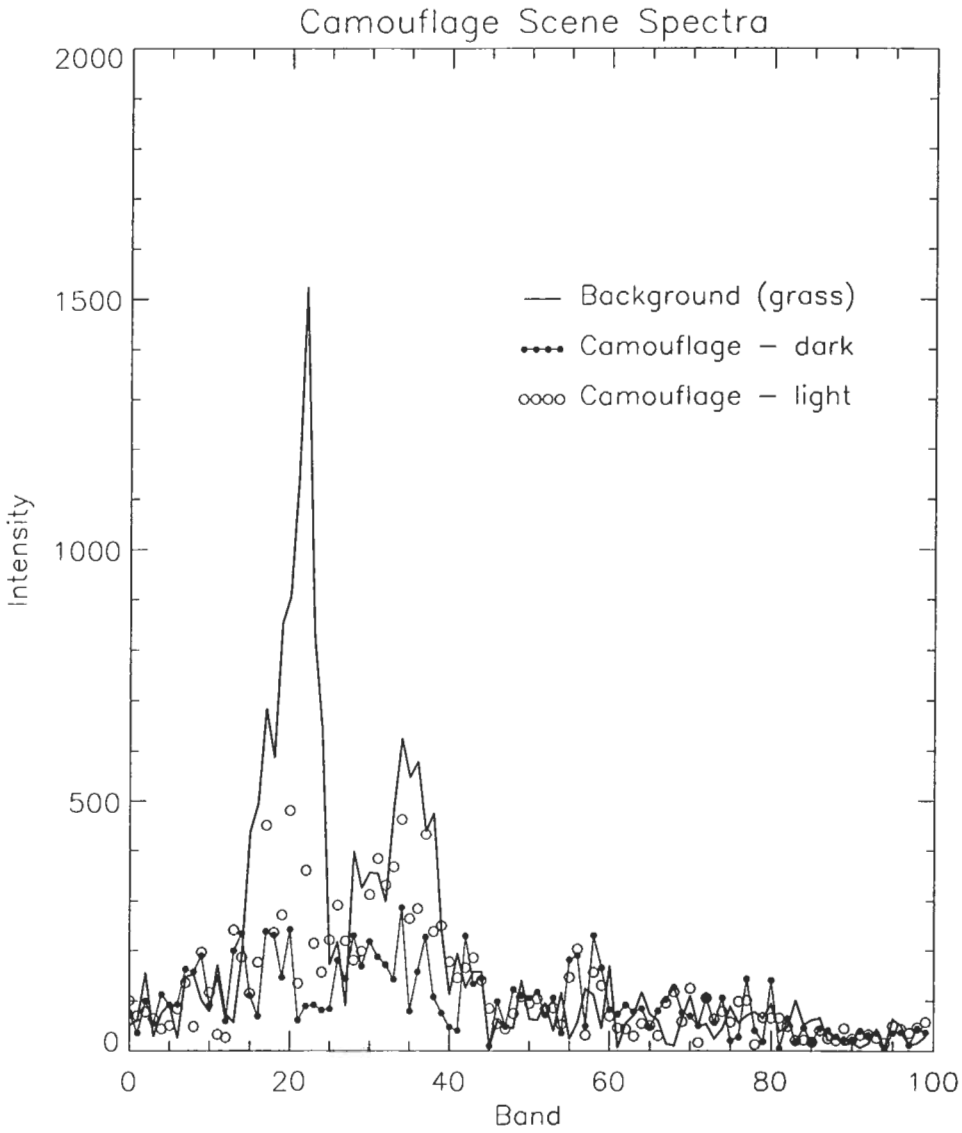

Figure 4.20 Representative Spectra of Camouflage Data

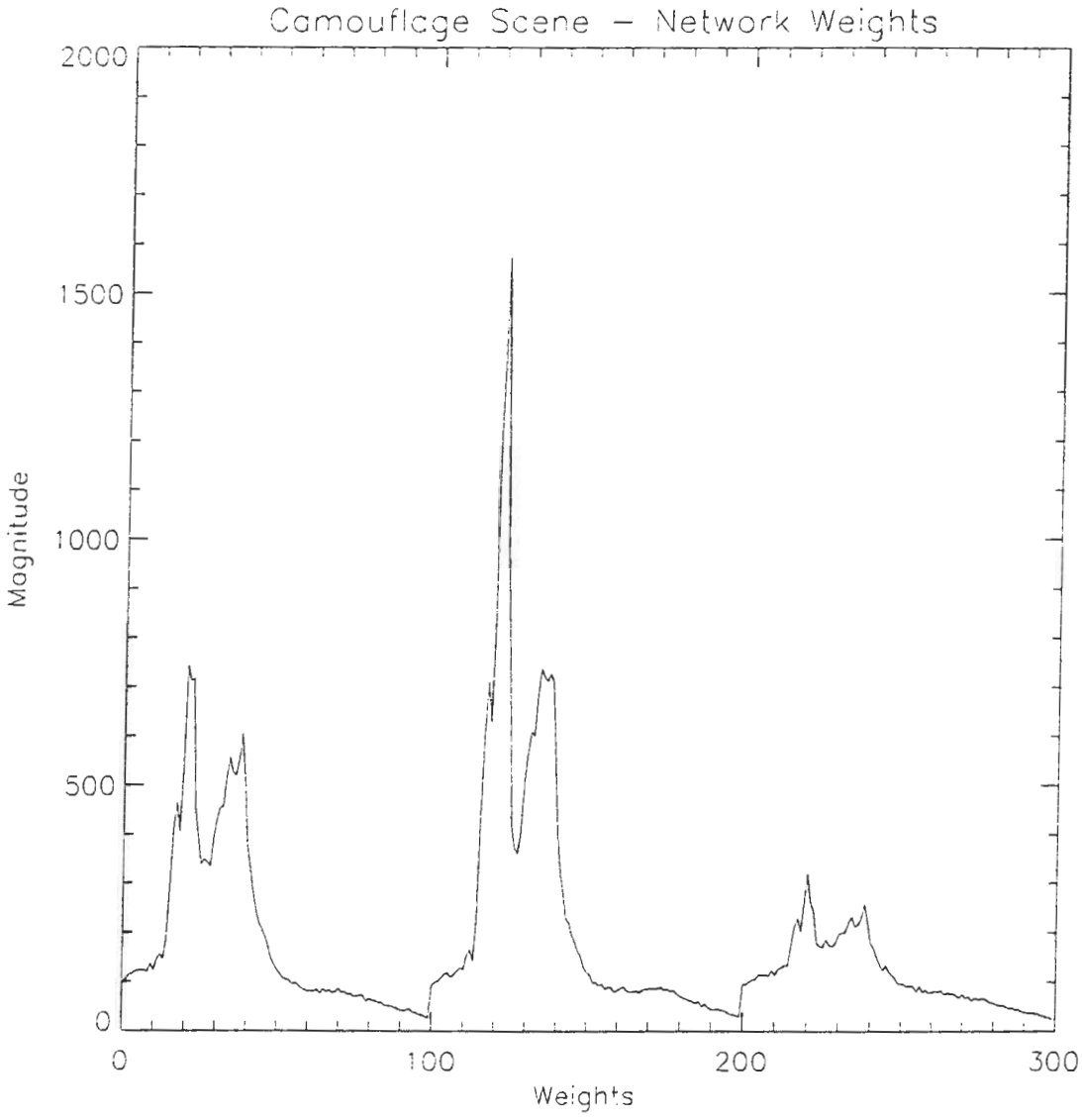

Figure 4.21 Final Weights for Network Categorization of Camouflage Data Using Absolute Sum of Differences

Figure 4.22 Sum of Absolute Differences Categorization of Camouflage Data

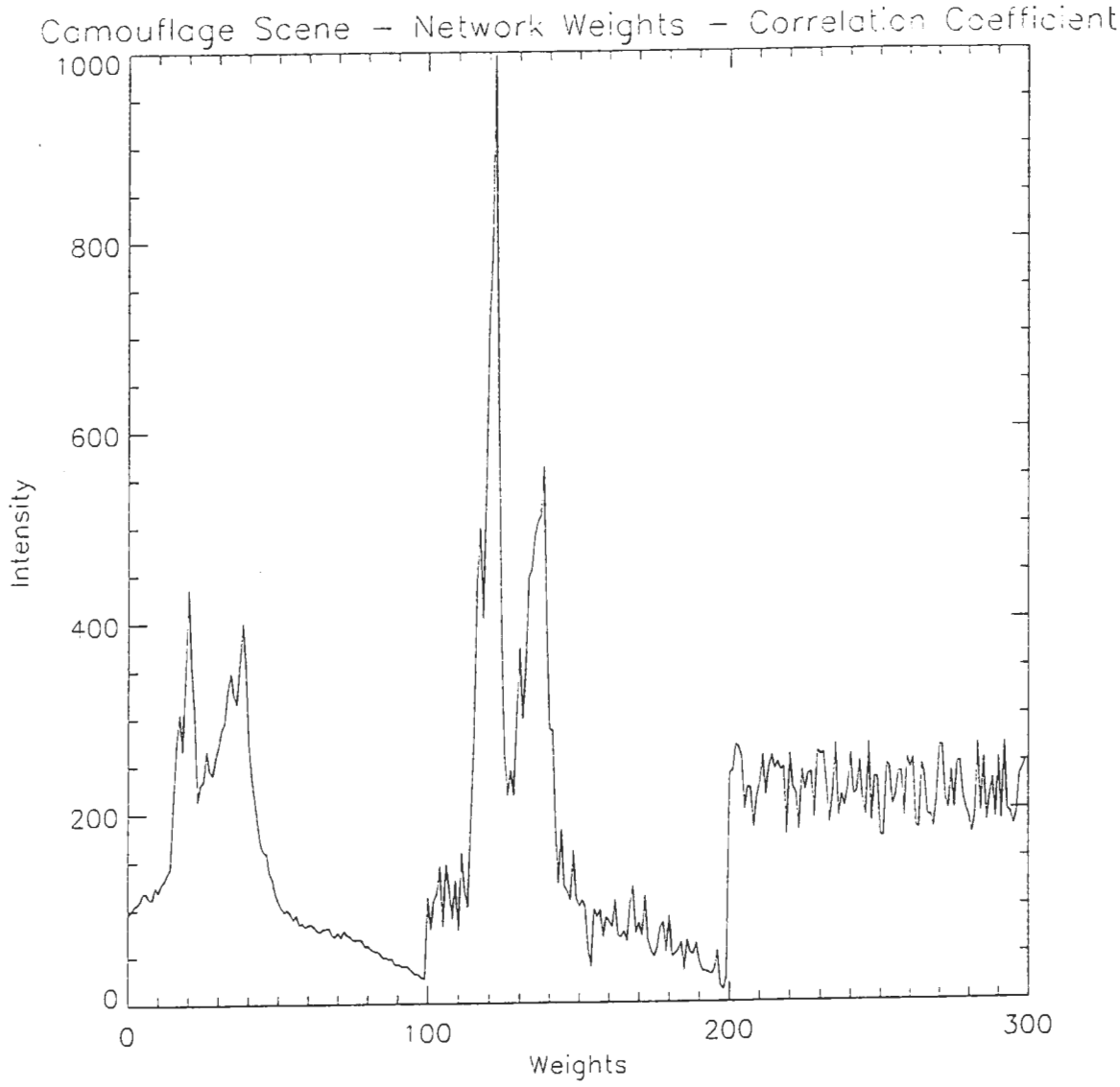

Figure 4.23 Final Weights for Network Categorization of Camouflage Data Using Correlation Function

Figure 4.24 Camouflage Image Categorization by Correlation Coeficient

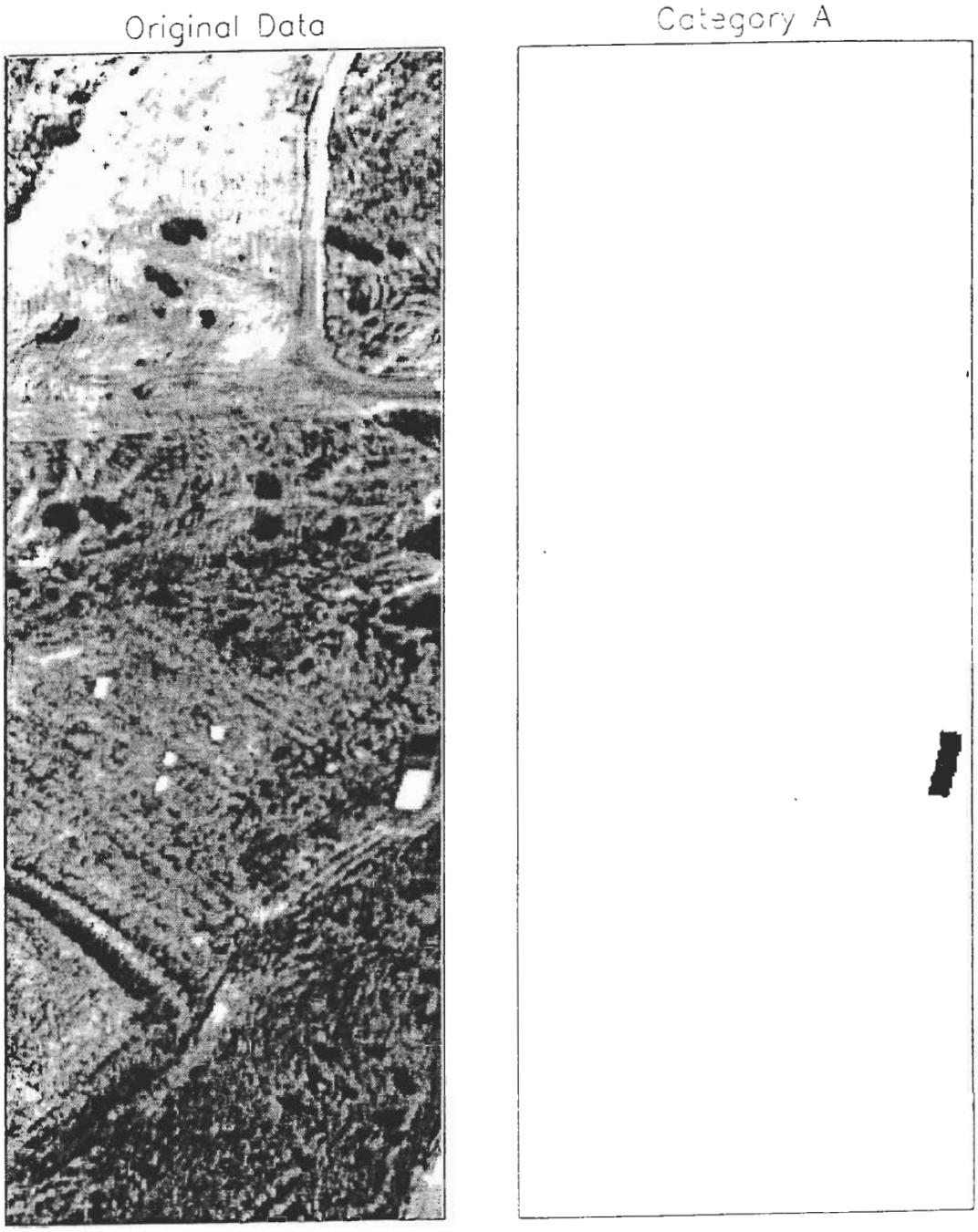

Figure 4.25 Network Category A for Desert Radiance Data, Calibration Panel

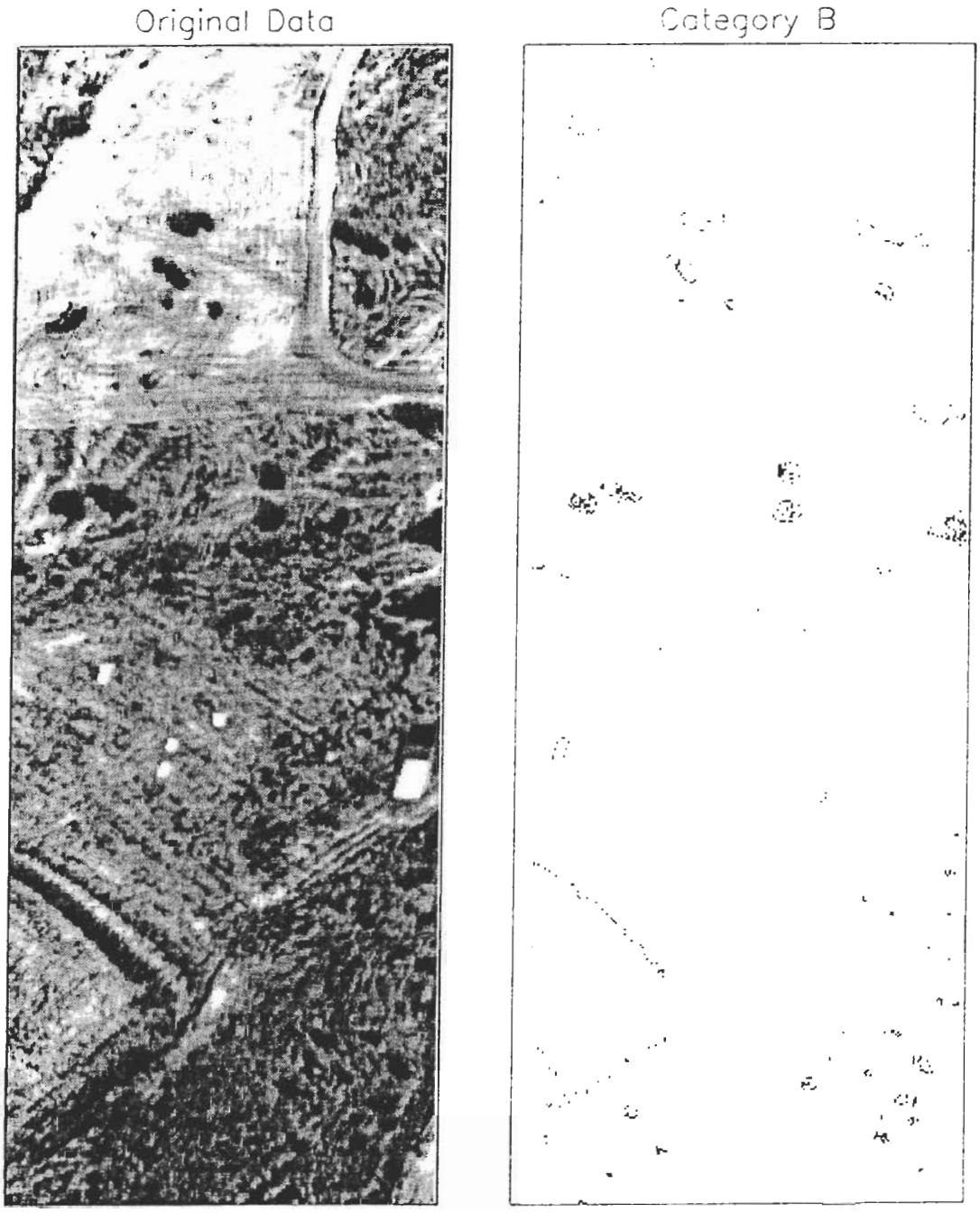

Figure 4.26 Network Category B for Desert Radiance Data, Shrubs

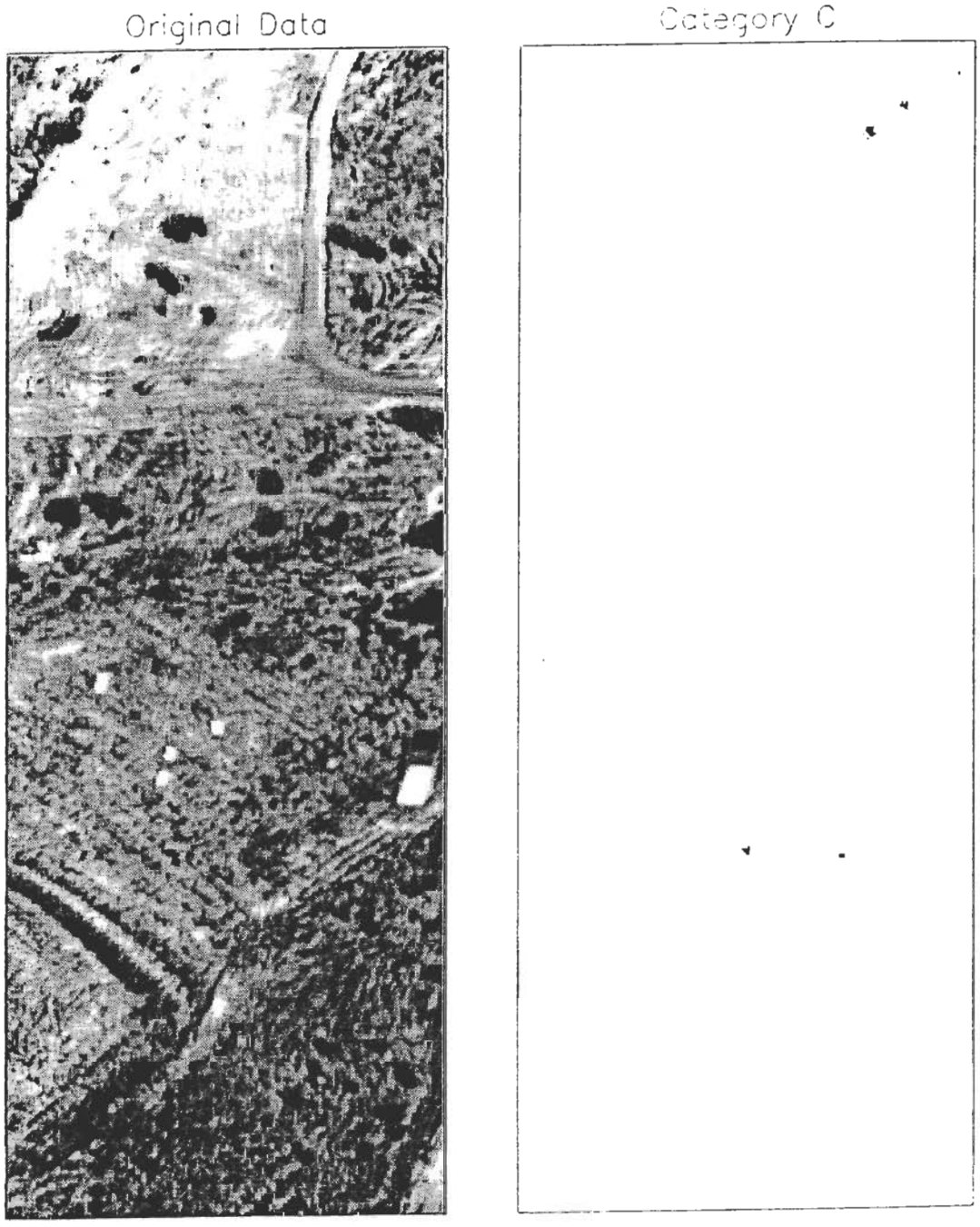

Figure 4.27 Network Category C for Desert Radiance Data, Target Panels

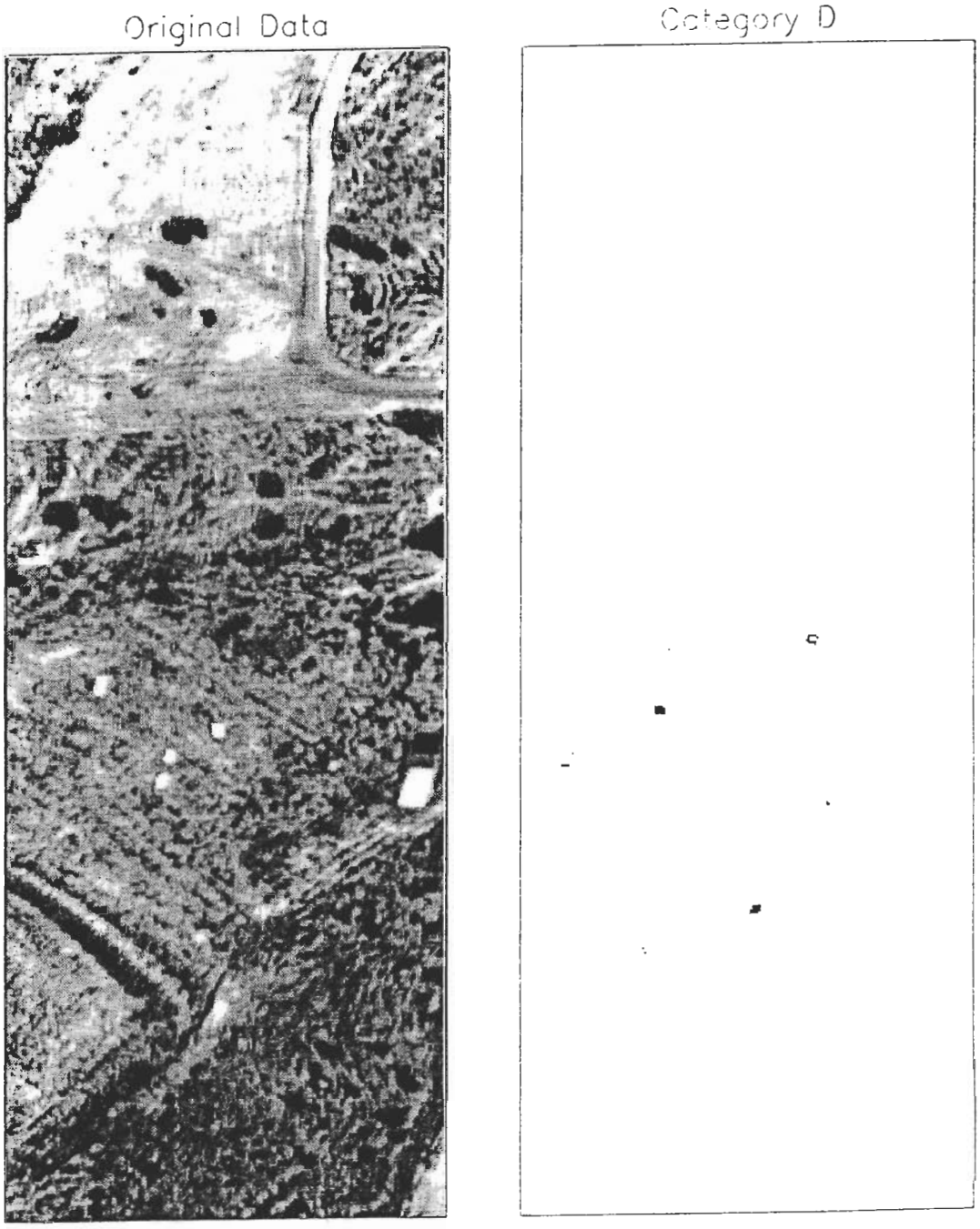

Figure 4.28 Network Category D for Desert Radiance Data, Target Panels

APPENDIX B

```
;&&&&&&  to be used with bsq data cube    &&&&&&&&&&&&&&&&&&&&
;&&&&&&  for hyperspectral imagery  &&&&&&&&&&&&&&&&&&&&&&

set_plot, 'x'            &    device, pseudo = 8, retain = 2

; initialize network size
maxgdn = 4   &   maxgde = 4   &   maxit = 5e3   ; adjust maxitr.....
xittr = 0

xsize = 99            & ysize = 300          &  zsize = 100
nvar  = zsize         & ncase = xsize * ysize
x2    = fltarr(nvar+1,ncase+1)        &  xm     = fltarr(nvar)
wei   = fltarr(ncase)                 &  values = fltarr(nvar)
list  = intarr(xsize,ysize)           &  icl    = intarr(ncase)
num   = intarr(ncase)
g = fltarr(nvar, maxgde, maxgdn)

 file2 = 'data.cube'    ; data file is arranged as bsq
 file7 = 'results.dat'

print, ' input  file ',  file2
print, ' output file ',  file7
print, 'Number of cases to be read: ', ncase
print, 'Number of variables to be read: ', nvar
print, 'Two Dimensional SOM has ', maxgde,' rows and ',  maxgdn,' columns'
print, 'Number of training cases ', maxit

; neighborhood size function
lat = 0
if (lat eq 0) then print, 'simple block party used'
if (lat ne 0) then print, ' Gaussian lateral inhibition function used'
```

```
;** set some constants to do with training to some arbitrary values
alpha0 = 0.4 &   alpha = alpha0
tmp = [maxgde, maxgdn]
dstep0 = max(tmp)/2.5    &   dstep = dstep0
dmaxtr = maxit

;****  initialize means
xm(*) = 0.0   &   idim = 1   &   idum = 0

;***  read data

irc = 1                    &    weight = fltarr(ncase)
camo = intarr(xsize,ysize,zsize)
openr, 3, file2
  forrd, 3, camo
close, 3

t   = total(camo,3)/100
x2  = reform(t,xsize*ysize)
x   = camo

for i = 0, 99 do begin
  tvscl,x(*,*,i)
endfor

weight = .6

print, ' all data read ok '

;****************************************
```

```
; calculate variable means
  xm = total(x2)/ncase

seed = 0.0
; initialize map weights

tmp = 0.01*xm*randomu( seed, nvar, maxgde, maxgdn)

for k = 0, nvar-1 do begin
   g(k,0:maxgde-1,0:maxgdn-1) = xm + tmp(k,0:maxgde-1,0:maxgdn-1)*xm/3.0
endfor
plot,g            &   wait, 1            & erase

;  other inits and form cumulative sampling sums
icl(*) = 0   &   wei(*) = 0

;  start iterations****************************
movers = 0            & kik = 0            & nup = 0
sumsam = wei(ncase-1)
dcon = 2.0/(dstep*dstep)   &   print, dstep,dcon

aver = 0.0            &   n=0

;*******************************************************************
;****************** the training process starts here **********************

for iter1 = 0L, maxit-1 do begin
  ; select training case at random but proportional to weighting:
n= n + 1
kk = intarr(xsize)            &  kk2 = intarr(ysize)

k=fix((xsize-1)*randomu(s))        ; index into x-dim of data
k2=fix((ysize-1)*randomu(s))       ; index into y-dim of data
```

```
values = x(k,k2,0:nvar-1)          ; randomly chosen spectrum

; locate the neuron closest to the randomly chosen input spectrum
; use correlation coefficient to determine winner

 best = .001              ; compare to each neuron
for j = 0, maxgde-1 do begin
for k = 0, maxgdn-1 do begin

 sum = 0L  &    mag_g = 0L  &    mag_v = 0L
  a = g(*,J,K)    & a = reform(a)            ; make sure vector is 1-D
  b = values(*)   & b = reform(float(b))
  c = a*b
  sum = total(c)  &     mag_g = sqrt(total(a*a)) &   mag_v = sqrt(total(b*b))
  corr = sum/(mag_g*mag_v)

 if (corr gt best) then begin
   best = corr   &   indj = j  &   indk = k
 endif

endfor
endfor
;******** now have location of neuron which most closely equals the data *****
aver = aver + best
; set classification

 new = (indj -1)* maxgde + indk
if (new ne icl(k*k2) )   then begin
  movers = movers +1
  icl (kk) = new
endif
```

```
; set distance threshold for block neighborhood
dsq = dstep * dstep              &    dcon = 2.0/dsq

; now update best matched neuron AND its neighbours
for j = 0, maxgdn-1 do begin
for k = 0, maxgde-1 do begin
 ;  calculate distance from winning neuron
 dis = (j-indj)^2 + (k-indk)^2

 ;  compute weights due to lateral inhibition
 if (lat eq 0) then begin
    if (dis gt dsq) then goto, c200
    syn = alpha
 endif else begin
    gaus = dis/dcon
    if (gaus gt 20.0) then begin
      gaus = 0.0
    endif else begin
      gaus = exp(-gaus)
    endelse
 endelse

; update weights
  if ( syn ne 0.0) then begin
    nup = nup + 1
    for L = 0, nvar -1 do begin
      G(L,K,J) = G(L,K,J) + syn * (values(L) - G(L,K,J) )
    endfor
  endif

c200:
endfor
endfor
```

```
; print something every now and again
kik = kik +1
 if(kik eq 1e2) then begin
   aver = aver/1e4
   plot, g, title = 'Iteration ' + string(iter1), ytitle = 'weights'
   print, 'iteration Number: ', iter1
   print, 'Moves: ', movers, '        Dstep: ',   dstep
   print, 'alpha: ', alpha,  '   Updates: ', nup
   print, 'correlation ', best
   movers = 0 & kik = 0  & aver = 0.0 & nup = 0
 endif

;  change training parameters
;  there will be a dramatic change in the training process when
;  the neighborhood shrinks to the point where single neurons are
;  being trained

 alpha = alpha0* (1.0 - iter1/dmaxtr)
 dstep = dstep0* (1.0 - iter1/dmaxtr)

endfor     ; iter1 loop at top

; **** end of training*************************************************

print, 'training is completed '

set_plot, 'PS'
  device, file='weights.ps', /portrait, bits_per_pixel=8, /close_file, /inches, $
    yoffset = 2.5, ysize = 6.0, xoffset = 1.25, xsize = 6.0
  plot, g, xtitle = 'Weights', ytitle = 'Intensity', title = 'Network Weights'
device, /close_file
```

```
;**************** classify data ************************************

; sequence through each pixel

  for i = 0, xsize-1 do begin
  for w = 0, ysize-1 do begin

  best = .001
; sequence through each neuron
   for j = 0, maxgde -1 do begin
   for k = 0, maxgdn -1 do begin

  sum = 0L  &    mag_g = 0L  &    mag_v = 0L
    a = g(*,J,K)  & a = reform(a) & b = x(i,w,*) & b = reform(float(b))
    c = a*b
    sum = total(c)
   mag_g = sqrt(total(a*a))   &     mag_v = sqrt(total(b*b))
   corr = sum/(mag_g*mag_v)

 if ( corr gt best) then begin
  best = corr    &   indj = j & indk = k
 endif

endfor
endfor
;************* save result    **************
 ; list is an array that has the same size as the original image
 ; the values stored in list are the categories for each pixel element
 list(i,w) = (indj) * maxgde + indk+1

endfor   ;  end of the i loop
print,'categorizing column: ', i, '  of ', xsize
endfor  ; end of the w loop
;************** done with categorization**************
```

```
print, 'done with categorization'


list=50*list    ; a little redundant, with tvscl next....
tmp = size(list)
set_plot, 'x'
window, 1, xsize = tmp(1) + 40 , ysize = tmp(2) +60, title = 'results'
tvscl,list, 20, 30
loadct, 13

set_plot, 'PS'
  device, file='results.ps', /portrait, /close_file, /inches, $
    yoffset = 2.5, ysize = 6.0,   xoffset = 3.25, xsize = 6.0, $
    bits_per_pixel=8
  tvscl,list
device, /close_file

openw, 7 , file7
 printf, 7 , list
close,7

set_plot, 'x'

end
```

LIST OF REFERENCES

Aleksander, I. and Morton, H., 1990. *An Introduction to Neural Computing,* Chapman and Hall

Brown, J.R. and DeRouin, E. E., 1992. "Comparing neural network classifiers and feature selection for target detection in hyperspectral imagery" in *Proceedings: Applications of Artificial Neural Networks III* , April 1992

Elachi, C. 1987. *Introduction to the Physics and Techniques of Remote Sensing,* John Wiley and Sons, New York

Fay, M. E., 1995. "An Analysis of Hyperspectral Imagery Data Colledted During Operation Desert Radiance", NPS, Montery, CA

Haykin, S., 1994. *Neural Networks: A Comprehensive Foundation,* Macmillan College Publishing Company

Hebb, D. O., 1949. *The Organization of Behavior,* Wiley, New York

Klein, M. V., 1970. *Optics,* John Wiley and Sons, New York

Kohonen, T., 1988. *Self-Organization and Associative Memory,* 3rd ed. New York: Springer-Verlag

Lucey, P., Williams, K., Hinck, K., Budney, C., Rafert, J., and Rusk, T. 1993. "A Cryogenically Cooled Spatially Modulated, Imaging, Fourier Transform Spectrometer For Remote Sensing Applications"

McCulloch, W. S. and Pitts, W., 1943. "A Logical Calculus of the Ideas Immanent in Nervous Activity", in *Bulletin of Mathematical Biophysics,* Vol 5 1943

Minsky, M. and Papert, S., 1969. *Perceptrons: An Introduction to Computational Geometry,* MIT Press, Massachusetts

Nasrabaldi, N. M., 1995. Lecture Notes, Short Course offered at the IS&T/SPIE Simposium on ELECTRONIC IMAGING SCIENCE & TECHNOLOGY, San Jose, Ca.

Openshaw, S., 1994. "Neuroclassification of Spatial Data", in *Neural Nets: Applications in Geography* (Hewitson and Crane), Kulwer Academic Publishers, London

Otten, J. L., Silco,T. L., Rafert, J. B., Sellar, R.G. and Gautreaux, M. M., 1995. "Hyperspectral Measurements of Common Camouflages"

Rinder, J. R., 1990. "Hyperspectral Imagery - What Is It? - What Can It Do?", Presented at the USACE Seventh Remote Sensing Symposiom

Rosenblatt, F., 1962. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanics,* Spartan Books, New York

Shen, S. S. and Horblit, B. D. "Application of neural networks to hyperspectral image analysis and interpretation" in *Proceedings: Applications of Artificial Neural Networks III* , April 1992

Vane, G., 1985. "High Spectral Resolution Remote Sensing of the Earth", *Sensors,* December,1985

Widrow, B. and Hoff, M. E., 1960. "Adaptive Switching Elements" in *IRE Wescon Convention Record* p 96-104

Westervelt, J., Krzysik, A. and Seel, K., 1995. "A Basic Introduction To Neural Networks"

INITIAL DISTRIBUTION LIST

	Number of Copies
1. Defince Technical Information Center Cameron Station Alexandrea, Virginia 22304-6145	2
2. Library, Code 52 Naval Postgraduate School Monterey, California 93943-5002	2
3. Richard C. Olsen Department of Physics Naval Postgraduate School Monterey California 93943-5002	5
4. Don Walters Department of Physics Naval Postgraduate School Monterey, California 93943-5002	1
5. LT Mark M Gautreaux 12400 Jefferson Hwy Apt 209 Baton Rouge, Louisiana 70816	1
6. John Otten Kestrel Corp. 6020 Academy Blvd NE Suite 104 Albuquerque, NM 87109	1
7. Hydice Program Office Naval Research Laboratories Code 9120, Bldg A59 4555 Overton Ave SE Washington DC 20375-5320	1

For large T and stationary E fields the integral is independent of t_o . This is equivalent to taking the limit as $T \rightarrow \infty$.

The terms $\langle E(t)^2 \rangle$ and $\langle E(t-\tau)^2 \rangle$ represent the respective intensities received in the beams from M_1 and M_2, and for a steady source are each equal to $\frac{1}{2} I_o$, where I_o is the total intensity at the detector if interference does not occur. The term $2\langle E(t)E(t-\tau) \rangle$ contains all the interference effects. If it is normalized by dividing by I_o, and we define

$$\gamma(\tau) = \frac{\langle E(t)E(t-\tau) \rangle}{\langle E(t)^2 \rangle} \tag{3.14}$$

then we have

$$I_d = I_o[1 + \gamma(\tau)] \tag{3.15}$$

This last equation has the same form as Eq. 3.9 and is deliberately written using the same function $\gamma(\tau)$. The two definitions of $\gamma(\tau)$, Eqs. 3.10 and 3.14, are equivalent (Klein, 1970 offers a rigorous proof of this in Appendix B). When written in the form of Eq. 3.14 $\gamma(\tau)$ is called the normalized autocorrelation function of the signal $E(t)$. It describes the mean correlation of E with itself at an earlier time. (Klein, 1970)

The equivalence of the two expressions for $\gamma(\tau)$, namely the Equation

$$\frac{\langle E(t)E(t-\tau) \rangle}{\langle E(t)^2 \rangle} = \int_0^\infty P(\nu) \cos 2\pi\nu\tau \, d\nu \tag{3.16}$$

is called the *Wiener-Khintchine theorem.* It says that the normalized autocorrelation function of a signal is the cosine Fourier transform of its spectral distribution function. The spectral distribution of a signal can be obtained by measuring its electric field for some period of time.