DETECTION AND CLASSIFICATION OF ULTRASONIC ECHOES USING NEURAL NETWORKS

BY

MEHMET SULEYMAN UNLUTURK

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
Illinois Institute of Technology

Approved Jafar Saniie
Adviser

Chicago, Illinois
December 1997

ACKNOWLEDGMENT

The author wishes to express his sincere gratitude to his major professor, Dr. Jafar Saniie for his guidance and assistance through all phases of this research. Dr. Saniie contributed a substantial amount of his time and effort on this thesis, and his help is greatly appreciated.

The author is very grateful to Dr. G. Williamson, Dr. G. Atkin, and Dr. P. Greene, who devoted the time to serve as the thesis committee.

This thesis is dedicated to author's wife, Sevcan Unluturk, and author's daughter, Buse Unluturk.

M.S.U.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

Figure | Page

Figure Page

Figure Page

Figure Page

ABSTRACT

Ultrasonic imaging techniques have been widely used for industrial and medical applications. There have been several challenging problems involved in these techniques such as detection of multiple interfering target echoes (e.g. related to cracks, defects, multiple layers) in the presence of scattering noise, and classification of grain echoes in order to characterize the materials nondestructively. Conventional imaging techniques lack the capability of resolving such echoes which are closely located in time and frequency domains in the presence of scattering noise. Neural networks are powerful tools for overcoming this challenging task due to their trainability and adaptability capabilities. This thesis presents neural network models to detect and characterize multiple target echoes in close proximity of each other for material evaluation. The neural network models are attractive, as they do not require any solution methodologies, or any mathematical models of the scattering functions in advance. Several problems dealing with ultrasonic imaging systems have been fully explored utilizing neural networks: i) deconvolution neural networks as a mean of detecting target echoes in the presence of grain scattering noise, ii) grain power spectrum neural networks as a mean of characterizing flaw echoes to classify different type of materials, iii) neural network filters as a mean of order statistic processing of multi-channel scattering signals.

In order to detect flaw echoes, deconvolution methods using neural networks are developed. Three novel design procedures have been developed in implementing deconvolution using neural network algorithms. The first method is called the deconvolution neural network (DNN), the second method is named the autoassociative deconvolution neural network (ADNN), and the third method is referred to as the probabilistic deconvolution neural network (PDNN). The DNN trains the network by employing the brute force and by exposing the network to a set of target echoes with and without noise. The ADNN processes the data for signal-to-noise ratio enhancement using an autoassociative neural network, and then applies the DNN. The PDNN consists of two processing stages. The first stage estimates parameters using Gram-Charlier approximation to describe the probability density functions corresponding to target echoes and scattering noise. Then, in the second processing block, these parameters are used to classify and detect multiple target echoes. Results obtained in the performance analysis of these algorithms indicate that multiple target echoes can be deconvolved and resolved accurately in the presence of noise.

A well-known method detecting flaw echoes in large grains is to utilize split-spectrum processing coupled with order statistic filters. A procedure has been developed utilizing neural networks to achieve sorting processing. In particular, minimum order statistic neural network (MinNNet), median order statistic neural network (MedNNet) and maximum order statistic neural network (MaxNNet) have been fully explored. These neural network filters find the minimum, the median or the maximum of input data respectively. Such neural networks can be used in sorting split-spectrum backscattered echoes in order to detect flaw in high grain scattering noise. Both simulated and experimental results indicate that neural network order statistic filters offer desirable performance on sorting data and detecting flaw echoes.

A design procedure for a novel application of neural networks has been developed to discriminate the frequency signatures inherent to ultrasonic microstructure scattering signals consisting of multiple unresolvable echoes of random amplitude and arrival time. This method is called the grain power spectrum neural network (GPSNN) which is trained to classify grain scattering signals for the nondestructive testing of materials. The materials tested for grain size discrimination are steel examples with grain sizes of 14 and 50 microns. The experimental grain signals are obtained using a broadband transducer with a 6.22 MHz center frequency. The GPSNN has 32 input nodes, 13 hidden neurons determined adaptively, and one summing output node. The adaptive hidden neuron algorithm avoids problems of overfitting or underfitting. In addition, Hinton diagrams have been utilized to display the optimality of GPSNN weights. A set of 4,490 training sequences is utilized to train the neural network. A new set of 12,572 testing sequences is acquired to test GPSNN performance. A comparative study of GPSNN with other designs of neural networks using ultrasonic scattering sequence and autocorrelation are also examined. Overall, GPSNN achieves an average recognition performance of over 98%. This high level of recognition suggests that the GPSNN is a promising method for ultrasonic nondestructive testing. Furthermore, this method is applicable to tissue characterization in ultrasonic medical imaging.

Based on analytical and experimental observations, one can conclude that backpropagation neural network models are encouraging and potentially useful techniques for nondestructive testing and quality control. Results presented through the thesis clearly suggest that neural networks can be used as an effective means for ultrasound signal processing.

CHAPTER I

INTRODUCTION

1.1 Ultrasonic Imaging System and Neural Networks

The detection of closely located target echoes in large-grained materials and the characterization of these echoes for material evaluation are important tasks in nondestructive testing [61, 63, 64, 66, 82, 83]*. Conventional imaging techniques lack the capability of resolving such echoes that are closely located in time and frequency domains, especially in scattering noise. Nonlinear imaging techniques like neural networks are important and useful tools for detection and characterization of such ultrasonic signals [12, 38, 65, 78, 79]. This thesis presents neural network models using backpropagation learning algorithm to detect multiple interfering target echoes, and characterize these echoes for material evaluation [65, 78, 79]. The neural network models are attractive because they do not require any solution methodologies or mathematical models of the scattering functions. Once the neural network is trained, decision on testing data is a real-time processing operation which is highly desirable in ultrasonic applications. In this investigation, adaptive hidden neuron algorithm is introduced to improve the performance of the backpropagation learning algorithm [65, 78, 79]. This objective is achieved in the decision process of finding the optimal number of hidden neurons. Hinton diagrams are employed to explain why these neural networks work successfully for the problems of ultrasound imaging systems [49]. The statistical

* Corresponds to numbered references in the bibliography.

properties of hidden neurons were examined further in order to analyze the appropriateness of neural networks for order statistic signal processing.

In order to detect target echoes in noisy environments, neural networks are used to implement the deconvolution methods [79]. In a linear time-invariant system, the output, y(n), is related to the input, x(n), by impulse response function of the system, h(n), using the convolution operation: $y(n) = h(n) * x(n)$. The deconvolution process is defined as finding a good estimate of h(n) from the knowledge of y(n) and x(n). The solution to this problem in frequency domain is given as $H_{est}(\omega) = \frac{Y(\omega)}{X(\omega)}$. The transfer function $H_{est}(\omega)$ is unbounded where $X(\omega)$ tends to zero. In this study, several methods based on utilization of deconvolution have been employed to solve this type of problem. One solution to this problem is to use neural networks that behaves like a deconvolution filter where the output is zero in the absence of a target and unity impulse in the presence of a target. These neural networks prevent the estimated transfer function $H_{est}(\omega)$ from being unbounded. This type of neural networks is stable, since it has the advantage of using time-domain convolutions rather than the frequency-domain divisions which is influenced by bad zeros of $X(\omega)$ and bad poles of $Y(\omega)$.

In this study, three neural network models have been developed in implementing deconvolution using backpropagation learning algorithm. The first network model is called the deconvolution neural network (DNN), the second network method is named the autoassociative deconvolution neural network (ADNN), and the third network design is referred to as the probabilistic deconvolution neural network (PDNN) [65, 73, 78, 79]. The target echoes with and without noise were included in the training phase of DNN.

The purpose of the training was to teach the DNN to give an output value of zero when the target is not presented or unity impulse when the target is presented as an input. The ADNN improves the signal-to-noise ratio of the noisy signal using an autoassociative neural network (ANN), and then applies that improved signal to DNN for the detection of target echoes. The PDNN consists of two processing stages. The first stage estimates the Gram-Charlier coefficients of the target echoes and the scattering noise [39, 79]. And the second processing block uses these parameters to detect multiple interfering target echoes. Experimental results show that multiple target echoes which are close to each other can be deconvolved and determined correctly in the presence of scattering noise.

Another solution for detecting target echoes in grain scattering noise is to use the split-spectrum processing with order statistic filters. Order statistic filtering has been extensively employed in signal and image processing areas [63, 64, 82]. Some of the order statistic filters that have been utilized widely in ultrasonic imaging systems are minimum, median, and maximum order statistic filters. These filters find the minimum, median, and maximum element in an input signal respectively [46, 87]. One of our objectives in this thesis is to develop and analyze the performance of three neural network order statistic filters to replace the conventional ones that are used in the split-spectrum processing. These neural network models are called minimum order statistic neural network filter (MinNNet), median order statistic neural network filter (MedNNet) and maximum order statistic neural network filter (MaxNNet). These filters find the minimum, the median or the maximum of input data respectively. Such neural network filters can be employed in sorting backscattered echoes in order to detect flaw echoes in noisy environments. In the design process of the neural network filters, backpropagation learning algorithm and adaptive hidden neuron algorithm were used. If we assume that there are m different input values need to be sorted by these neural networks, there are m! (very large number) different signal patterns of input that give the same output. Therefore one may expect that neural networks may not offer highly accurate results. In spite of this disadvantage, neural network filters provide good approximation which appears to be ultrasonic flaw detection from statistical point of view. The training set of data consists of uniform random numbers as an input to the neural network and the output contains of minimum, median and maximum rank of these random numbers for MinNNet, MedNNet, and MaxNNet respectively. After the neural networks were trained, we used the neural network weights to derive the probability density function of the output to find out if the optimal solution has been reached or not. This pdf gives us some analytical point of view why the neural network filters are capable of sorting the input data. Another method to explain the operation of these neural networks is to utilize Hinton diagrams to examine the neural network weights [49]. Experimental results indicate that order statistic neural network filters can be used to sort the input data and detect the target echoes in noisy environments.

The importance of evaluating the microstructure of materials ultrasonically has been long recognized. In particular, it is highly desirable to estimate grain size or classify materials based on the scattering properties of their microstructure [63, 82, 83]. Backscattered grain echoes are random signals that bear information related to both the grain size and frequency of sound. In ultrasonic grain size characterization a model for the grain signal consists of the convolution of components representing the contribution of the measuring system impulse response (i.e., the interrogating ultrasonic wavelet) and the grain scattering function. This function contains information related to many random physical parameters such as grain size, shape, orientation, boundary characteristics, and chemical constituents. Consequently, the grain scattering signal becomes random and exhibits a great deal of variability in the time domain. Therefore, spectral analysis is often adopted as an alternate method for signal characterization [46, 55, 87].

In the Rayleigh scattering region (the wavelength, λ, is larger than the average grain diameter, A) the scattering coefficients vary with the third power of the grain diameter and the fourth power of the frequency, while the absorption coefficient increases linearly with frequency [46]. The high frequency component of the interrogating ultrasonic wavelet backscatters with higher intensity than the lower frequency components. This situation results in a higher expected frequency than that of the original interrogating wavelet. In this study, we have developed the design procedure for a neural network to discriminate the frequency signatures inherent in ultrasonic grain scattering signals. This method, called the grain power spectrum neural network (GPSNN), offers practical advantages such as real-time processing, adaptability and training capability. GPSNN deduces the relationship between the measurement power spectrum and classification output without knowing the scattering model, physical parameters, or the solution methodology. With the neural network, as each set of input vectors is applied to the neural network, the hidden layers configure themselves to recognize certain frequency features of the input vectors related to the scattering properties of the materials. After the GPSNN is fully trained, each hidden neuron represents certain frequency characteristics of the total input space. Therefore, when the power spectrum of a new grain scattering signal is applied to GPSNN, each neuron is able to respond to the presence of a particular subset of frequency information which it was trained to recognize. The GPSNN has 32 input nodes, 13 hidden neurons determined adaptively, and one summing node. A set of 4,490 training sequences is utilized to train the neural network. A new set of 12,572 testing sequences is used to test GPSNN performance. The samples tested for grain size discrimination are steel with grain sizes of 14 and 50 microns. GPSNN achieves an average recognition performance of over 98% which tells us that GPSNN is an effective method in nondestructive testing. Furthermore, GPSNN can be adopted in ultrasonic medical imaging problems such as tissue characterization.

All these novel neural network models are encouraging and suggest that they are potentially useful for nondestructive testing and quality control. Overall, neural networks can be used as an effective means for ultrasound signal processing. Next section provides the theoretical background for backpropagation learning algorithm.

1.2 Backpropagation Learning Algorithm

The supervised backpropagation learning algorithm is the most broadly accepted learning technique for design of neural networks [15, 21, 49]. This algorithm is used to train the deconvolution neural networks, the grain power spectrum neural network, and the order statistic neural network filters. These neural networks perform a distinguished nonlinear mapping which can be stated in terms of a given input/output data sets. These data sets are called the learning examples.

A neural network structure is given in Figure 1.1. In backpropagation learning algorithm, after an input pattern, x_{pi} (i = 1, ..., N), is applied to the input layer, it is propagated through upper layers until an output is generated at the output layer. Then it is compared to the desired output, and the error signal is computed for each output unit. The error is propagated backward to nodes in the intermediate layers. However the effect of the error propagated on nodes depends roughly on the contribution node made to the total error. Based on the error signal received, connection weights, w_{ji}, are updated for convergence for all the training patterns. The significance of this process is that, as the network trains, nodes in the intermediate layers organize themselves to recognize different features of the total input space. As a result all hidden-layer units in the neural network are associated with specific features of the input pattern. A neural network is called a mapping network if it is able to compute some functional relationship between its input and output. It is an advantage to have a system like this, because if we do not know how to describe the functional relationship between the input and the output in advance, neural networks can achieve this information by utilizing examples of the correct mapping [15, 21, 49, 65, 78, 79].

A set of p-vectors are examples of a functional mapping

$$y = f(x): x \in \Re^N, y \in \Re^M. \qquad (1.1)$$

and set of p-vectors are given as

$$(x_1^T, y_1^T), (x_2^T, y_2^T), \ldots, (x_p^T, y_p^T),$$

$$x_i^T = [x_{i1}, x_{i2}, \ldots, x_{iN}], y_j^T = [y_{j1}, y_{j2}, \ldots, y_{jM}], \qquad (1.2)$$

The output of each neuron at the hidden layer can be calculated as follows

$$net_{pj}^h = \sum_{i=1}^{N} w_{ji}^h x_{pi}, \qquad (1.3)$$

where N is the total number of input neurons, w_{ji}^h is the weight from i th input node to the j th hidden node.

Activation function of this node is given as

$$i_{pj} = F_j^h(net_{pj}^h). \qquad (1.4)$$

where $F^h_j(\bullet)$ is hyperbolic tangent activation function and given in Figure 1.2. The hyperbolic tangent activation function is defined as

$$F^h{}_j(n) = \tanh(n). \qquad (1.5)$$

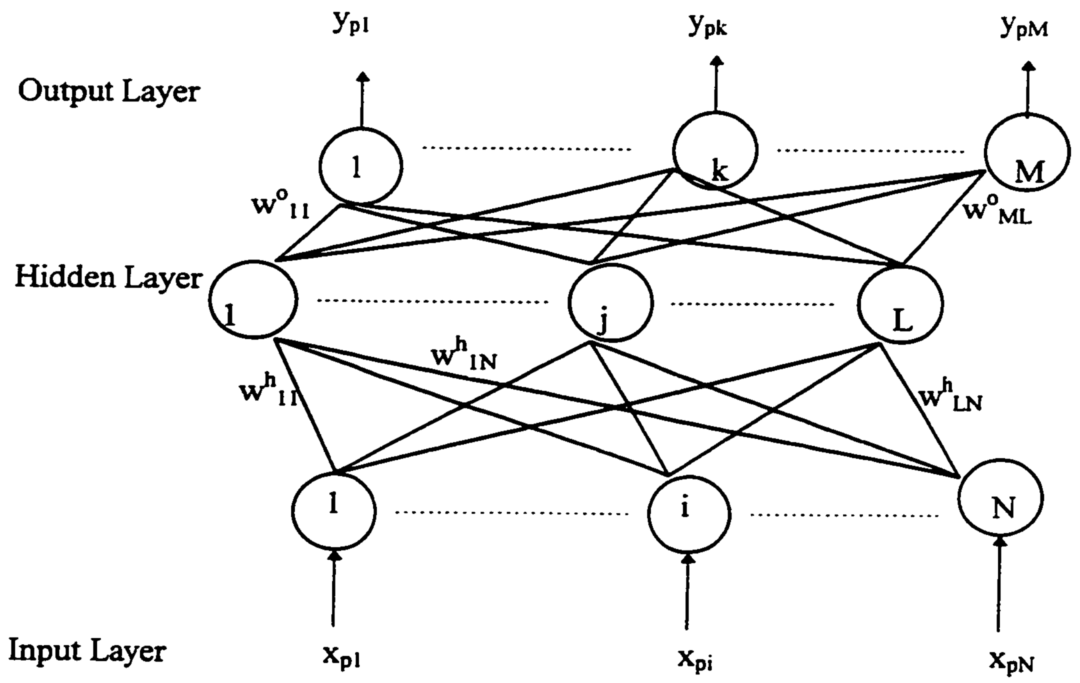


Figure 1.1 A General Neural Network Structure

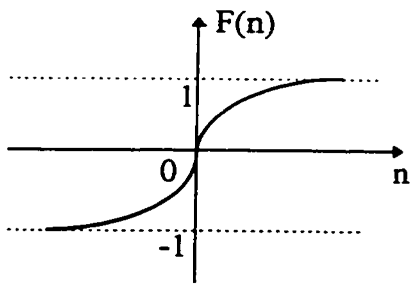


Figure 1.2 Hyperbolic Tangent Activation Function

$$net_{pk}{}^{o} = \sum_{j=1}^{L} w_{kj}{}^{o} i_{pj}, \tag{1.6}$$

where L is the total number of hidden neurons and the activation of this node is

$$a_{pk} = F_k{}^{o}(net_{pk}{}^{o}). \tag{1.7}$$

where $F^o{}_k(\bullet)$ is linear activation function and given in Figure 1.3. The linear activation function is defined as

$$F^o{}_k(n) = n. \tag{1.8}$$

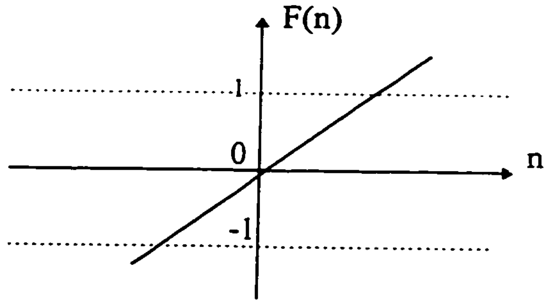


Figure 1.3 Linear Activation Function

The following presents steps for training the neural network:

1. Apply an input vector to the network and calculate the corresponding values.

2. Compare the actual outputs with the correct outputs and determine a measure of the error.

3. Determine in which direction (+ or -) to change each weight in order to reduce the error.

4. Determine the amount by which to change each weight.

5. Apply corrections to weights.

6. Repeat items 1 through 5 with all the training vectors until the error for all vectors in the training set is reduced to an acceptable value.

In order to update the output layer weights, we must calculate the error, E_p, as the following

$$E_p = \frac{1}{2}\sum_{k=1}^{M} \rho_{pk}^{2},$$

$$\rho_{pk} = (y_{pk} - a_{pk}), \tag{1.9}$$

where M is the total number of output neurons, E_p is the p th error, y_{pk} is the desired output and a_{pk} is the actual output for the k th output neuron and p is the number of patterns in the training set.

To determine the direction in which to change weights, we calculate the negative of the gradient of E_p with respect to the weight, w_{kj}.

$$\frac{\partial E_p}{\partial w_{kj}^{o}} = -(y_{pk} - a_{pk}) F_k^{o'}(net_{pk}^{o}) i_{pj}. \tag{1.10}$$

$$w_{kj}^{o}(m+1) = w_{kj}^{o}(m) + \eta (y_{pk} - a_{pk}) F_k^{o'}(net_{pk}^{o}) i_{pj}. \tag{1.11}$$

where η is called the learning rate parameter. One can see from the Equation (1.11) that the transfer function, F, must be differentiable. For that reason, we use the linear activation function at the output layer. Figure 1.3 shows a linear activation function.

The error given in Equation (1.9) should be related to the output values on the hidden layer. From Equations (1.3) and (1.4), i_{pj} depends on the weights of hidden layer.

$$\frac{\partial E_p}{\partial w_{ji}{}^h} = \frac{1}{2}\sum_{k=1}^{M}\frac{\partial (y_{pk} - a_{pk})^2}{\partial w_{ji}{}^h} \tag{1.12}$$

By utilizing the chain rule on Equation (1.12), one gets

$$\frac{\partial E_p}{\partial w_{ji}{}^h} = -\sum_{k=1}^{M}(y_{pk} - a_{pk})\frac{\partial a_{pk}}{\partial net_{pk}{}^o}\frac{\partial net_{pk}{}^o}{\partial i_{pj}}\frac{\partial i_{pj}}{\partial net_{pj}{}^h}\frac{\partial net_{pj}{}^h}{\partial w_{ji}{}^h}, \tag{1.13}$$

We substitute Equations (1.3) through (1.8) in Equation (1.13) to get relationships between updates of the hidden weights and the error propagated back

$$\Delta(w_{ji}{}^h) = \eta F_j^{h'}(net_{pj}{}^h)x_{pi}\sum_{k=1}^{M}(y_{pk} - a_{pk})F_k^{o'}(net_{pk}{}^o)w_{kj}{}^o. \tag{1.14}$$

The known errors on the output layer are propagated back to the hidden layer to determine appropriate weight changes on this layer

$$\rho_{pj}^{\ h} = F_j^{\ h'}(net_{pj}^{\ h})\sum_{k=1}^{M}(y_{pk} - a_{pk})F_k^{\ o'}(net_{pk}^{\ o})w_{kj}^{\ o}$$

$$w_{ji}^{\ h}(m+1) = w_{ji}^{\ h}(m) + \eta\rho_{pj}^{\ h}x_{pi}. \tag{1.15}$$

For the hidden layer activation function, a hyperbolic tangent activation function is chosen. Figure 1.2 shows the hyperbolic tangent activation function. There are some other activation functions such as log-sigmoid activation functions that can be used instead of hyperbolic tangent functions. Figure 1.4 shows a log-sigmoid function. Log-sigmoid function is defined as

$$F(n) = \frac{1}{1+e^{-n}}. \tag{1.16}$$

From Figure 1.4, one can see that the neural network hidden neuron outputs are limited with positive values between 0 and 1. In the field of ultrasound signal processing, we need negative activation values besides the positive values to detect the target echoes.

The process of choosing the right activation function for the hidden layers is problem dependent. Some promising techniques which greatly improve the performance of the neural network are given in Chapter 2. It appears that a neural network trained with these techniques not only allows an escape from a local minimum, but also has a better ability to recognize noisy target echoes that have never been presented to the neural network during the training process.

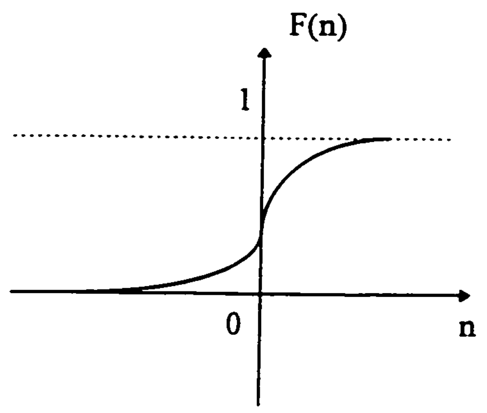


Figure 1.4 Log-sigmoid Function

1.3 Preview of the Remaining Chapters

The significance of examining the microstructure of materials ultrasonically has long been recognized. Detection of flaws and classification of materials based on their scattering properties are the issues that have been investigated in this thesis using neural networks. Chapter 2 begins this investigation by introducing the deconvolution neural networks to detect multiple target echoes which are interfered with each other in noisy environments. These neural networks were trained with one, two and three echoes with and without noise. Performance evaluation of these neural networks was further investigated through simulated echoes. To enhance the performance of deconvolution neural networks, autoassociative neural networks have been developed. These neural networks are utilized to improve the signal-to-noise ratio, and then the improved signal is applied to DNNs. The adaptive hidden neuron algorithm is introduced to find out the optimum number of hidden neurons. Experimental results that are depicted in Chapter 2 indicate that the adaptive hidden neuron algorithm improves the backpropagation learning algorithm and causes it to escape from the local minimum and reach the global minimum.

Chapter 3 approaches the deconvolution problem by estimating the Gram-Charlier coefficients of the target echoes and the scattering noise. The Gram-Charlier coefficients are utilized to construct the probabilistic deconvolution neural networks to achieve a high probability of detection for a reasonably low signal-to-noise ratio.

Chapter 4 presents another technique that can be used for detection of target echoes. This technique is called split-spectrum processing. The purpose of Chapter 4 is to develop minimum, median, and maximum order statistic neural network filters to replace the conventional ones in split-spectrum processing. Hinton diagrams and probability density functions of the output neurons are utilized to explain why these neural network filters can be successfully used in split-spectrum processing.

A novel neural network structure has been investigated to discriminate the frequency signatures of ultrasonic microstructure scattering signals in Chapter 5. Chapter 5 starts with exploring the scattering functions for ultrasonic imaging systems and introduces the grain power spectrum neural network to differentiate two different grain signals using their frequency signatures. Finally, Chapter 6 summarizes the thesis work and introduces the future work.

CHAPTER II

DECONVOLUTION NEURAL NETWORKS FOR ULTRASONIC TESTING

This chapter presents two novel neural network models in order to implement deconvolution using backpropagation learning algorithm. The first model is called the deconvolution neural network (DNN), and the second model is named the autoassociative deconvolution neural network (ADNN). The DNN utilizes the target echoes with and without noise, and generates an output value of zero or unity impulse whether the flaw is absent or present in the input data. The ADNN employs the autoassociative neural network for signal-to-noise ratio enhancement, and then applies the enhanced signal to deconvolution neural network for detection of target echoes. Testing results of these neural network models verify that target echoes which are interfered with each other can be detected and determined accurately in the presence of noise.

2.1 Introduction

This study presents design techniques for deconvolution neural networks utilizing the backpropagation learning algorithm in order to detect multiple target echoes in noisy environments [4, 17, 34, 35, 50, 52, 60, 74]. A study of this type can be useful for detecting ultrasonic flaw echoes in scattering noise, and used as an equalizer in high additive noise communication channels.

In the field of communication systems, the communication channel is modeled as an ideal low-pass filter. However, this model is physically unfeasible and difficult to approximate in practical systems. The physical nature of the transmission medium (or channel) causes the received symbol to be affected by the presence of the adjacent symbols. This overlap between adjacent symbols is called intersymbol interference (ISI) [38]. Since the deconvolution neural networks are nonlinear imaging processors, they can be used to recover the transmitted symbols. This scheme is referred to as an equalization scheme. Other neural network techniques that have been implemented to solve the problem of ISI are presented in [38] and [12]. In [38] authors employed the recurrent neural networks (RNNs) with a reasonable size to find the inverse of a communication channel. Other notable neural network type is radial basis function network that can recover the corrupted symbols in case of a known channel [12]. Results from testing of these neural networks show significant performance improvements.

Investigation that has been presented in this chapter is different than those that are given in [38] and [12]. Our deconvolution neural network technique is mainly concerned with multiple interfering echoes corrupted by uniform random noise. Methods presented in this chapter are capable of finding solutions for the problem of deconvolution without knowing anything about the physical parameters of the transmission channel, and further they have real-time processing, adaptability and training capabilities. In target detection, the measurement signal is determined in terms of input (i.e., wavelet), the impulse response function h(n) (target) and noise $v(n)$ as $y(n) = h(n) * x(n) + v(n)$ where * represents the convolution operation. Then the deconvolution becomes a process of finding a good estimate of h(n) from the knowledge of y(n) and x(n). To find the target impulse response function, h(n), the backpropagation learning algorithm is used in designing the deconvolution neural network (DNN). Neural networks are, in their most general sense, a collection of various layers of nodes which can be connected in a variety of configurations [15, 21, 49]. They have found applications in many areas including ultrasound [65, 78, 79] for detection and characterization. Each node consists of the weighted sum of the nodes in the preceding layer passed through the hyperbolic tangent function. A set of desired output values is then compared to the actual output of the neural network for every set of input values. The weights are then appropriately updated using the gradient of the output error with respect to the weight being updated (see Chapter 1).

An adaptive hidden neuron algorithm is included in the design of the deconvolution neural network (DNN). This adaptive hidden neuron algorithm is promising for determining the optimal number of hidden neurons. The next section provides design techniques concerned with the deconvolution neural network (DNN) in order to achieve a high probability of detection for a reasonably low signal-to-noise ratio. In addition, Hinton diagrams are used to describe how these DNNs operate [49, 52, 78, 79]. In Section 2.4, the autoassociative neural network is introduced to improve signal-to-noise ratio, followed by the deconvolution neural network (autoassociative deconvolution neural network, ADNN). In Section 2.5, the performance of DNN has been evaluated using simulated data. In Section 2.6, DNN was compared with the least squares filter which is called the Wiener filter [36, 55] due to its robustness to the noise.

2.2 Design of Deconvolution Neural Networks

The deconvolution neural network (DNN) is implemented using the backpropagation algorithm and by exposing the network to a set of target echoes. The block diagram of DNN is shown in Figure 2.1. In detection problems, the received signal, r(n), is given as (see Figure 2.2)

$$r(n) = \begin{cases} u(n) * \sum_i a_i \delta(n-n_i) + \nu(n) \Rightarrow \text{Target + Noise} \\ \nu(n) \Rightarrow \text{Noise} \end{cases} \tag{2.1}$$

where u(n) is the detection wavelet, v(n) is the noise, and a_i is the amplitude (or reflectivity which corresponds to the size of the target) of the target impulse, $\delta(n-n_i)$, detected at n_i (i.e., location of the target). Our objective in this study is to detect the location of the target at the output of the deconvolution neural network. To prevent the deconvolution neural network from making a false decision, data normalization is performed to eliminate the effect of signal offsets and measurement scales. Normalization is given as

$$x(n) = \frac{r(n) - \mu}{\sigma}, \ n = 1, \ldots, N$$

$$\mu = \frac{1}{N}\sum_{n=1}^{N} r(n),$$

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}(r(n) - \mu)^2} \tag{2.2}$$

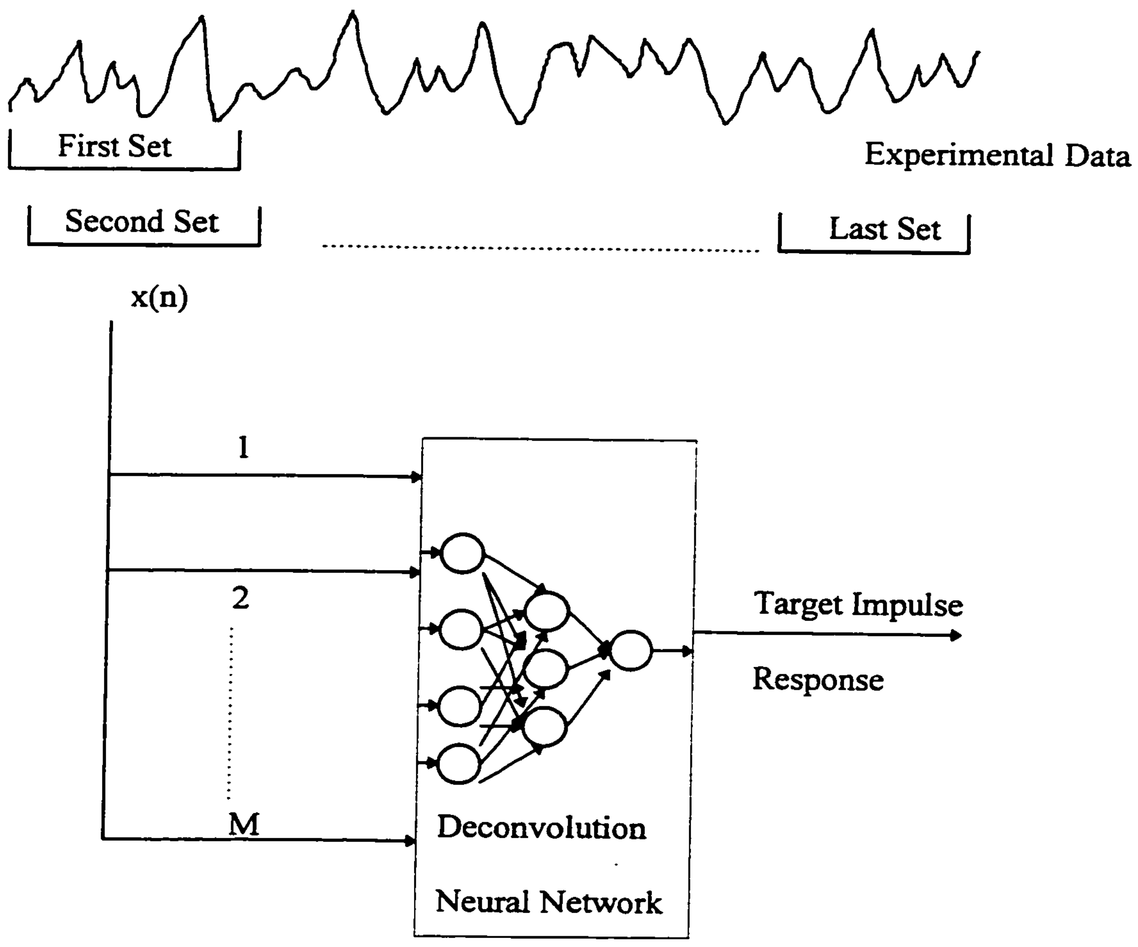


Figure 2.1 Block Diagram of the Deconvolution Neural Network (DNN)

where μ and σ are the estimated mean (i.e., signal offset) and the estimated standard deviation (i.e., measurement scale).

The input signal to the deconvolution neural network (see Figure 2.1) is created using a sliding window. The size of the sliding window is M, and the step between two successive windows is one sample. The first set is taken from the beginning of the data.

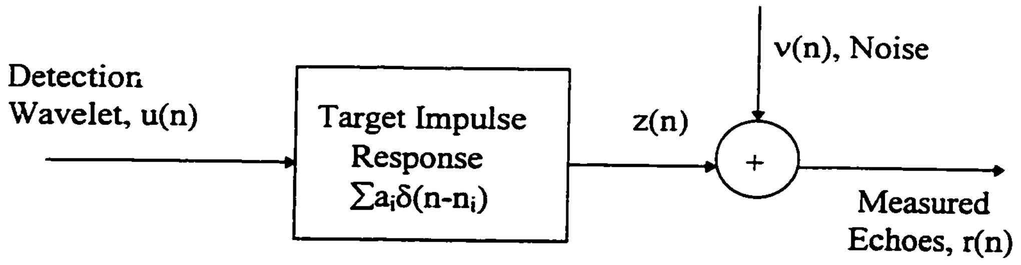


Figure 2.2 System Model of Target Echo Detection

The second set is one sample to the right of the first set, and this is repeated until the window covers the entire N samples of the measured signal. Then, the training matrix for the deconvolution neural network can be formed as

$$X = \begin{bmatrix} x(1) & x(2) & \cdots & x(N-M+1) \\ x(2) & x(3) & \cdots & x(N-M+2) \\ \cdots & \cdots & \cdots & \cdots \\ x(M) & x(M+1) & \cdots & x(N) \end{bmatrix} \quad (2.3)$$

where each column represents one set of normalized input data with an array length of M. Note that a total of N-M+1 set of input data are presented to the deconvolution neural network. Figure 2.3 shows a deconvolution neural network. The output vector of a deconvolution neural network (see Figure 2.3), $\bar{y}$, can be calculated as

$$\bar{y} = W^{o} \tanh(W^{h} X) \quad (2.4)$$

where X is defined in Equation (2.3), W^h and W^o are the weight matrixes for the hidden layer and the output layer respectively:

$$W^o = \left[w_{11}^{o} \cdots w_{1j}^{o} \cdots w_{1L}^{o}\right], \quad W^h = \begin{bmatrix} w_{11}^{h} & w_{12}^{h} & \cdots & w_{1i}^{h} & \cdots & w_{1M}^{h} \\ w_{21}^{h} & w_{22}^{h} & \cdots & w_{2i}^{h} & \cdots & w_{2M}^{h} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ w_{L1}^{h} & w_{L2}^{h} & \cdots & w_{Li}^{h} & \cdots & w_{LM}^{h} \end{bmatrix} \tag{2.5}$$

Backpropagation learning algorithm is used to estimate W^h and W^o for the optimal design of DNN. Then, the output vector, $\bar{y}$, is defined as

$$\bar{y} = [y(1),\ldots,y(i),\ldots y(N-M+1)] \tag{2.6}$$

where y(k) {k = 1, 2, ..., N-M+1} is equal to 1 if the input of the deconvolution neural network represents the target and noise class, otherwise y(k) is equal to 0 if the input of the deconvolution neural network represents the noise class only.

In deconvolution neural network design, a key issue is determining the number of hidden neurons. Improper selection of hidden neurons results in overfitting and underfitting problems. Overfitting (i.e., too many neurons) performs satisfactory for design data, but fails significantly for test data. Underfitting (i.e., too few hidden neurons) causes unsatisfactory convergence (i.e., DNN is not fully trainable). To avoid problems of overfitting and underfitting an adaptive hidden neuron algorithm for determining the number of hidden neuron is required (see Figure 2.4). The adaptive hidden neuron algorithm starts with three hidden neurons and adaptively increases the number of hidden neurons until convergence is guaranteed. This approach avoids both problems of overfitting and underfitting [9, 32, 85]. Note that the performance of the adaptive hidden neuron algorithm depends on the new data. New set of data should be independent of the training data. If the testing data is appeared to be excessively the same as the training data, the performance of the DNN is successful; on the other hand, it is disaster for the data that represents the rest of the population. Ability to accomplish the error goal (sum-squared-error, SSE) is limited by the extent of the new data. The SSE is computed by summing the squared differences between the actual output of the neural network and the desired output value. This error is used in updating the neural network weights using a backpropagation algorithm.

In reality, it might be difficult to collect that many new set of data to reach the performance goal (sum-squared-error). Using sliding window technique (see Figure 2.1) could create as many samples as possible for the training of deconvolution neural networks. Finally, the number of hidden neurons is increased by one when the SSE does not meet the appropriate criterion after every 5000 epochs (an epoch is defined as a single sweep through all sample vectors). With the addition of one neuron to the neural network, the weights for existing neurons are kept the same, and the weights to the new hidden neuron from the input neurons and output neuron are initialized to small random numbers. Then, the training process is repeated to estimate the weights of the neural

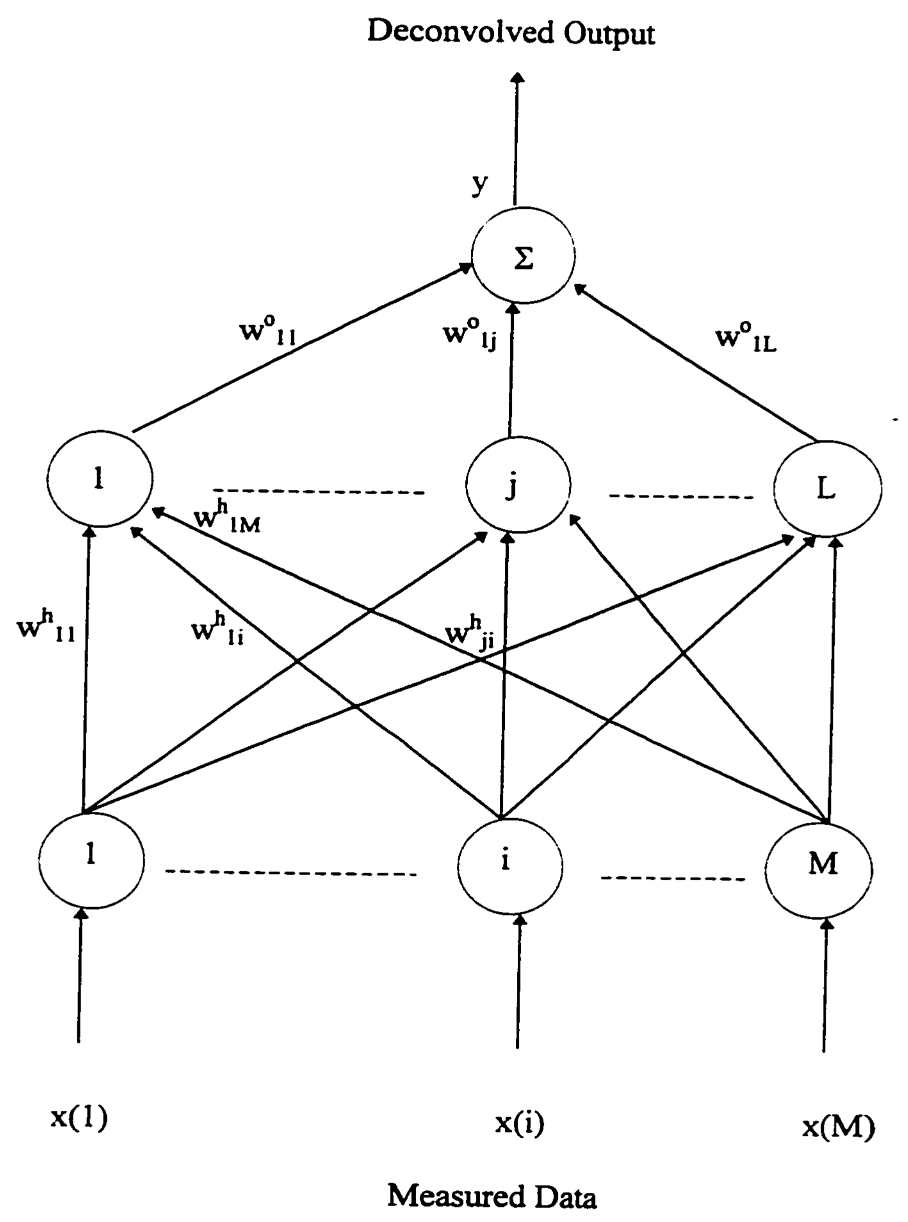


Figure 2.3 Canonic Form for the Deconvolution Neural Network

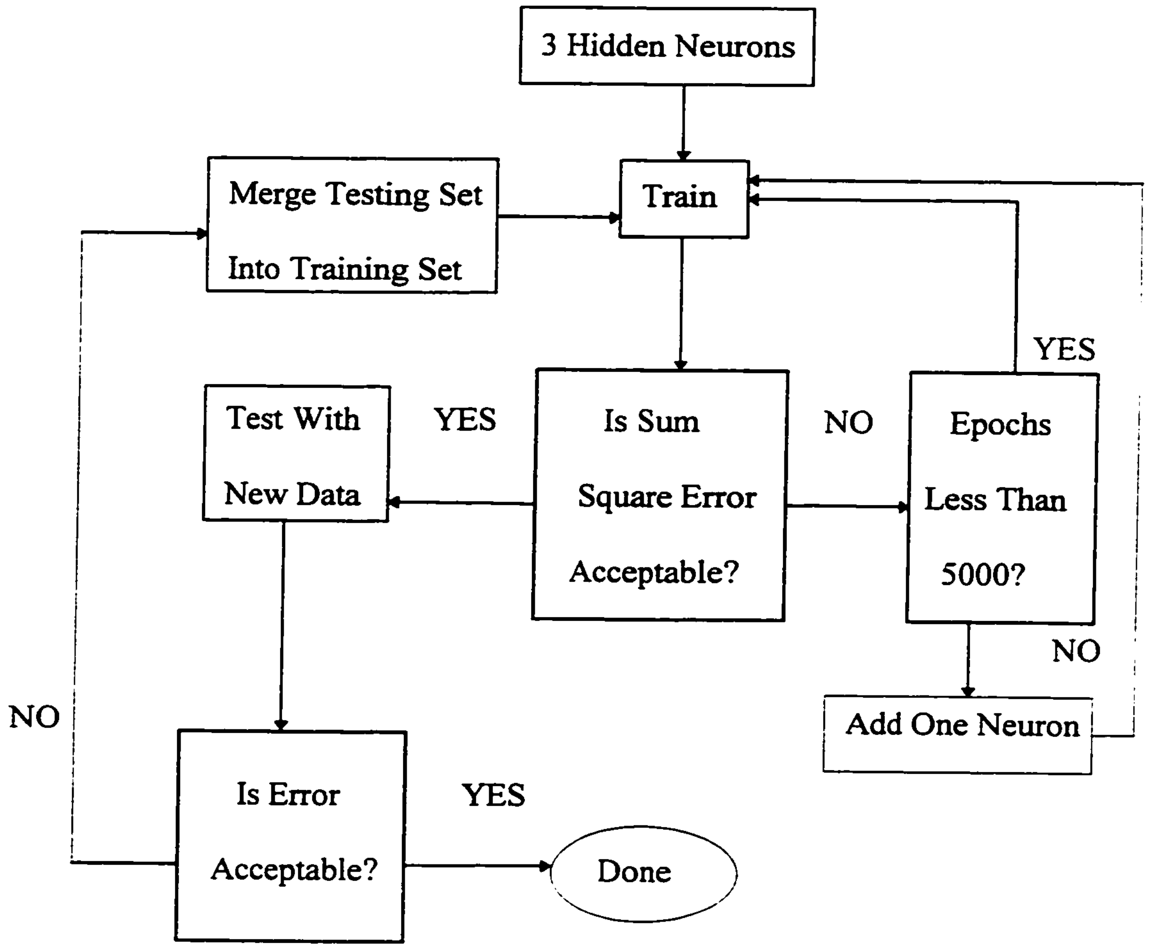


Figure 2.4 Adaptive Hidden Neuron Algorithm for the Optimal Design of DNN

network. If the SSE criterion is met, the trained neural network is tested with a new set of data which should be independent of the training set. If the testing SSE is met, then the training is complete, otherwise, the testing set is merged into the training set and the training is restarted. As a result, overfitting and underfitting are ruled out using adaptive hidden neuron algorithm.

2.3 Results and Discussion

The main objective for using DNN is to detect target impulses using ultrasonic measurements. The experimental data is obtained using 5 MHz broadband transducer with a 50 MHz sampling rate for data acquisition. The total number of training vectors are equal to 3615 with an array length of 39 points which is the same as the number of input neurons (M = 39). The training matrix (see Equation (2.3)) consists of samples of one, two and three echoes with and without noise. And the noise is created using a uniform random number generator that generates numbers between 0 and 1. The optimal number of hidden neurons is found to be 46 using the adaptive hidden neuron algorithm. Signal-to-noise ratio (SNR) is defined as

$$SNR = 20\log_{10}\left(\frac{Max(|z(n)|)}{Max(|v(n)|)}\right), \tag{2.7}$$

where z(n) is the multiple wavelets, and v(n) is the noise. The above definition of SNR is appropriate in this research since our objective is finding the whereabouts of the echoes by looking at the signal amplitudes. The SNR that was used in the training of deconvolution neural network was between 8-16 dB. Table 2.1 summarizes these training results for the DNN. Note that the total number of training vectors that was applied to DNN for training was 3615 and they were applied 215535 times to estimate the optimal number of hidden neurons and their weights in order to accomplish the SSE of 0.02.

Figure 2.5a presents an echo whose location is at 57 without noise. The output of the deconvolution neural network is presented in Figure 2.5b. Detected echo is at location 57. Figure 2.5c shows a testing result for one echo located at 57 with signal-to-noise ratio 12 dB. Echo is detected at location 57 in Figure 2.5d. Figure 2.6a depicts two interfering echoes located at 57 and 60 with signal-to-noise ratio (SNR) of 12 dB. Detected echoes are at locations 57 and 60 in Figure 2.6b. Figure 2.6c presents the testing result for three interfering echoes located at 57, 59 and 61 with signal-to-noise ratio 12 dB. Echoes are detected at locations 57, 59 and 61 in Figure 2.6d. These results demonstrate that deconvolution neural network (DNN) can be used as an echo detector when the backscattered signal has SNR of 12 dB. Note that for better testing results for lower SNRs, the training SNR should be lowered than 8 dB.

The success of deconvolution neural network may be obscured if it is not known how it works. In this chapter, Hinton Diagrams are presented to explain the optimal characteristics of the deconvolution neural network. The purpose of Hinton diagrams is to display the weights of the deconvolution neural network. In the Hinton diagrams the size of each square is proportional to the value of the weight. The plus, +, sign indicates a positive magnitude weight, and no sign indicates a negative magnitude weight. Hinton diagrams display the graphical distribution of weights suggesting that all hidden nodes contribute to the decision. In other words, optimal design of deconvolution neural network has been achieved. This implies that when an input signal is different from the desired echo (e.g., noise) the output is close to zero, and when the input matches the

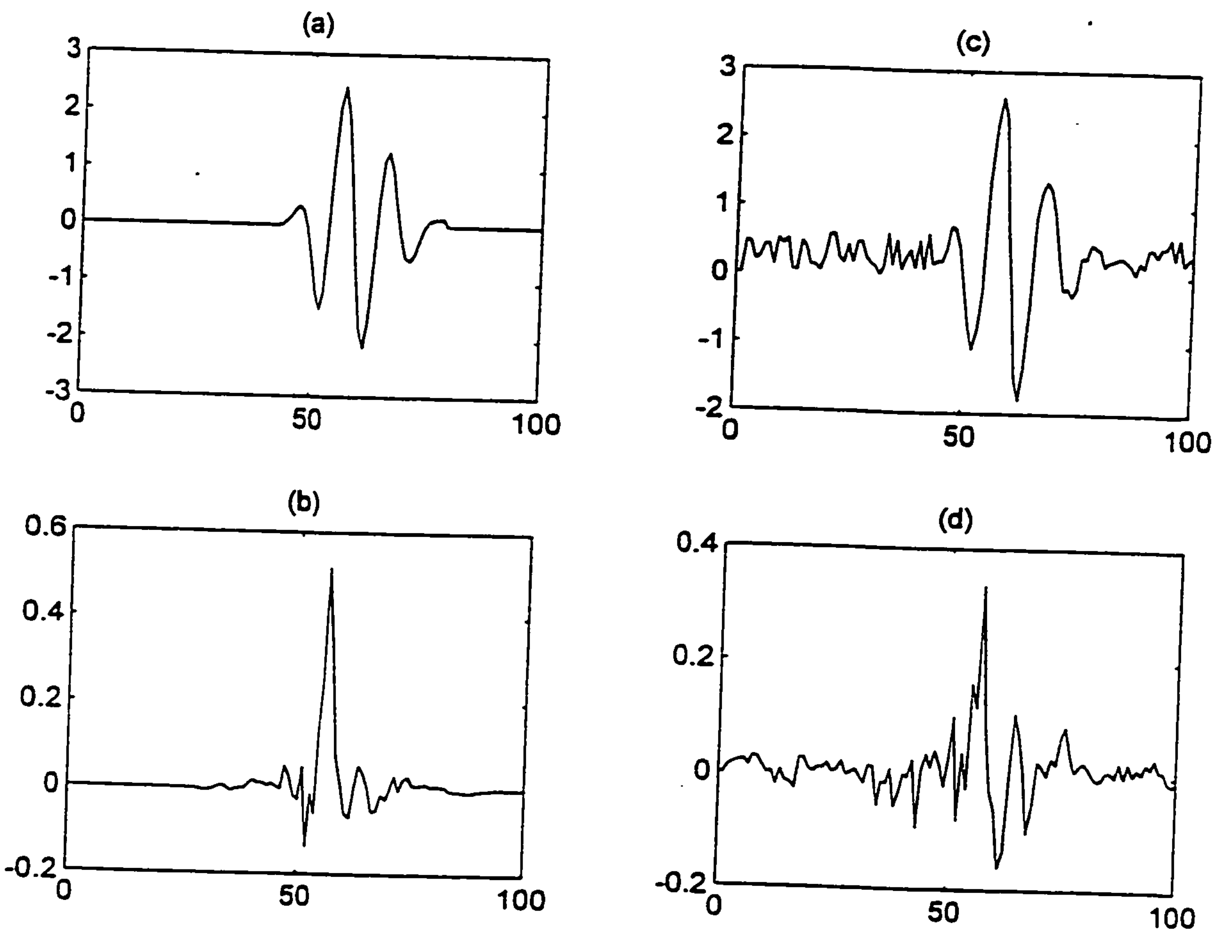


Figure 2.5 (a) One Echo is Located at Location 57 without Any Noise, (b) Output of DNN, and Detected Echo is at Location 57, (c) One Echo is Located at Location 57 with SNR = 12 dB, (d) Output of DNN, and Detected Echo is at Location 57

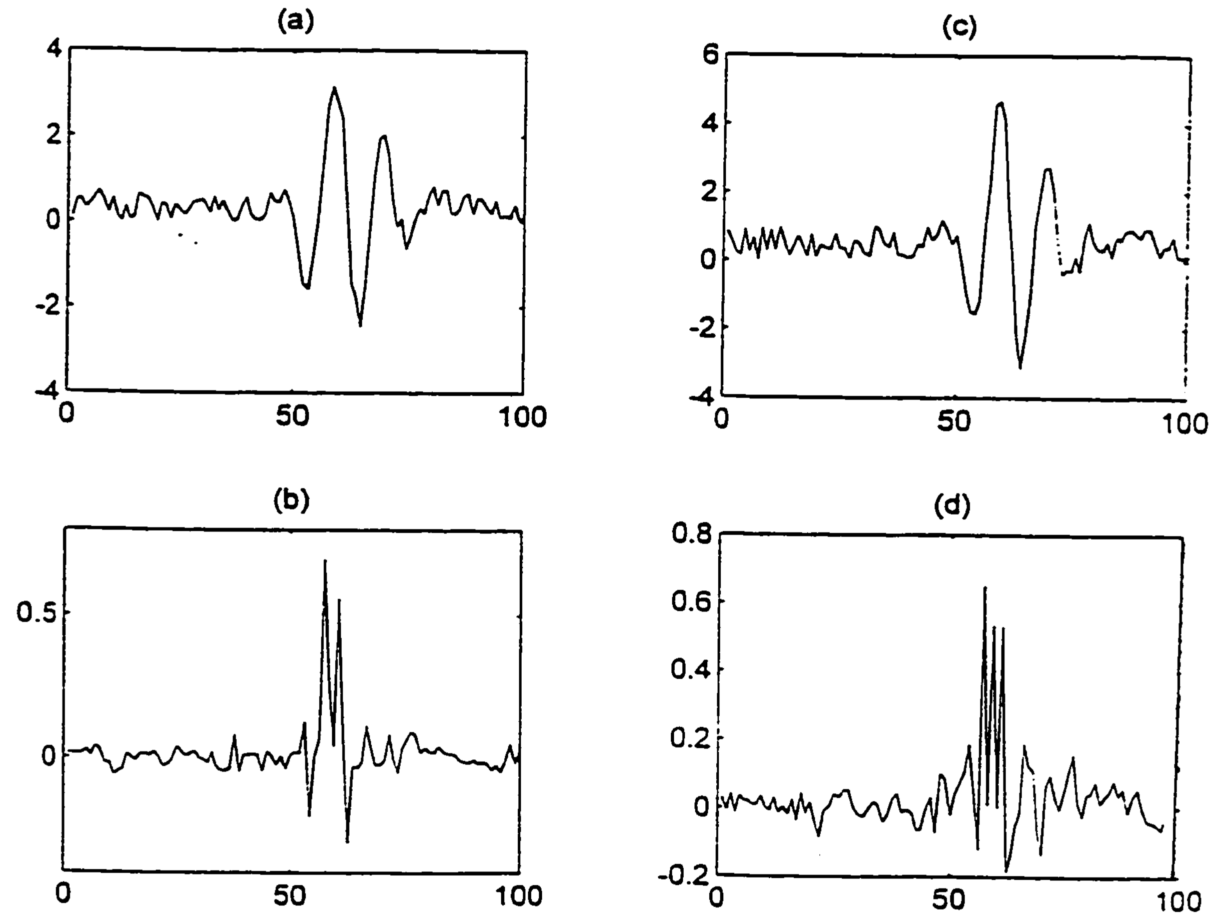


Figure 2.6 (a) Two Echoes are Located at Locations 57 and 60 with SNR = 12 dB, (b) Output of DNN, and Detected Echoes are at Locations 57 and 60, (c) Three Echoes are Located at Locations 57, 59, and 61 with SNR = 11 dB, (d) Output of DNN, and Detected Echoes are at Locations 57, 59, and 61

Table 2.1 Training Results for DNN Using Experimental Data

	DNN
Number of Sample Vectors	3615
Sum-Squared-Error	0.02
Number of Inputs	39
Number of Outputs	1
Number of Hidden Neurons	46
Epoch	215535
SNR	8-16 dB

desired echo, the output is an impulse function. Figure 2.7 depicts the Hinton diagram of the deconvolution neural network. The first row shows the weights from the first hidden neuron to the inputs, the second row depicts the weights from the second hidden neuron to the inputs and so on. Note that the number of hidden neurons is equal to 46 in DNN. The reason to have that many hidden neurons is that the adaptive hidden neuron algorithm determined the number of hidden neurons as 46 to reach the training SSE of 0.02. Furthermore, some of these weights are close to zero (see Figure 2.7). This does not necessarily mean that the input is unimportant. Several small weights for that input can sum up to a significant effect. A small weight affects a hidden neuron only slightly. But, if that hidden neuron has a big weight connecting to the output neuron, it has a big influence on the output signal. Results obtained from experimental data indicate that the

DNN needs that many hidden neurons even some of them are smaller compared to the rest of the weights in the hidden layer (see Figures 2.5 and 2.6).

2.4 Autoassociative Deconvolution Neural Network

This section introduces the autoassociative deconvolution neural network in order to improve signal-to-noise ratio (SNR) before applying the deconvolution neural network to the noisy signal. The block diagram of the autoassociative deconvolution neural network (ADNN) is depicted in Figure 2.8. The number of output neurons in the autoassociative neural network is equal to that of input neurons which is 39 for our research. The training matrix, X (see Equation (2.3)), has the samples of one, two and three echoes with and without noise. After training of this neural network, if a testing resembles one that is used in the training phase, the neural network generates an output which is close to a training output. This means that if a testing input is identical to one of the training patterns and is corrupted by noise, the network filters the noise and outputs the desired echo [28, 41, 44, 49, 51]. Table 2.2 summarizes training results for the autoassociative neural network. It should be noted that 3615 training data sets were applied 33985 times to estimate the optimal number of hidden neurons which was 9 and their weights in order to satisfy the SSE of 0.02. The training SNR for ANN was between 8-16 dB.

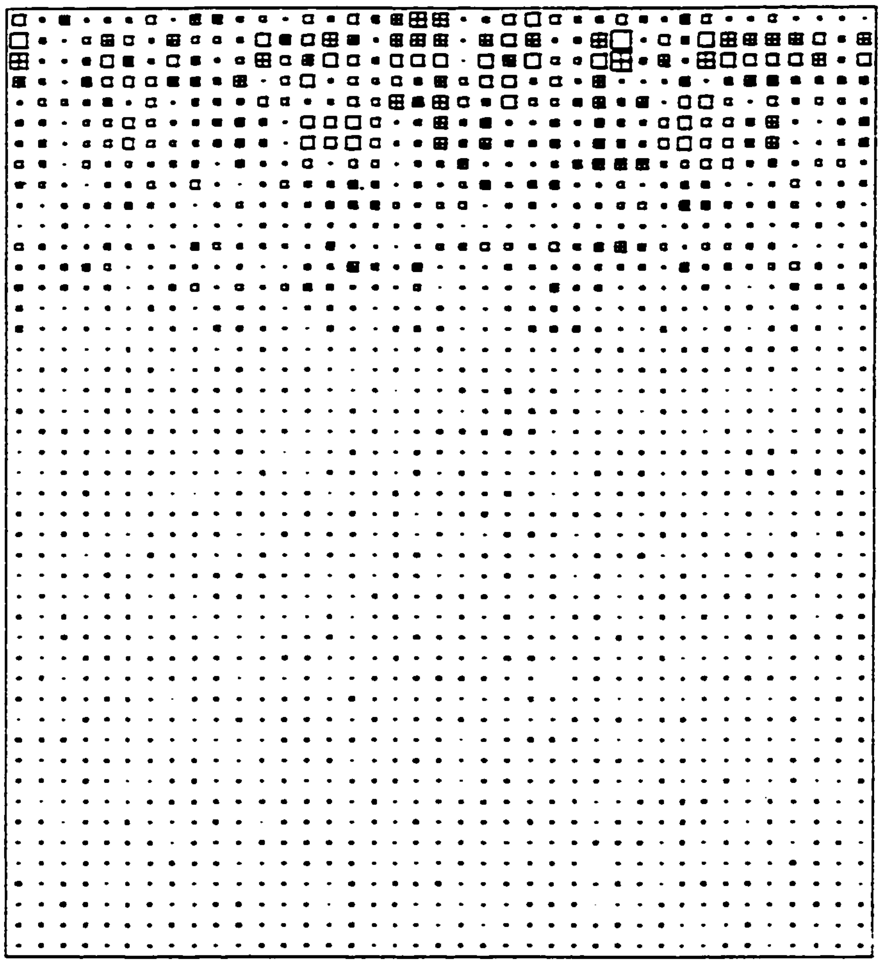


Figure 2.7 Hinton Diagram for the Hidden Layer

Table 2.2 Training Results for ANN Using Experimental Data

	ANN
Number of Sample Vectors	3615
Sum-squared-error	0.02
Number of Inputs	39
Number of Outputs	39
Number of Hidden Neurons	9
Epoch	33985
SNR	8-16 dB

We have evaluated the performance of ADNN using ultrasonic experimental data. Figures 2.9a and 2.9b display two interfering echoes located at locations 57 and 61 with signal-to-noise ratio 4 dB and the output of the deconvolution neural network respectively. Detected echoes are at locations 54 and 61. Figure 2.9c displays the output of ANN after the signal that is given in Figure 2.9a was applied to ANN. Figure 2.9d depicts the output of DNN after the output of ANN is applied to DNN. Echoes are correctly detected at locations 57 and 61. These experimental results demonstrate that SNR enhancement improves the performance of the deconvolution neural network for echo detection significantly.

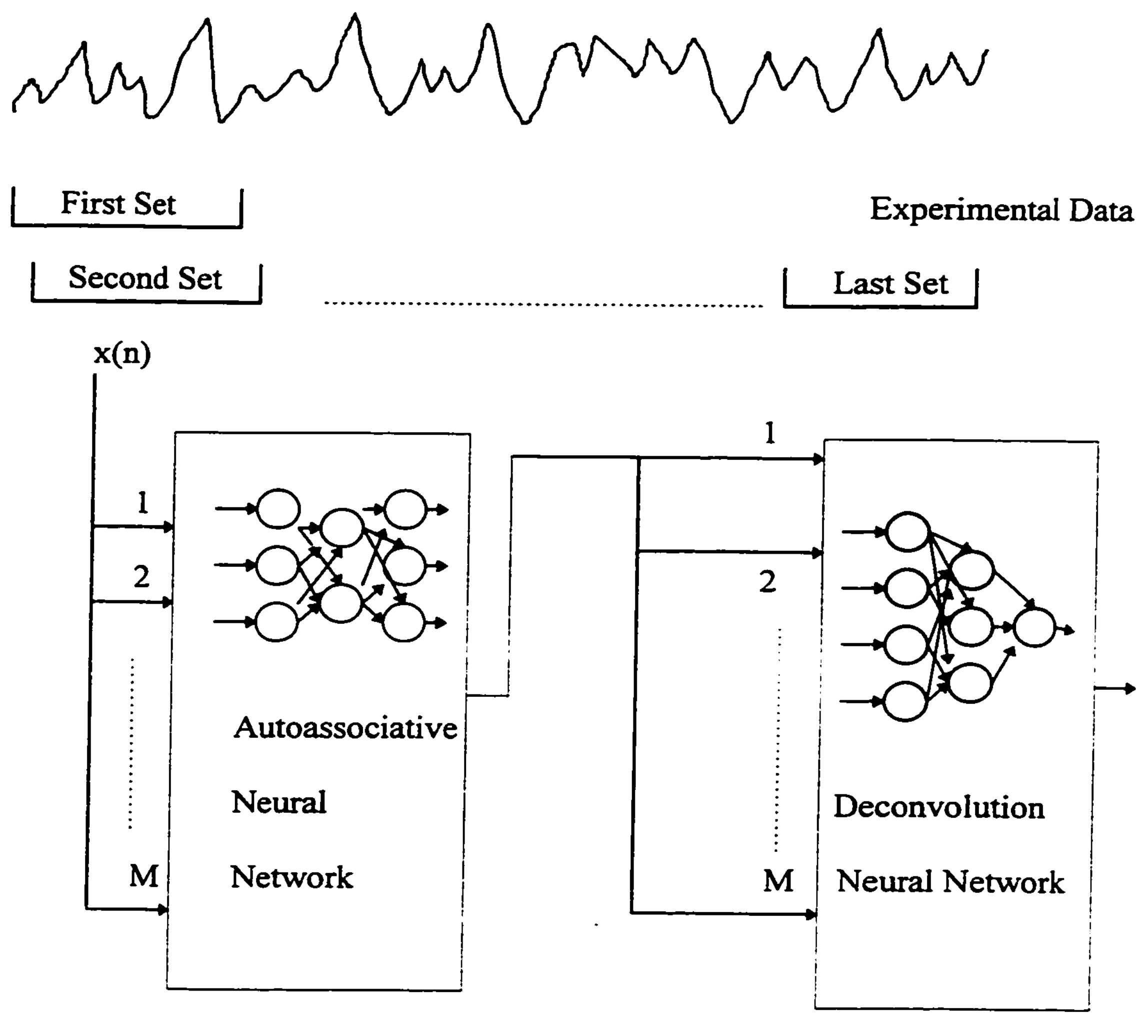


Figure 2.8 Autoassociative Deconvolution Neural Network

2.5 Performance Evaluation of DNN

The performance of a deconvolution neural network has been evaluated using simulated data set which is modeled as

$$u(n) = A\cos(2\pi f_c(n-n_o)+\phi)e^{-\alpha(n-n_0)^2} \tag{2.8}$$

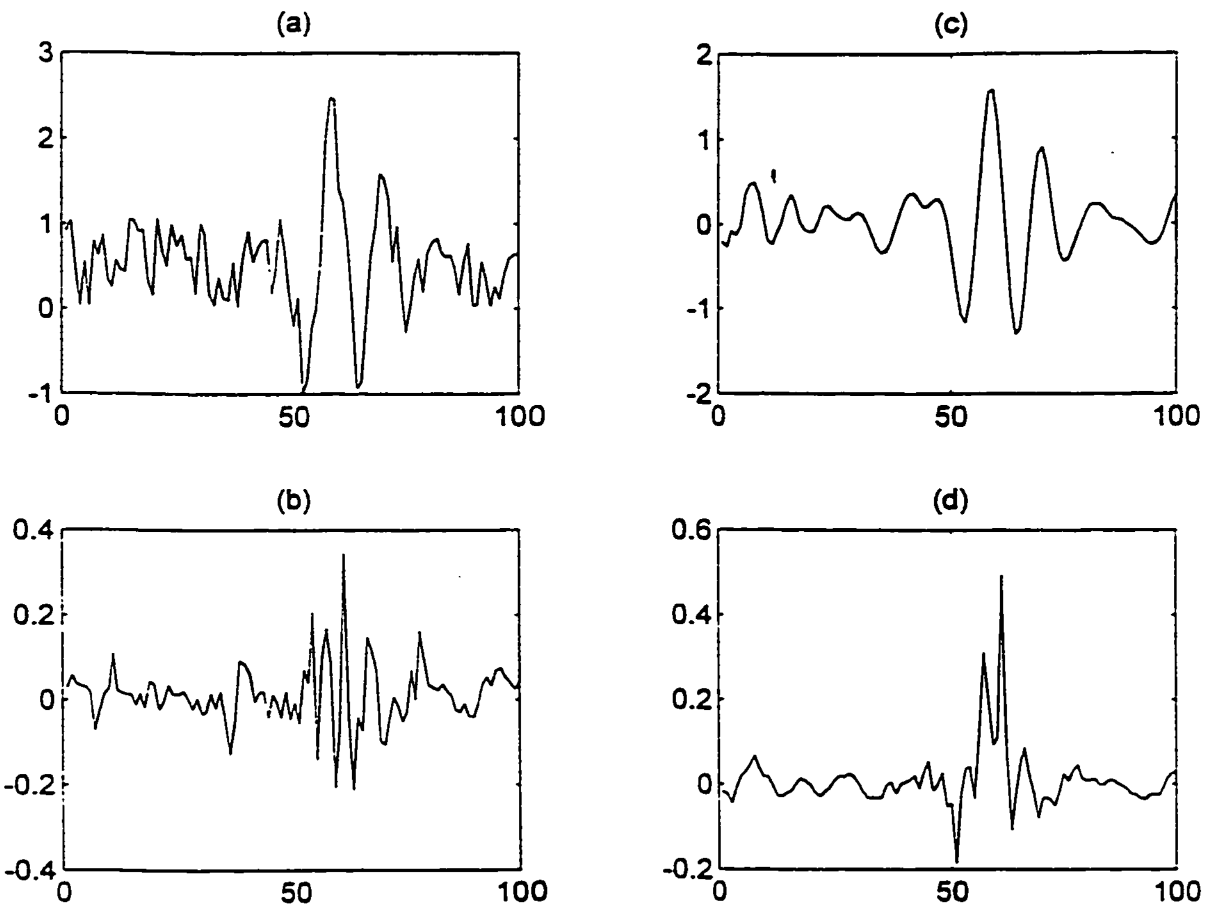


Figure 2.9 (a) Two Echoes are Located at Locations 57 and 61 with SNR = 4 dB, (b) Output of DNN, and Detected Echoes are at Locations 54 and 61, (c) Output of ANN, (d) Output of DNN is Shown after the Output of ANN is Applied to DNN (SNR improvement), and Detected Echoes are at Locations 57 and 61

where u(n) is the simulated detection wavelet (see Figure 2.2), A is the amplitude (= 0.47 units), f_c is the center frequency (= 4.8 MHz), n_o is the delay (= 0.36 sec), ϕ is the phase (= 82°), and α is the bandwidth factor (= 25.3). This model represents a back-scattered echo corresponding to the properties of the materials. The goal of this investigation is to measure the performance of the deconvolution neural network when the above parameters are changed. Received signal in this case is composed of two terms (see Figure 2.2), and then the Equation (2.1) is rewritten as

$$r(n) = z(n) + \nu(n),$$

$$z(n) = u(n) * \sum_i a_i \delta(n - n_i),$$

$$r(n) = A\cos(2\pi f_c(n - n_o) + \phi)e^{-\alpha(n-n_o)^2} * \sum_i a_i \delta(n - n_i) + \nu(n) \qquad (2.9)$$

where $\nu(n)$ is the noise.

Figure 2.10a shows a simulated testing echo located at location 57 (α = 25.3, f_c = 4.8 MHz, ϕ = 82° , n_o = 0.36 secs, A = 0.47 units). The result is an impulse at location 57 in Figure 2.10b. Figures 2.10c and 2.10e depict the testing results when the phase of the simulated echo is changed only (ϕ = 114° and 172° respectively). Echoes are detected at locations 56 and 55 in Figures 2.10d and 2.10f respectively. Equation (2.8) can be rewritten as

$$u(n) = A\cos(2\pi f_c(n - n_o))e^{-\alpha(n-n_o)^2}\cos\phi \; - \; A\sin(2\pi f_c(n - n_o))e^{-\alpha(n-n_o)^2}\sin\phi \qquad (2.10)$$

The effect of the second term in Equation (2.10) is observed in the output of deconvolution neural network as a second impulse which has a negative amplitude in Figures 2.10d and 2.10f. Figure 2.11 shows the performance of the deconvolution neural network when the center frequency of the simulated echo is changed only. In Figure 2.11a the simulated echo is located at location 57 and has a center frequency of 4 MHz. Echo is detected at location 59 in Figure 2.11b. Figure 2.11c has a simulated echo located at location 57 and has a center frequency of 3 MHz. Echo is detected at location 59 in Figure 2.11d. One can notice that the performance of the deconvolution neural network deteriorates as the frequency of the simulated echo changes. To overcome that problem, one should train the deconvolution neural network with the samples from these frequencies. Next section displays the results when Wiener filter deconvolution is applied to the ultrasound signals. Then, the results of Wiener filter were compared with the results of DNN.

2.6 Comparison of DNN with Wiener Filter Deconvolution

Experimental results reveal that the DNN is an effective technique for deconvolving closely located multiple ultrasound target echoes in noisy environments. In this section, DNN is compared with Wiener filter deconvolution which has been widely used in signal and image processing systems [36, 55]. This deconvolution filter gives the best linear mean square estimate of the target impulse response function from the observations. In this study, we evaluated the performance of Wiener filter deconvolution for detection of ultrasound echoes in presence of noise.

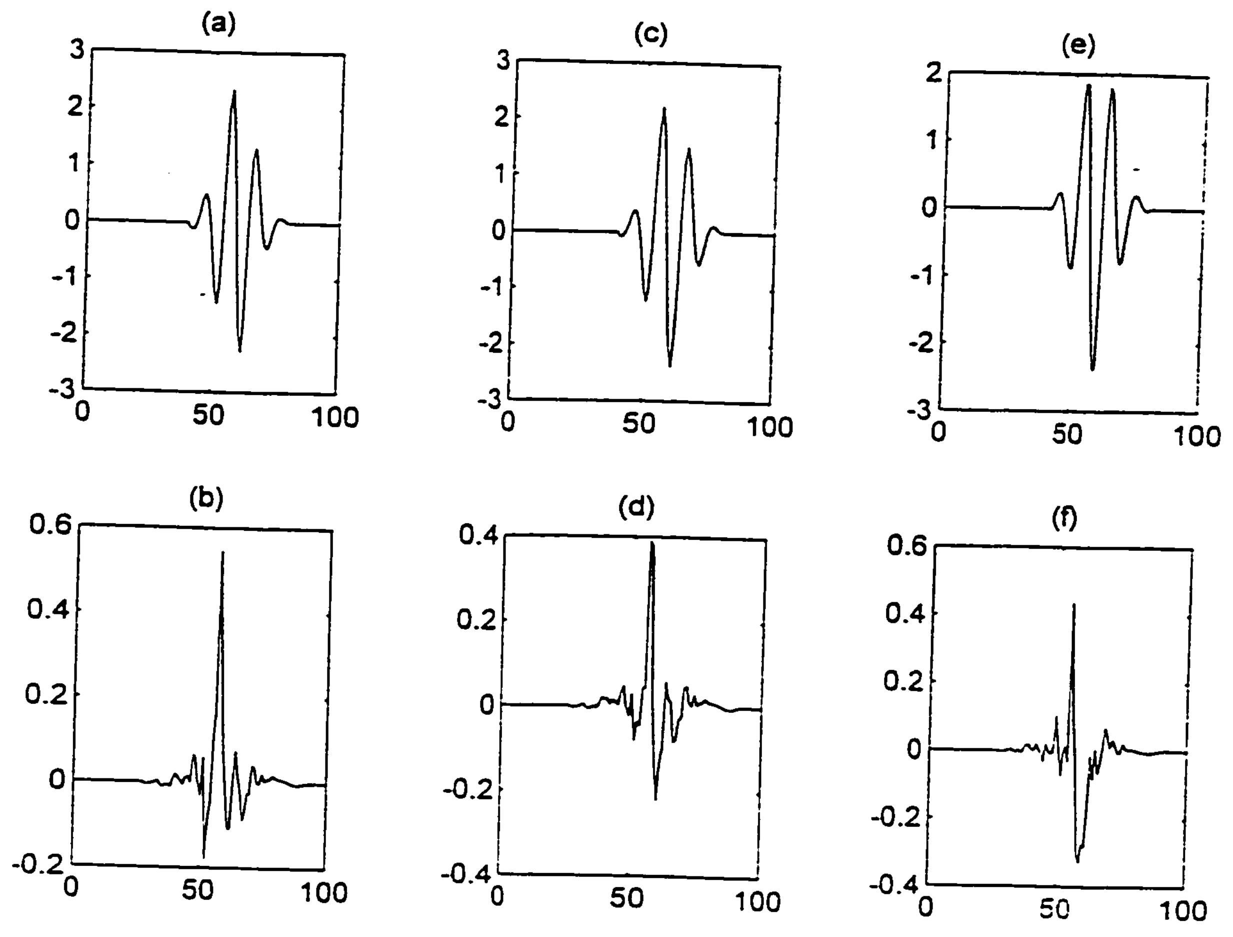


Figure 2.10 (a) Echo is Located at Location 57 with $\alpha = 25.3$, $f_c = 4.8$ MHz, $\phi = 82°$, $n_o = 0.36$ secs, A = 0.47 units, (b) Output of DNN, and Detected Echo is at Location 57, (c) Echo is Located at Location 57 with $\alpha = 25.3$, $f_c = 4.801$ MHz, $\phi = 114°$, $n_o = 0.36$ secs, A = 0.47 units, (d) Output of DNN, and Detected Echo is at Location 56, (e) Echo is Located at Location 57 with $\alpha = 25.3$, $f_c = 4.801$ MHz, $\phi = 172°$, $n_o = 0.36$ secs, A = 0.47 units, (f) Output of DNN, and Detected Echo is at Location 55

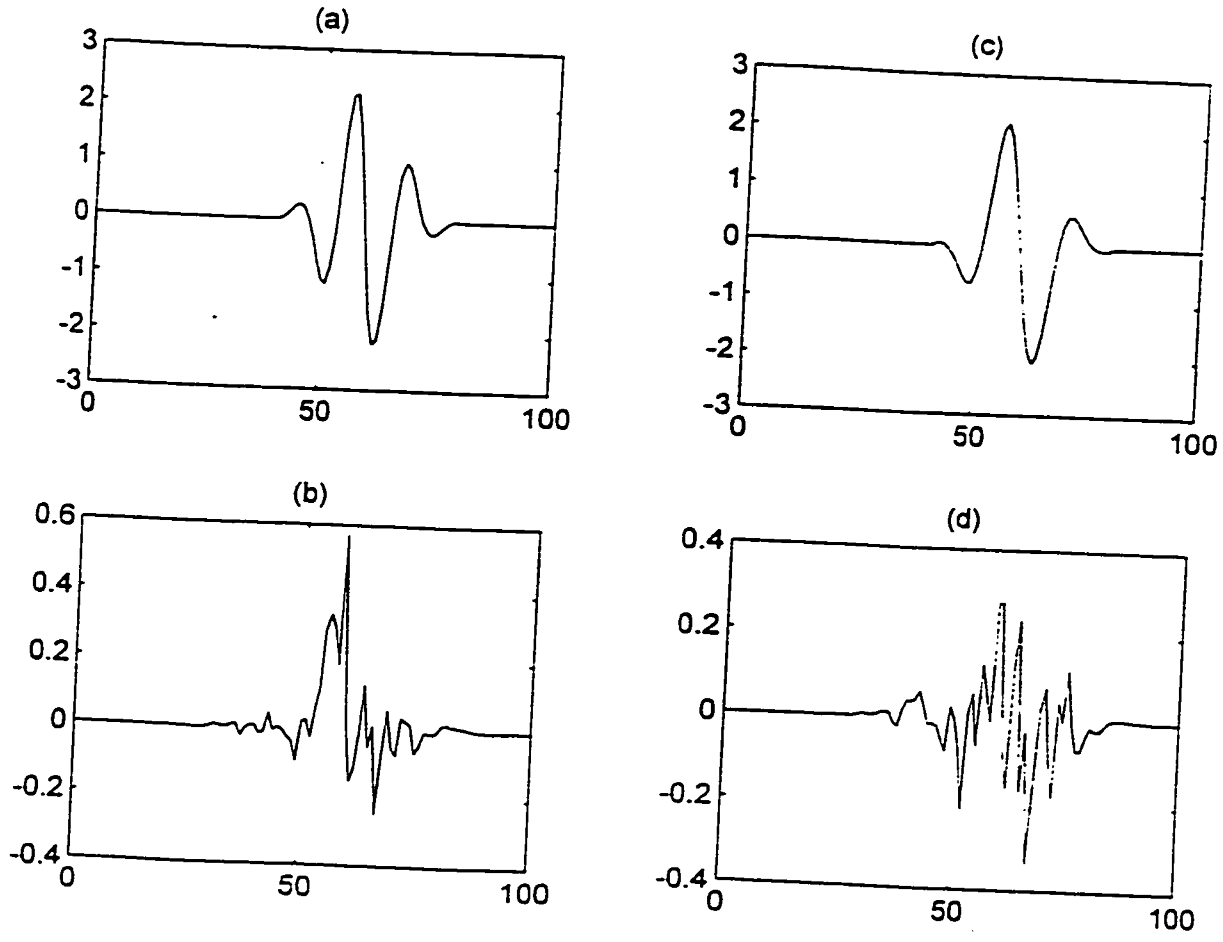


Figure 2.11 (a) Echo is Located at Location 57 with $\alpha = 25.3$, $f_c = 4$ MHz, $\phi = 82°$, $n_o =$ 0.36 secs, A = 0.47 units, (b) Output of DNN, and Detected Echo is at Location 59, (c) Echo is Located at Location 57 with $\alpha = 25.3$, $f_c = 3$ MHz, $\phi = 82°$, $n_o =$ 0.36 secs, A = 0.47 units, (d) Output of DNN, and Detected Echo is at Location 59

Deconvolution based on the Wiener filter requires knowledge of the characteristics of the noise process in terms of its spectral density. The measured echoes, r(n), can be explained in terms of the discrete process z(n), and the noise v(n) as: r(n) = z(n) + v(n), where z(n) = u(n) * y(n), u(n) is the detection wavelet and y(n) is the target impulse response function (see Figure 2.12).

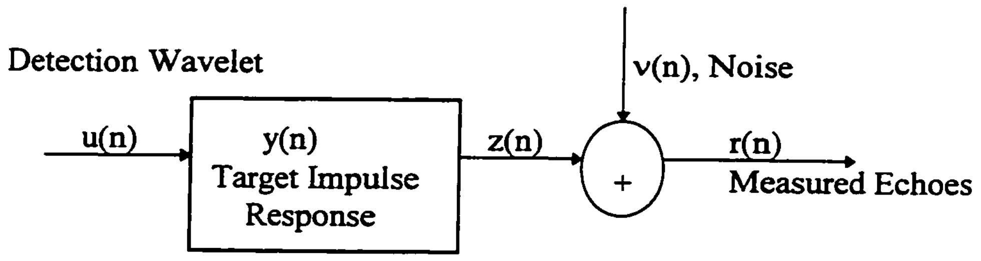


Figure 2.12 Block Diagram for Target Echo Detection

The estimation of z(n), $\hat{z}(n)$, is subject to error as follows,

$$\varepsilon = \sum_{n=-\infty}^{\infty} (z(n) - \hat{z}(n))^2 \tag{2.11}$$

Equation (2.11) can be minimized by applying an optimal linear operation, h(n), (i.e. Wiener filter) on the r(n). This means that the output of the h(n) is the estimator of z(n)

$$\hat{z}(n) = r(n) * h(n) \tag{2.12}$$

We assume that the noise, v(n), and the desired discrete process, z(n), are uncorrelated (i.e., they are also orthogonal since their mean is zero). Hence, the optimum H(z) (i.e., H(z) is the Z transform of h(n)) can be obtained by solving [55] the following equation

$$H(z)=\frac{S_{zz}(z)}{S_{zz}(z)+S_{vv}(z)} \tag{2.13}$$

where $S_{zz}(z)$ and $S_{vv}(z)$ are the power spectral density functions of the discrete process, z(n) and the noise, v(n), respectively. This filter is called the Wiener filter. Then the optimal target impulse response, $\hat{y}(n)$, in terms of the Wiener filter is given as

$$\hat{Y}(z)=\frac{R(z)}{U(z)}\left(\frac{S_{zz}(z)}{S_{zz}(z)+S_{vv}(z)}\right) \tag{2.14}$$

The Wiener filter has a desirable behavior. When there is no noise in the environment, H(z) becomes one and the target impulse response is the ratio of $\frac{R(z)}{U(z)}$. On the other hand, poor signal-to-noise ratio makes the H(z) approach zero and results in reducing the value of the optimal estimate of the target impulse response.

Figure 2.13a shows one target echo that is applied to the Wiener filter and Figure 2.13b displays the optimal target impulse response. One target echo is deconvolved at location 57 in case of no noise. In case of noise (SNR = 12 dB), single target echo (see Figure 2.13c) is applied to the Wiener filter and the detected echo is at location 57 which is shown in Figure 2.13d. Two target echoes which are located at locations 57 and 60 (SNR = 12 dB) are applied to the Wiener filter (see Figure 2.14a). And the result is depicted in Figure 2.14b and two echoes are detected at locations 58 and 61. Figure 2.14c shows three target echoes which are located at locations 57, 59, and 61 (SNR = 11 dB). The result is depicted in Figure 2.14d. One echo is clearly detected at location 58, however it is hard to observe the other two echoes in Figure 2.14d. Figures 2.5 and 2.6 depict the results of DNN for the same echoes. By looking at Figure 2.14, Wiener filter is failed at deconvolving closely located multiple target echoes in presence of noise. On the other hand, DNN is quite successful in detecting the same multiple target echoes. One can conclude that DNN technique is superior to Wiener filter deconvolution by comparing Figures 2.6 and 2.14.

2.7 Conclusion

Two novel design procedures have been developed in this chapter. DNN is the first method that deconvolves multiple interfering target echoes in noisy environments. ADNN is the second method that enhances the signal-to-noise ratio, and then applies the DNN. Results obtained from testing the DNNs and ADNNs are encouraging and potentially useful for nondestructive testing and quality control. These neural network models have the capability of solving deconvolution problems successfully and do not need any information about the type of transmission channel. After the training phase of

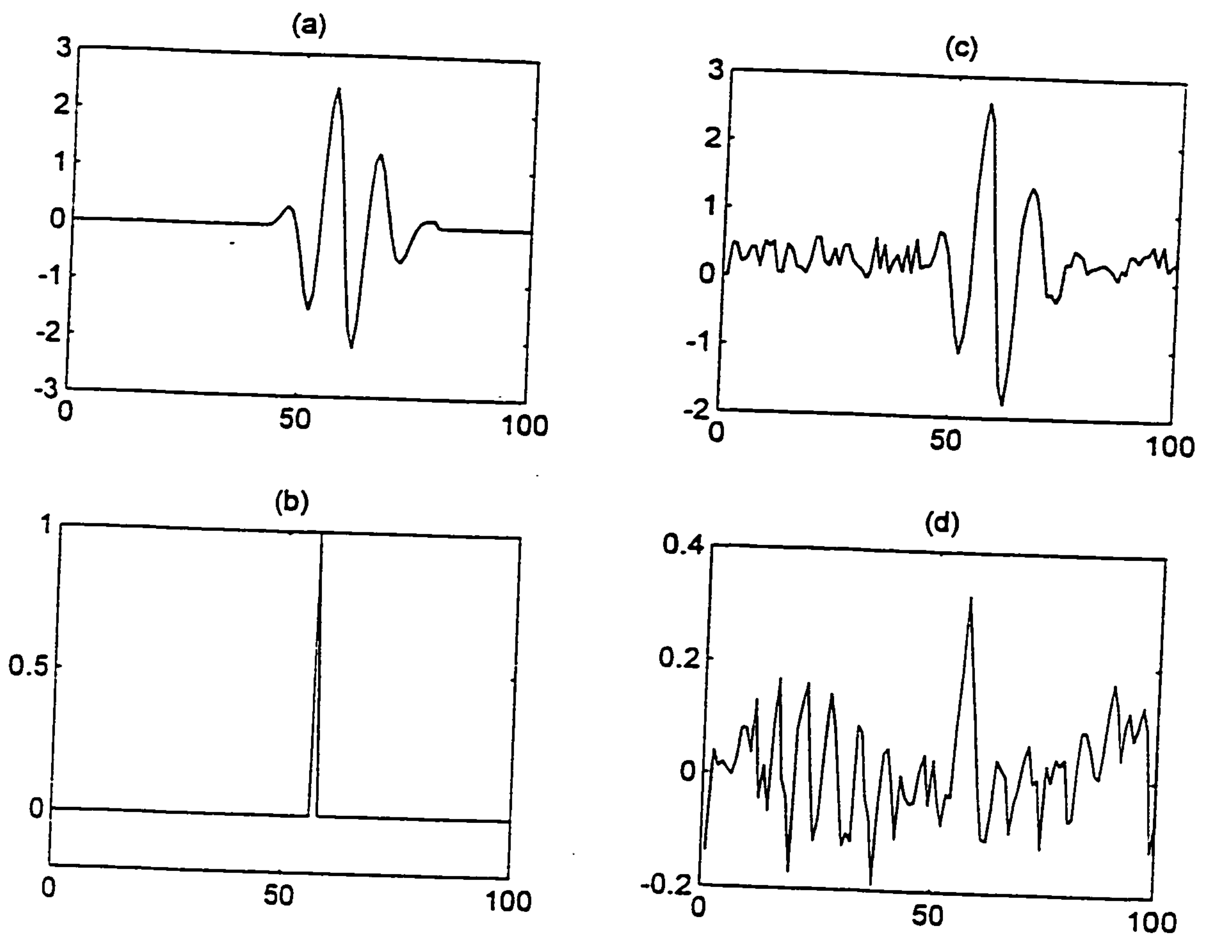


Figure 2.13 (a) One Echo is Located at Location 57 without Any Noise, (b) Output of Wiener filter, and Echo is Detected at Location 57 (c) One Echo is Located at Location 57 with SNR = 12 dB, (d) Output of Wiener filter, and Echo is Detected at Location 57

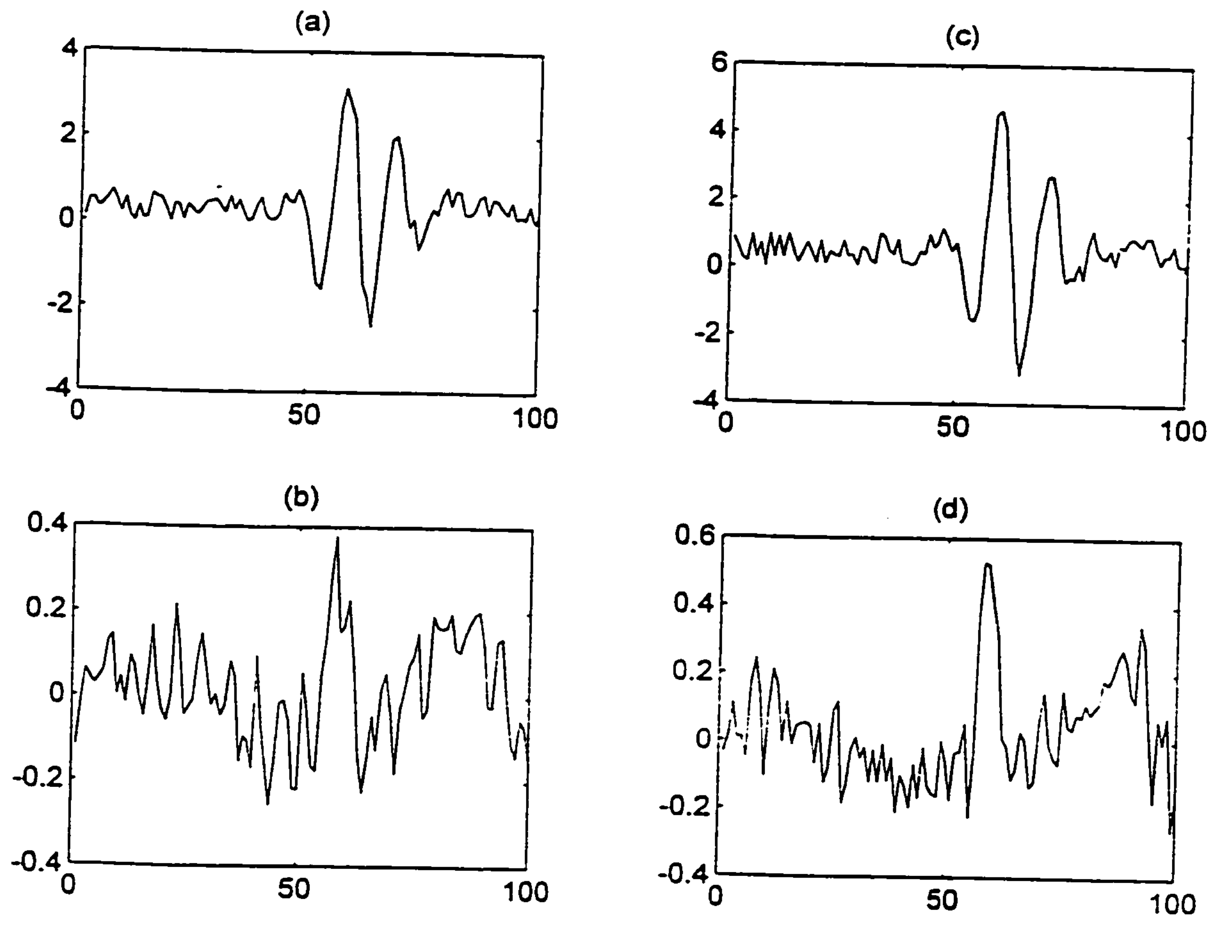


Figure 2.14 (a) Two Echoes are Located at Locations 57 and 60 (SNR = 12 dB), (b) Output of Wiener filter, and Echoes are Detected at Locations 58 and 61, (c) Three Echoes are Located at Locations 57, 59 and 61 with SNR = 11 dB, (d) Output of Wiener filter and Only One Echo is Detected at Location 58

these neural networks, the testing is a real-time processing which is desirable for ultrasonic imaging systems.

The key issue in the design of the neural networks is to use the adaptive hidden neuron algorithm. This algorithm is promising for determining the optimum number of hidden neurons. Problems of overfitting and underfitting can be avoided by including the adaptive hidden neuron algorithm in the design procedures. Experimental results show that the adaptive hidden neuron algorithm can be used efficiently with the backpropagation learning algorithm.

We employ Hinton Diagrams to explain how these neural networks work. Hinton diagrams display the graphical distribution of weights suggesting that all hidden nodes contribute to the decision. Experimental results and Hinton diagrams show that the optimal designs of DNN and ADNN have been achieved.

Testing results which were created using simulated echoes depict that changes in phase and frequency lessen the performance of the DNNs. To overcome that problem, DNNs should be trained with the samples of these echoes. Sometimes, finding that many samples could be a problem. Using sliding window technique could create as many samples as possible for the training process.

DNN has been compared with the Wiener filter deconvolution. Wiener filter has been widely used for detection in presence of noise. Experimental results reveal that the DNN does a superior job in deconvolving the multiple target echoes in grain scattering noise than that of Wiener filter deconvolution. Finally, we can say that DNNs and

ADNNs can be utilized effectively in the detection process of target echoes which are interfered with each other in the presence of measurement noise.

CHAPTER III

PROBABILISTIC DECONVOLUTION NEURAL NETWORK

In this chapter, probabilistic deconvolution neural network (PDNN) has been developed as a third method to solve the deconvolution problem. The advantage of utilizing PDNN is that the complexity and the total number of weights are less compared to those of DNN (see Chapter 2), while the performance of PDNN is the same as that of DNN. The PDNN consists of two processing stages. The first stage estimates Gram-Charlier coefficients [40] corresponding to target echoes and scattering noise. Then, in the second processing block, these parameters are applied to resolve multiple echoes. Results obtained in testing the PDNN show that multiple interfering echoes can be detected accurately in the presence of noise.

3.1 Introduction

The performance of PDNN is limited by the presence of microstructure noise [21, 61, 63, 64, 66, 82, 83]. Although the input signal (see Figure 3.1) to PDNN contains information related to flaw structure, this information is often masked by unwanted echoes caused by microstructure scattering. Let's represent this input signal as an m-component multivariate random vector $\vec{X} = [x(1), x(2), \ldots, x(m)]$ (m is 39 in our research which represents the length of the target echo). Then, there are two classes from which our input signal is drawn. These classes can be defined as

$H_0 = noise,$

$\text{H}_1 = \text{target echoes} + \text{noise.}$

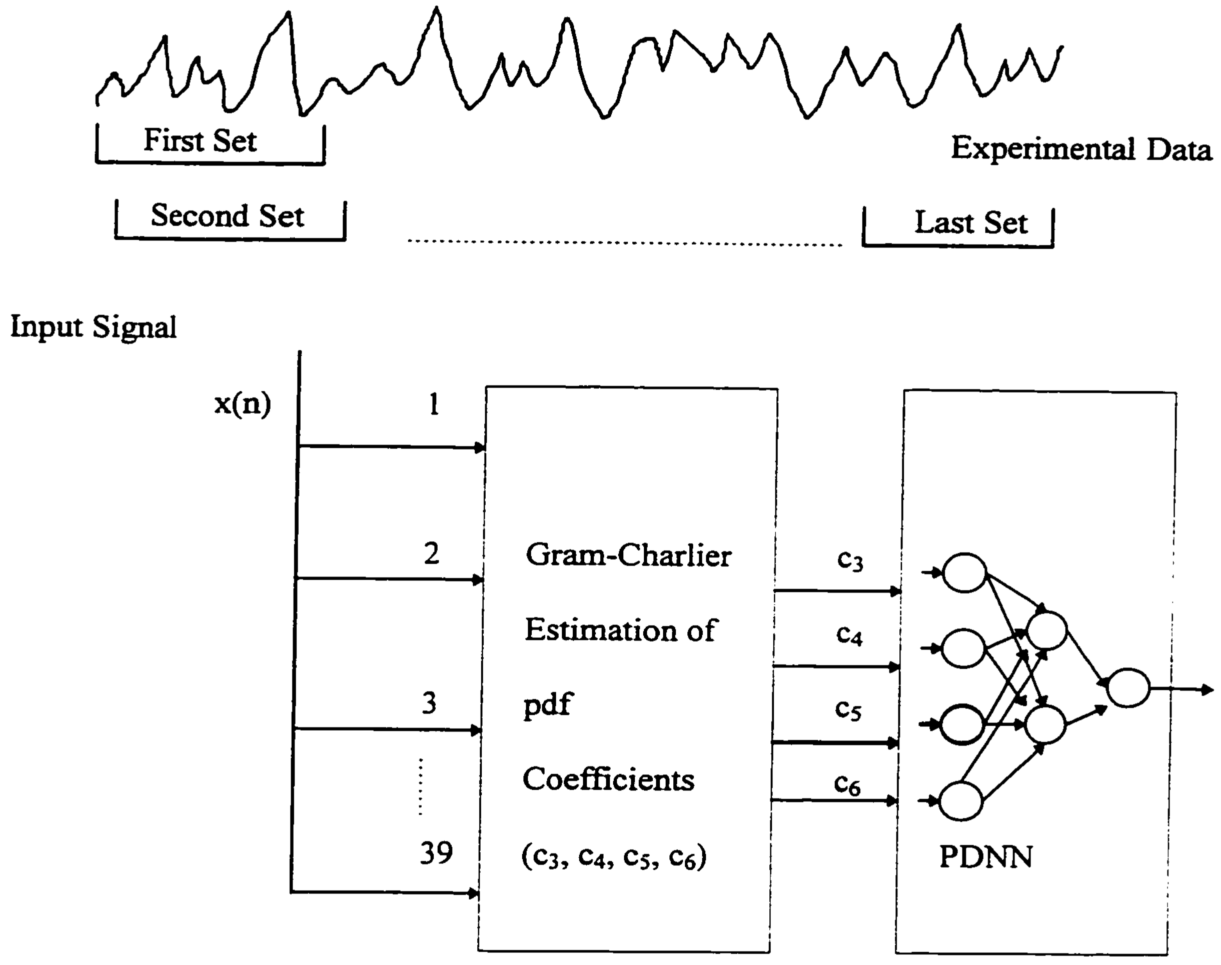


Figure 3.1 Probabilistic Deconvolution Neural Network

The prior probability of an unknown sample being drawn from class k is h_k (k = 0, 1). The cost making a wrong decision for the class k is v_k. Note that in many ultrasound signal processing problems, the prior probabilities, h_k, and the cost probabilities, v_k, are taken as being equal, and hence can be ignored. The training matrix consists of n_l

samples known to be from class H_0, and n_2 samples known to be from class H_1. The problem is to find a neural network structure for determining the class from which an unknown signal is taken. If we know the probability density functions $f_k(\vec{X})$ for all classes, the Bayes optimal decision rule [63, 64, 67, 82, 83] is to classify $\vec{X}$ into class l if

$$h_l v_l f_l(\vec{X}) > h_j v_j f_j(\vec{X}) \tag{3.1}$$

where $j \neq l$.

A major problem with Equation (3.1) is that the probability density functions of the classes are unknown. We use Gram-Charlier series expansion to estimate these unknown probability density functions. In this case the training matrix has the Gram-Charlier coefficients (see Figure 3.1). Our objective in this chapter is using these coefficients for classification. If the neural network has been previously configured and trained on known Gram-Charlier coefficients, then to determine the class of the unknown signal, we need only to feed its Gram-Charlier coefficients. Since Gram-Charlier coefficients are used as an input to PDNN, PDNN is also named the parametric neural network [6, 14, 47, 57, 69]. The advantage of using a parametric neural network is that the number of weights in PDNN are less and the structure of PDNN is simplified compared to DNN (see Chapter 2). Next section introduces the Gram-Charlier series expansion using basis functions in order to approximate the probability density functions of the unknown classes, H_k (k = 0, 1).

3.2 Gram-Charlier Series

The Gram-Charlier series expansion of the probability density function of a random variable with mean μ and variance σ^2 can be represented as

$$\rho(x) = \frac{1}{\sigma}\sum_{i=0}^{\infty} c_i \Phi^{(i)}\left(\frac{x-\mu}{\sigma}\right), \tag{3.2}$$

where $\Phi(x)$ is a Gaussian probability density function and $\Phi^{(i)}(x)$ represents the i-th derivative of $\Phi(x)$. For normalized data where ($\mu = 0$, $\sigma^2 = 1$, $c_0 = 1$), the above equation can be simplified to

$$\rho(x) = \left(\Phi(x) + c_3\Phi^{(3)}(x) + c_4\Phi^{(4)}(x) + c_5\Phi^{(5)}(x) + c_6\Phi^{(6)}(x) + \cdots\right), \tag{3.3}$$

where c_i coefficients are related to the central moments of $\rho(x)$. In a sense, derivatives of the Gaussian function in Equation (3.3) provide us the different frequency information of the input signal (see Figure 3.1). Some derivatives of the Gaussian function are presented in Figure 3.2. Furthermore $c_i\Phi^{(i)}$'s are orthogonal functions presenting unique information about the target and noise distribution. This leads us to conclude that PDNN

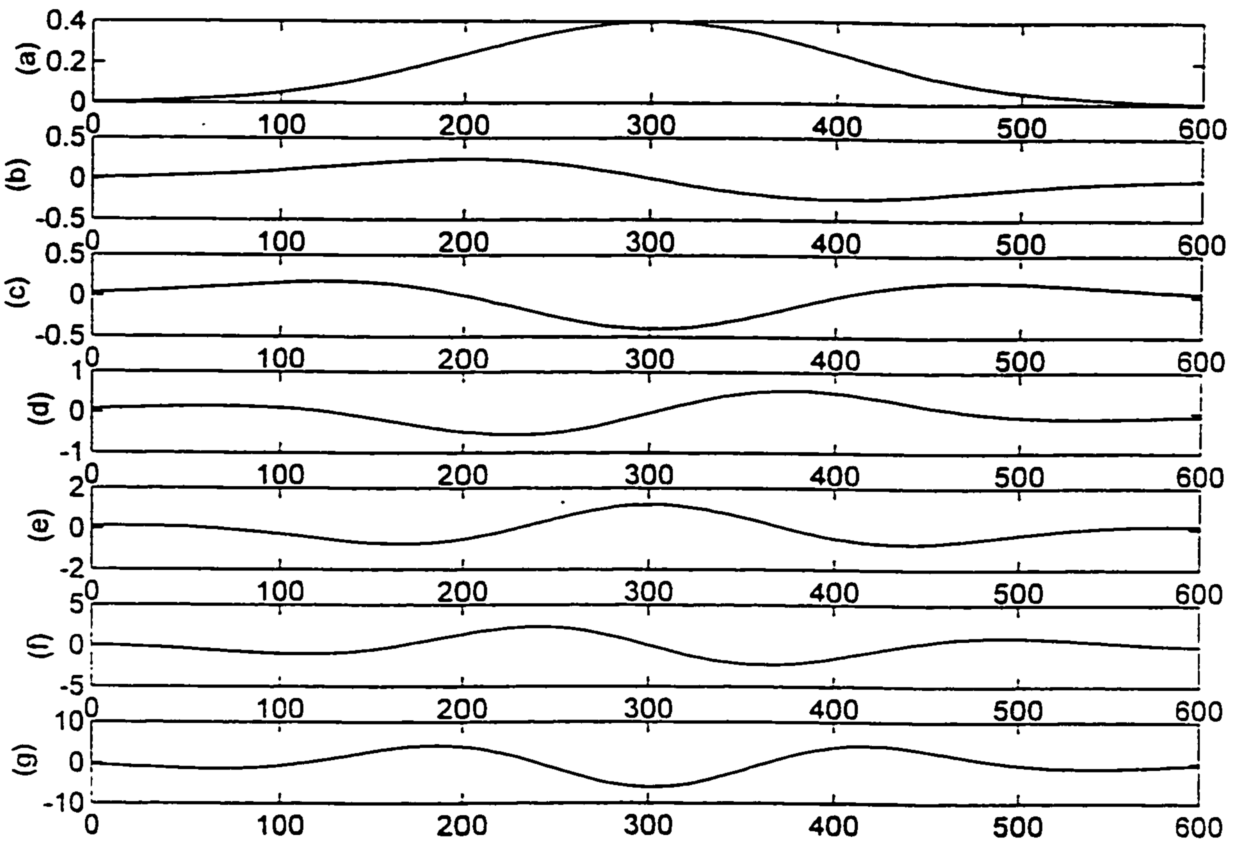


Figure 3.2 (a) Gaussian Function, (b) First Derivative of a Gaussian Function, (c) Second Derivative of a Gaussian Function, (d) Third Derivative of a Gaussian Function, (e) Fourth Derivative of a Gaussian Function, (f) Fifth Derivative of a Gaussian Function, (g) Sixth Derivative of a Gaussian Function

based on decomposition of pdf by Gram-Charlier series is well suited for ultrasonic flaw detection. To derive the c_i's, let's define the β(x) as

$$\beta(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}, \tag{3.4}$$

and consider the successive derivatives of β(x) with respect to x. We have

$$\frac{d\beta(x)}{dx} = -x\beta(x),$$

$$\frac{d^2\beta(x)}{dx^2} = (x^2 - 1)\beta(x),$$

$$\frac{d^3\beta(x)}{dx^3} = (3x - x^3)\beta(x), \tag{3.5}$$

and so on. The result is a polynomial in x multiplied by β(x). Then we introduce the polynomial $A_r(x)$ which is called Tchebycheff-Hermite polynomial [40]. This polynomial is defined as

$$(-1)^r \frac{d^r\beta(x)}{dx^r} = A_r(x)\beta(x). \tag{3.6}$$

From Equation (3.6), $A_0(x) = 1$. Let's change the argument in Equation (3.4), we have a new equation as

$$\beta(x-t) = \frac{1}{\sqrt{2\pi}} e^{(\frac{-1}{2}x^2 + tx - \frac{1}{2}t^2)} = \beta(x) e^{(tx - \frac{1}{2}t^2)}, \tag{3.7}$$

and by Taylor's theorem we could write the Equation (3.7) as

$$\beta(x-t) = \sum_{i=0}^{\infty} \frac{(-1)^i}{i!} t^i \frac{d^i \beta(x)}{dx^i} = \sum_{i=0}^{\infty} \frac{t^i}{i!} A_i(x) \beta(x). \tag{3.8}$$

It follows that

$$A_i(x) = x^i - \frac{i^{[2]}}{2.1!} x^{i-2} + \frac{i^{[4]}}{2^2.2!} x^{i-4} - \frac{i^{[6]}}{2^3.3!} x^{i-6} + \cdots \tag{3.9}$$

where $i^{[k]} = \binom{i}{k} = \frac{i!}{(i-k)!}$. The Hermite polynomial is used as basis functions to approximate the unknown density function in Gram-Charlier series expansion.

The first 6 polynomials are

$$A_0 = 1,\ A_1 = x,$$

$$A_2 = x^2 - 1,\ A_3 = x^3 - 3x,$$

$$A_4 = x^4 - 6x^2 + 3,\ A_5 = x^5 - 10x^3 + 15x,$$

$$A_6 = x^6 - 15x^4 + 45x^2 - 15. \tag{3.10}$$

The Hermite polynomials are orthogonal to each other with respect to the kernel, β(x). Next section first proves this orthogonal property and then derives the coefficients, c_t in terms of central moments of ρ(x).

3.3 Orthogonal Property of Hermite Polynomials

Hermite polynomials have an important orthogonal property that is defined as

$$\int_{-\infty}^{\infty} A_m(x)A_n(x)\beta(x)dx = \begin{cases} 0, m \neq n \\ n!, m = n \end{cases} \qquad (3.11)$$

To prove the orthogonal property of polynomials given in Equation (3.11), one can integrate by parts and using Equation (3.6), (n ≥ m)

$$\int_{-\infty}^{\infty} A_m A_n \beta(x)dx = (-1)^n \int_{-\infty}^{\infty} A_m \frac{d^n \beta(x)}{dx^n}dx,$$

$$\int_{-\infty}^{\infty} A_m A_n \beta(x)dx = (-1)^n \left[A_m \frac{d^{n-1}\beta(x)}{dx^{n-1}} \right]_{-\infty}^{\infty} + (-1)^{n+1} \int_{-\infty}^{\infty} \frac{dA_m}{dx} \frac{d^{n-1}\beta(x)}{dx^{n-1}}dx, \qquad (3.12)$$

where the term in brackets goes to zero. For the last term in Equation (3.12), let's use the equation given below

$$\frac{dA_i(x)}{dx} = iA_{i-1}(x),$$

$$\frac{d^j A_i(x)}{dx^j} = \binom{i}{j} A_{i-j}(x),$$

$$\binom{i}{j} = \frac{i!}{(i-j)!}. \tag{3.13}$$

By the help of Equation (3.13), the last integration in Equation (3.12) becomes

$$m(-1)^{n+1} \int_{-\infty}^{\infty} A_{m-1} \frac{d^{n-1}\beta(x)}{dx^{n-1}} dx. \tag{3.14}$$

If we continue taking integral by parts, we find either zero or m ! if m = n.

Suppose now that a probability density function can be written as a series of derivatives of β(x). (Later on, conditions under which such an expansion is valid are given.) Then, we have

$$\rho(x) = \sum_{i=0}^{\infty} c_i A_i(x) \beta(x). \tag{3.15}$$

Multiplying by $A_r(x)$ and integrating from -∞ to ∞ and using the relationship given in Equation (3.11), namely orthogonal property, we get

$$c_r = \frac{1}{r!} \int_{-\infty}^{\infty} \rho(x) A_r(x) dx. \tag{3.16}$$

Because of the orthogonal property, Gram-Charlier coefficients are independent, not correlated with each other, and represent the statistical differences between noise and backscattered echoes. As a result, Gram-Charlier coefficients are used for the problem of classification. Substituting the value of $A_r(x)$ given in Equation (3.9) into Equation (3.16), we find

$$c_r = \frac{1}{r!} \left\{ \Lambda_r - \frac{r^{[2]}}{2.1!} \Lambda_{r-2} + \frac{r^{[4]}}{2^2 .2!} \Lambda_{r-4} - \cdots \right\}. \tag{3.17}$$

In particular, for moments,

$$c_0 = 1,\ c_1 = \Lambda_1,\ c_2 = \frac{1}{2}(\Lambda_2 - 1),$$

$$c_3 = \frac{1}{6}\Lambda_3,\ c_4 = \frac{1}{24}(\Lambda_4 - 6\Lambda_2 + 3),$$

$$c_5 = \frac{1}{120}(\Lambda_5 - 10\Lambda_3),$$

$$c_6 = \frac{1}{720}(\Lambda_6 - 15\Lambda_4 + 45\Lambda_2 - 15). \tag{3.18}$$

As a result, we find the expansion for the probability density function

$$\rho(x) = \beta(x)\left\{1+\frac{1}{2}(\Lambda_2 - 1)A_2 + \frac{1}{6}\Lambda_3 A_3 + \frac{1}{24}(\Lambda_4 - 6\Lambda_2 + 3)A_4 + \cdots\right\}. \qquad (3.19)$$

If the ρ(x) is in standard measure which means that its variance is unity and its mean is equal to zero, the series become

$$\rho(x) = \beta(x)\left\{1+\frac{1}{6}\Lambda_3 A_3 + \frac{1}{24}(\Lambda_4 - 3)A_4 + \cdots\right\} \qquad (3.20)$$

$$m_k = \frac{1}{n}\sum_{i=1}^{n}[x(i)]^k,$$

$$\Lambda_3 = m_3 - 3m_2 m_1 + 2m_1^3,$$

$$\Lambda_4 = m_4 - 4m_3 m_1 + 6m_2 m_1^2 - 3m_1^4,$$

$$\Lambda_5 = m_5 - 5m_4 m_1 + 10m_3 m_1^2 - 10m_2 m_1^3 + 4m_1^5,$$

$$\Lambda_6 = m_6 - 6m_5 m_1 + 15m_4 m_1^2 - 20m_3 m_1^3 + 15m_2 m_1^4 - 5m_1^6 . \qquad (3.21)$$

Equation (3.20) is the so-called Gram-Charlier series [40]. In this chapter it has assumed that the probability density function possesses a convergent Gram-Charlier series. Next section introduces the Cramer theorem to give out the conditions under which Gram-Charlier series is convergent.

3.4 Convergence of Gram-Charlier Series

Cramer [40] stated that if ρ(x) is a function which has continuous derivative such that

$$\int_{-\infty}^{\infty}\left(\frac{d\rho(x)}{dx}\right)^2 e^{-\frac{1}{2}x^2}\,dx, \tag{3.22}$$

the integral in Equation (3.22) converges and ρ(x) tends to zero as $|x|$ tends to infinity, then ρ(x) may be developed in series given in Equation (3.15) and c_r's are calculated using the integral given in Equation (3.16). These coefficients are used as an input to PDNN to detect the target echoes. The design technique is introduced in the next section.

3.5 Probabilistic Deconvolution Neural Networks

The block diagram for PDNN is shown in Figure 3.1. Note that the length of the target echo, n, is 39 points. Training of PDNN is achieved using a sliding window from two classes of signals (see Chapter 2 for a sliding window technique). One class represents the noise and the other one represents the target plus noise. The coefficients c_i for $i > 6$ are small compared to other c_i for $i < 6$. Therefore, the coefficients c_3, c_4, c_5, and c_6 are used as features for classification using neural networks trained by a backpropagation algorithm. The output of the neural network is 1 representing target plus noise or 0 representing noise only. In neural network design, a key issue is determining the number of hidden neurons. Adaptive hidden neuron algorithm was utilized in the decision of selecting the optimal number of hidden neurons which was 58 for PDNN (see Chapter 2 for more information about adaptive hidden neuron algorithm). Training results for PDNN are illustrated in Table 3.1. It should be noted that a total of 3615 training sets was applied to PDNN 275100 times to reach a sum-squared-error of 0.02. For adequate training, training set consisted of samples from one, two and three echoes with and without noise. Training SNR was between 8-16 dB. Some testing results along with the figures are presented in the next section. Furthermore, Hinton diagrams are introduced to explain how PDNN operates.

Table 3.1 Training Results for PDNN

	PDNN
Number of Sample Vectors	3615
Sum-Squared-Error	0.02
Number of Inputs	4
Number of Outputs	1
Number of Hidden Neurons	58
Epoch	275100
SNR	8-16 dB

3.6 Testing Results of PDNN

The performance of PDNN is evaluated using simulated data. Figure 3.3a depicts a testing result for one echo located at location 97. (SNR = 3 dB). Detected echo is at location 97 in Figure 3.3b. Figure 3.3c shows two interfering echoes located at locations 97 and 99 with signal-to-noise ratio 8 dB. Echoes are detected at locations 97 and 99 in Figure 3.3d. Figure 3.4a presents two echoes which are located at locations 97 and 100 with SNR of 8 dB. Echoes are detected at locations 97 and 100 in Figure 3.4b. Figure 3.4c shows three echoes located at locations 97, 99 and 101. SNR is 8 dB. Echoes are detected at locations 97, 99 and 101 in Figure 3.4d. Results show that PDNN is able to correctly detect the multiple interfering echoes in noisy environments (SNR = 3 dB). This performance is accomplished with a total number of 290 weights, however DNN has 1840 weights in its structure to reach the same performance. Using Gram-Charlier coefficients lessens the number of weights tremendously. Figure 3.5 depicts the Hinton diagram of PDNN. Note that the number of hidden neurons is 58 which was acquired by adaptive hidden neuron algorithm. Results obtained testing PDNN (see Figures 3.3 and 3.4) prove that the optimal design of PDNN has been accomplished.

3.7 Conclusion

In Chapter 2, deconvolution neural networks are utilized to detect multiple flaw echoes in noisy environments (SNR = 8 dB). For lower SNRs, the ANN has been developed to remove the noise from the input signal. And, then the improved signal is applied to DNN for better detection (SNR = 4 dB). The purpose of Chapter 3 is to introduce a single neural network model that achieves the same performance as that of DNN and ANN with less number of weights. The PDNN has been developed utilizing the Gram-Charlier coefficients of flaw and scattering noise. Employing Gram-Charlier

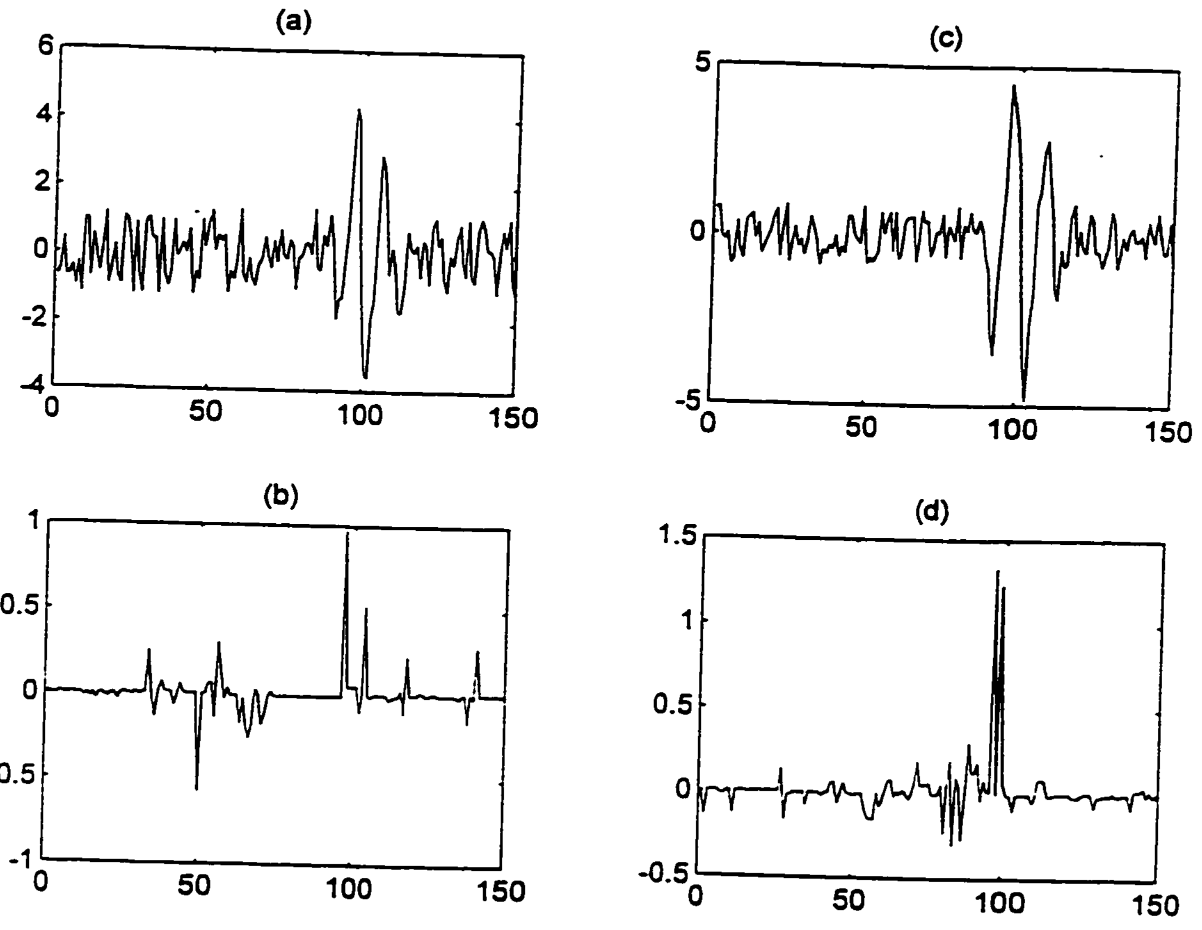


Figure 3.3 (a) One Echo is at Location 97 with SNR = 3 dB, (b) Output of PDNN, and Detected Echo is at Location 97, (c) Two Echoes are at Locations 97 and 99 with SNR = 8 dB, (d) Output of PDNN, and Detected Echoes are at Locations 97 and 99

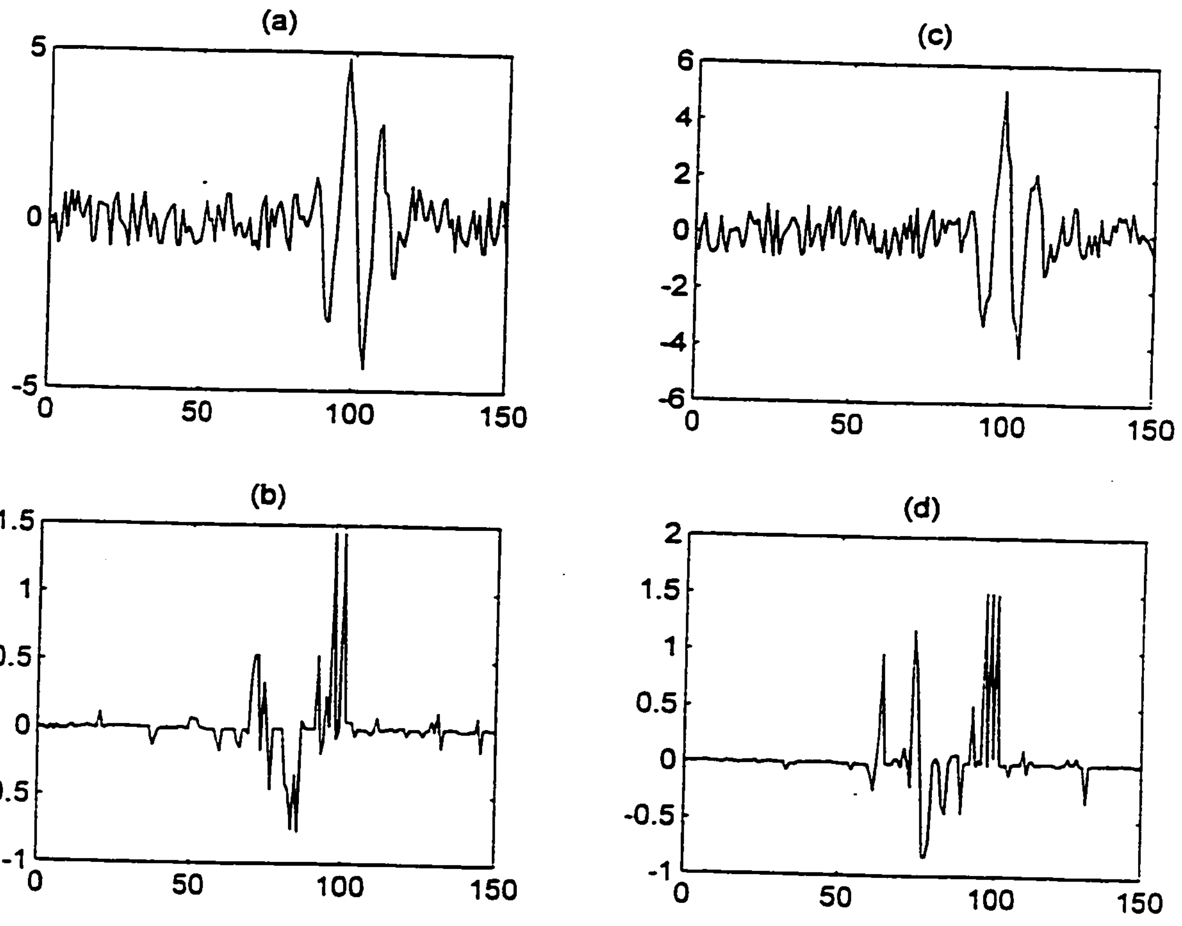


Figure 3.4 (a) Two Echoes are at Locations 97, 100 with SNR = 8 dB, (b) Output of PDNN, Detected Echoes are at Locations 97 and 100, (c) Three Echoes are at Locations 97, 99, and 101 with SNR = 8 dB, (d) Output of PDNN, Detected Echoes are at Locations 97, 99 and 101

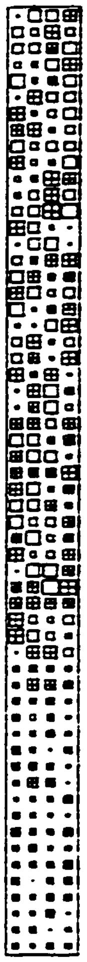


Figure 3.5 Hinton Diagram of PDNN

coefficients in the design of PDNN offer practical advantages such as the total number of input neurons becomes 4 instead of 39 which is the length of the target echo. One can still use the same neural network model even if the length of the echo is different than 39. Results show that the PDNN is trainable and adaptable for any size of flaws and deconvolves them successfully in noisy environments.

Utilizing adaptive hidden neuron algorithm with backpropagation learning algorithm in the training phase of PDNN searches for the optimal number of hidden neurons in the hidden layer. This searching algorithm avoids overfitting and underfitting problems and helps the PDNN to escape from local minimums and reach the global minimum.

Hinton diagrams have been examined to find out if PDNN has the optimal weights or not. Examination of these figures and experimental results reveal that PDNN can be employed effectively for the problems of ultrasonic imaging systems.

CHAPTER IV

ORDER STATISTIC FILTERS

This chapter presents another technique that utilizes the split-spectrum processing in order to detect target echoes in noisy environments. We developed neural network filters to substitute the conventional filters that are used in the split-spectrum processing. These filters are called minimum neural network order statistic filter (MinNNet), maximum neural network order statistic filter (MaxNNet), and median neural network order statistic filter (MedNNet) which find minimum, maximum, and median rank of the input signal respectively. The purpose of this chapter is to evaluate the robustness of these neural networks using simulation and experimental data. Results obtained in the performance analysis of these filters indicate that neural network order statistic filters can be efficiently utilized in split-spectrum processing to detect flaw echoes in grain scattering noise.

4.1 Introduction

Order statistic filtering has been widely used in the field of signal and image processing [46, 62, 63, 64, 82, 87]. The main idea behind an order statistic filter is that finding the m-th largest element in an input signal. Different values for m result in various members of order statistic filters [82]. Some of the members, such as minimum, median, and maximum filters, have been used extensively in signal and image processing.

In this chapter, three novel neural network models are introduced to find minimum, median, and maximum of the input signal. To solve the sorting problem by using neural networks, backpropagation learning algorithm (see Chapter 1) and adaptive hidden neuron algorithm (see Chapter 2) are utilized in the construction of the neural network filters. Our primary concern lies in the preparation of the training matrix that is used in the training phase [5, 20, 24]. If the length of the input data is n, there are n! different signal patterns of the input data that gives the same output. It is difficult to train a neural network filter with that many input data. As a result, the neural network filter might not provide 100 % accurate results. In spite of this drawback, neural network filters provide good approximation other than exact values for the sorted result and perhaps from statistical point of view, this might be sufficient for the problem of sorting. In certain application, it is desirable to develop a novel neural network model of high processing speed which can be used in finding the estimates of minimum, median, and maximum of the input signal. To achieve this, simulation data is used in the training phase. The training set of data consists of uniform random numbers as an input to the neural network. The output is minimum, median or maximum rank of input numbers. After the training phase, the neural network filter weights are used to derive the probability density function (pdf) of the output. This provides us some analytical point of view why neural network filters are able to find good estimates of ranks. In addition to pdf method, Hinton diagrams are also utilized to examine the optimality of neural network weights [15, 21, 49].

The remainder of this chapter is arranged as follows: Section 4.2 introduces the pdf of the order statistic filters. Section 4.3 derives the probability density function for neural network filters. Section 4.4 presents the design technique for the neural network filter, and explains how a neural network filter operates using probability density functions of the hidden layer neurons and Hinton diagrams. Section 4.5 presents the performance evaluation of the neural networks and the testing results. Section 4.6 utilizes the neural network filters in split spectrum processing (SSP) to detect the target echoes in noisy environments.

4.2 The PDF of Order Statistic Filters

The order statistic filter accepts an array of n real numbers as an input and outputs the m-th rank of these numbers. The m-th rank is equal to one of the n real numbers that is less than or equal to n-m real numbers and greater than or equal to m-1 values. The order statistic (OS) filter can be defined as

$$x_{(m)} = OS_{input}(x_1, x_2, \cdots, x_n), \quad for \;\; 1 \leq \mathrm{m} \leq \mathrm{n} \qquad (4.1)$$

where x_i (i = 1,...,n) is the unordered real number in the array, n is the total number of real numbers in that array, and m is the rank of the ordered array of real numbers that becomes the output, $x_{(m)}$, of the order statistic filter. The values for m and n could be any number, but in our design of neural network order statistic filters, n is 40, m is 1, 20 and 40 for MinNNet, MedNNet, and MaxNNet respectively. The output of these neural networks are the estimates of their respective ranks. These outputs are not the exact values of one of the neural network input values (see Equation (4.1)). However, from the statistical point of view, these estimates might be successfully used instead of the actual values of the ranks. In areas where the target can be differentiated from the noise statistically, the neural network filters might give superb responses, and their approximations of the desired values as well as the speed of their responses might detect the target accurately in highly noisy environments.

Order statistic filters have been used extensively in both image and signal processing, such as they are used for noise power estimation of sonar signals for rejection of bad data (outliers) [87]. They also have been utilized in processing image [46], radar, speech [55], and ultrasound signals because of their noise suppression and feature preservation properties.

Order statistic filters involve finding the relationship between the input and the output of statistical behavior of the data. This is particular interest in the ultrasonic defect echoes in high level of scattering noise [64]. Let's assume x_i's are independent and identically distributed (i.i.d.) with the probability distribution function $F_X(\bullet)$, and the probability density function $f_X(\bullet)$. The distribution function for the output of the OS filter is given by [64]

$$F_{X_{(m)}}(x_{(m)}) = \frac{n!}{(m-1)!(n-m)!} F_X^{m-1}(x_{(m)})(1 - F_X(x_{(m)}))^{n-m} f_X(x_{(m)}) dx_{(m)} \qquad (4.2)$$

for $1 \leq m \leq n$, where $x_{(m)}$ represents a real number from the ordered array, and X is the random variable for the unordered input of the order statistic filter, and $X_{(m)}$ is the random variable for the ordered array. The density function can be found by taking the derivative of Equation (4.2)

$$f_{X_{(m)}}(x_{(m)}) = \frac{n!}{(m-1)!(n-m)!} F_X^{m-1}(x_{(m)})(1 - F_X(x_{(m)}))^{n-m} \cdot f_X(x_{(m)}) \tag{4.3}$$

for $1 \leq m \leq n$. Equation (4.3) describes the statistics of the output of a general OS filter when the density function of the input signal is i.i.d. This equation demonstrates that the input density function, $f_X(\bullet)$, is weighted by another function that is known as sort function [64] and it is given by

$$g_{(m)}(z) = \frac{n!}{(m-1)!(n-m)!} z^{m-1}(1-z)^{n-m} \tag{4.4}$$

for $0 \leq z \leq 1$, where

$$z = F_X(x_{(m)}) \tag{4.5}$$

The expected value for the output of the order statistic filter is

$$E\{x_{(m)}\} = \int_0^1 F_X^{-1}(z) g_{(m)}(z) dz \tag{4.6}$$

The function, $g_{(m)}(\bullet)$, acts as a weighting function to emphasize a particular region of the inverse distribution function over the integration as given in Equation (4.6). Figure 4.1 shows the normalized sort functions for the minimum, the median and the maximum order statistic filters. This figure presents that the order statistic filtering operation filters out the signal samples outside the region marked by the filter, while it favors the ones that fall into the region. This is beneficial when there are some statistical differences between target and noise classes which can be acquired by neural network filters. Next section deals with deriving the pdf for a neural network filter.

4.3 The PDF of Neural Network Order Statistic Filters

Once the weights of the neural network order statistic filters are known, it is helpful deriving the pdf of them. Using these pdf's, one can decide if the ideal design of a neural network filter has been found or not. The remainder of this section mainly concerns of finding the pdf's of the neural network order statistic filters. From Figure 4.2, one can write N equations as follows assuming there are N hidden neurons in the hidden layer

$$
\begin{aligned}
y_{11} &= \tanh(w_{11}{}^{h}x_1 + w_{12}{}^{h}x_2 + \cdots + w_{1M}{}^{h}x_M) \\
y_{21} &= \tanh(w_{21}{}^{h}x_1 + w_{22}{}^{h}x_2 + \cdots + w_{2M}{}^{h}x_M) \\
&\vdots \\
y_{i1} &= \tanh(w_{i1}{}^{h}x_1 + w_{i2}{}^{h}x_2 + \cdots + w_{iM}{}^{h}x_M) \\
&\vdots \\
y_{N1} &= \tanh(w_{N1}{}^{h}x_1 + w_{N2}{}^{h}x_2 + \cdots + w_{NM}{}^{h}x_M) \qquad (4.7)
\end{aligned}
$$

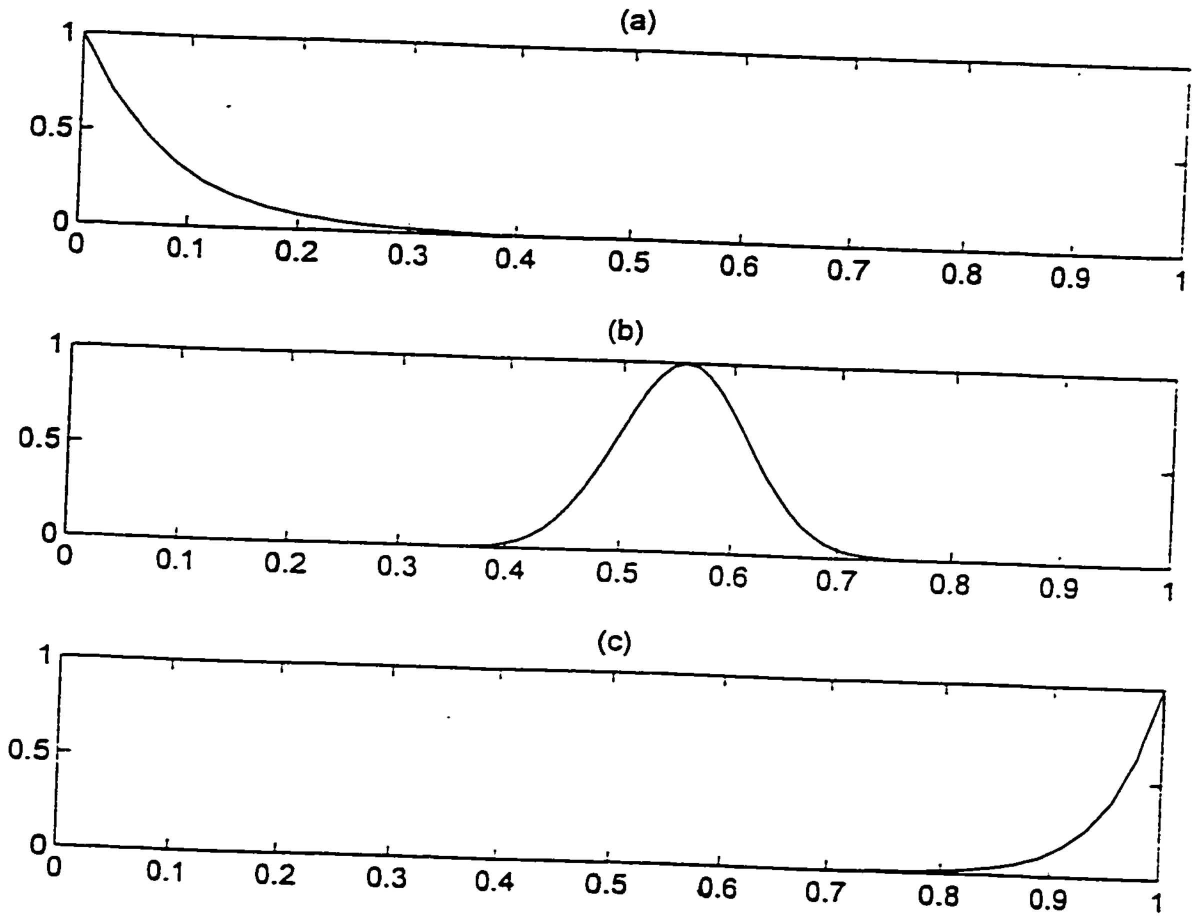


Figure 4.1 (a) The normalized sort function for Minimum OS Filter, (b) The normalized sort function for Median OS Filter, (c) The normalized sort function for Maximum OS Filter

where i = 1, ..., N. To find the pdf of y_{11}, which is one function of M random variables, we introduce auxiliary variables $y_1, y_2, \ldots, y_{M-1}$ which are defined below

$$\begin{aligned}
y_{11} &= \tanh(w_{11}{}^h x_1 + w_{12}{}^h x_2 + \cdots + w_{1M}{}^h x_M) \\
y_1 &= x_1 \\
y_2 &= x_2 \\
&\vdots \\
&\vdots \\
y_{M-1} &= x_{M-1}
\end{aligned} \tag{4.8}$$

After finding the pdf of y_{11}, the pdf's for the rest of the hidden neurons are similar to that of y_{11}. Above equations are to find pdf of y_{11} and similar procedures are used to find the pdf of $y_{21}, \ldots, y_{i1}, \ldots, y_{N1}$.

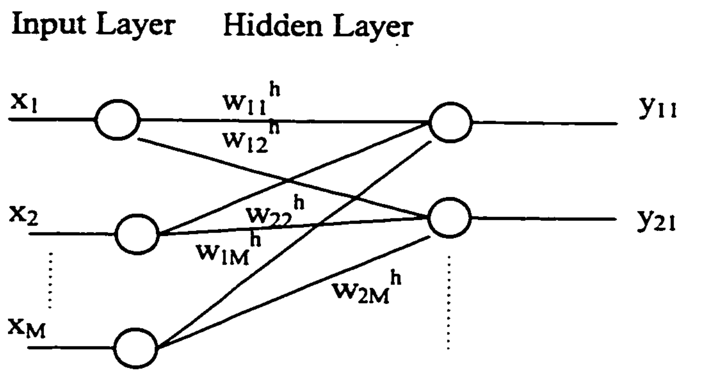


Figure 4.2 The Input and Hidden Layers of the Neural Network Filter

The Jacobian transformation (4.8) is given as follows

$$J(x_1,x_2,\cdots,x_M)=\begin{vmatrix}\frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_M}\\ \vdots & \vdots & \vdots\\ \frac{\partial y_{11}}{\partial x_1} & \cdots & \frac{\partial y_{11}}{\partial x_M}\end{vmatrix}$$

$$J=\begin{vmatrix}1 & \cdots & 0\\ 0 & \cdots & 0\\ \frac{\partial y_{11}}{\partial x_1} & \cdots & \frac{\partial y_{11}}{\partial x_M}\end{vmatrix}=\left|\frac{\partial y_{11}}{\partial x_M}\right|=w_{1M}{}^h(1-y_{11}{}^2) \qquad (4.9)$$

then the joint density function of (4.8) can be found as

$$f_{Y_{11}Y_1\cdots Y_{M-1}}(y_{11},y_1,y_2,\cdots,y_{M-1})=\frac{f_X(x_1,x_2,\cdots,x_M)}{|J(x_1,x_2,\cdots,x_M)|}=\frac{f_X(x_1,x_2,\cdots,x_M)}{|w_{1M}{}^h(1-y_{11}{}^2)|} \qquad (4.10)$$

Since x_i's are assumed to be i.i.d., their joint density function can be written as

$$f_X(x_1,x_2,\cdots,x_M)=f_{X_1}(x_1)f_{X_2}(x_2)\cdots f_{X_M}(x_M) \qquad (4.11)$$

then, the probability density function of y_{11} can be found by integrating the joint density function that is given in (4.10)

$$f_{Y_{11}}(y_{11})=\iiint\cdots\int\frac{f_{X_1}(x_1)f_{X_2}(x_2)\cdots f_{X_M}(x_M)}{|w_{1M}{}^h(1-y_{11}{}^2)|}dx_1dx_2\cdots dx_{M-1} \qquad (4.12)$$

which can be numerically calculated.

Then, the pdf of y_{i1} ($i = 1, \ldots, N$) results in

$$f_{Y_{i1}}(y_{i1}) = \iiint \cdots \int \frac{f_{X_1}(x_1) f_{X_2}(x_2) \cdots f_{X_M}(x_M)}{\left| w_{iM}{}^h (1 - y_{i1}{}^2) \right|} dx_1 dx_2 \cdots dx_{M-1} \tag{4.13}$$

After finding the pdf's for the hidden layer output neurons, we can proceed to the output neuron to find the pdf of the neural network filter, and the output of the neural network filter is presented as (see Figure 4.3).

Output Layer

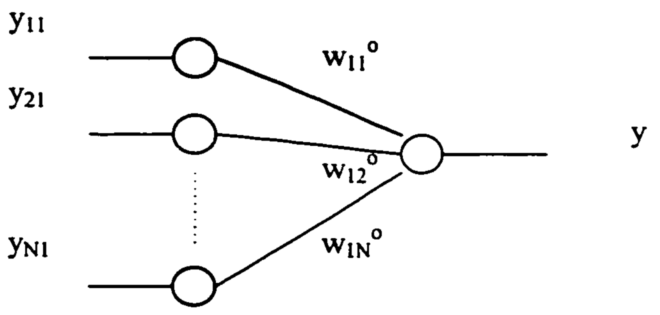


Figure 4.3 The Output Layer of the Neural Network Filter

$$\begin{aligned} y &= w_{11}{}^o y_{11} + w_{12}{}^o y_{21} + \cdots + w_{1N}{}^o y_{N1} \\ y &= u_{11} + u_{12} + \cdots + u_{1N} \end{aligned} \tag{4.14}$$

Then, the pdf of u_{11} is formed as

$$f_{U_{11}}(u_{11}) = \frac{1}{\left|w_{11}{}^{o}\right|} f_{Y_{11}}\left(\frac{u_{11}}{w_{11}{}^{o}}\right) \tag{4.15}$$

Then, the pdf for u_{1i} is represented as

$$f_{U_{1i}}(u_{1i}) = \frac{1}{\left|w_{1i}{}^{o}\right|} f_{Y_{1i}}\left(\frac{u_{1i}}{w_{1i}{}^{o}}\right) \tag{4.16}$$

where i = 1, ..., N

Assuming u_{1i}'s are independent random variables, the pdf of the neural network filter is presented as

$$f_Y(y) = f_{U_{11}}(u_{11}) * f_{U_{12}}(u_{12}) * \cdots * f_{U_{1N}}(u_{1N}) \tag{4.17}$$

where * represents the convolution operation. Equation (4.17) which depicts the formula for the pdf of neural network filter is not as simple as the pdf of an order statistic filter that is presented in Equation (4.3) due to the non-linearity property of the hyperbolic tangent transfer function. One can try an alternative approach to find a simple expression for the pdf of a neural network filter to prove that the ideal structure of a neural network filter has been found. Using Figure 4.4, one can write N equations as below

$$\begin{aligned}
z_{11} &= w_{11}{}^h x_1 + w_{12}{}^h x_2 + \cdots + w_{1M}{}^h x_M \\
z_{12} &= w_{21}{}^h x_1 + w_{22}{}^h x_2 + \cdots + w_{2M}{}^h x_M \\
&\vdots \\
z_{i1} &= w_{i1}{}^h x_1 + w_{i2}{}^h x_2 + \cdots + w_{iM}{}^h x_M \\
&\vdots \\
z_{N1} &= w_{N1}{}^h x_1 + w_{N2}{}^h x_2 + \cdots + w_{NM}{}^h x_M
\end{aligned} \qquad (4.18)$$

where x_i's are i.i.d. uniform random variables and using the central limit theorem results in the probability density function of the hidden neuron output as

$$f_{z_{i1}}(z_{i1}) \cong \frac{1}{\sigma_i\sqrt{2\pi}} e^{\frac{-1}{2}\left(\frac{z_{i1}-\eta_i}{\sigma_i}\right)^2} \qquad (4.19)$$

where the mean η_i is given as

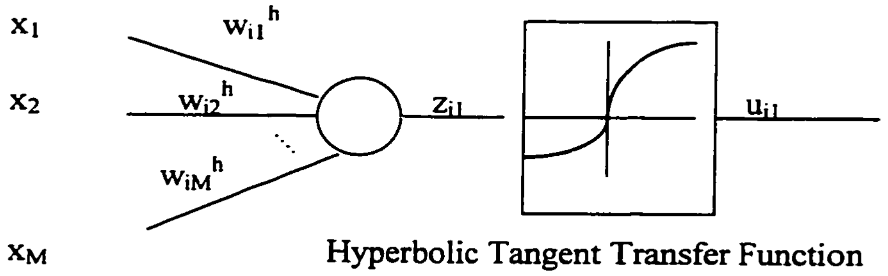


Figure 4.4 One Hidden Neuron with its Hyperbolic Tangent Transfer Function

$$\begin{aligned}
\eta_i &= E\{z_{i1}\} = w_{i1}{}^h E\{x_1\} + w_{i2}{}^h E\{x_2\} + \cdots + w_{iM}{}^h E\{x_M\} \\
\eta_i &= \eta\{w_{i1}{}^h + w_{i2}{}^h + \cdots + w_{iM}{}^h\} \\
\eta &= E\{x_1\} = E\{x_2\} = \cdots = E\{x_M\}
\end{aligned} \qquad (4.20)$$

and the variance, σ_i^2, is found as

$$\begin{aligned} \sigma_i^2 &= \sigma_{i1}^2 + \sigma_{i2}^2 + \cdots + \sigma_{iM}^2 \\ \sigma_i^2 &= \sigma^2\left\{w^2{}_{i1}{}^h + w^2{}_{i2}{}^h + \cdots + w^2{}_{iM}{}^h\right\} \\ \sigma^2 &= E\left\{(x_j - \eta)^2\right\} \end{aligned} \tag{4.21}$$

where i = 1, ..., N, and j = 1, ..., M. The output after the hyperbolic tangent transfer function, u_{i1}, is defined as (see Figure 4.4)

$$u_{i1} = \tanh(z_{i1}) \tag{4.22}$$

From Equation (4.22), one can write the probability density function of the hidden layer neurons as

$$\begin{aligned} f_{U_{i1}}(u_{i1}) &= \frac{1}{\left|1 - u_{i1}^2\right|} f_{Z_{i1}}(z_{i1} = \arctan h(u_{i1})) \\ \arctan h(u_{i1}) &= \frac{-1}{2}\ln\left(\frac{1 - u_{i1}}{1 + u_{i1}}\right) \end{aligned} \tag{4.23}$$

where i = 1, ..., N. The effect of the hyperbolic tangent function can be elaborated by looking at the two pdf's that are given in Equations (4.17) and (4.23). These two equations can be used to investigate the optimality of the neural network filters. It can be further addressed by using the histograms of their output data. By examining these histograms, one can decide if the correct weights are found or not. In the next section, the neural network order statistic filter structures are introduced and the probability density functions, the histograms of the output data, and the Hinton diagrams are presented to explain how they work.

4.4 The Design of a Neural Network Filter

Figure 4.5 displays the structure for the order statistic neural network. As mentioned before, the number of inputs is chosen to be 40 (M = 40, but M could be chosen to be any number). The neural network filter is a fully connected feedforward neural network. When an unordered uniform random numbers are presented as an input to the neural network, each output of the hidden neurons is weighted sum of the input nodes passed through a hyperbolic tangent function. Then, the output rank of the neural network is the weighted sum of all the hidden neurons. This actual output rank of the neural network is then compared to the desired rank for every set of input values which are uniform random numbers. Furthermore, the neural network error is defined as the difference of these outputs. Then, the weights of the neural network are updated using the gradient of this output error.

An adaptive hidden neuron algorithm is also applied to the neural network filters during the process of training. The purpose of this algorithm is finding the optimal number of hidden neurons [15, 21, 49] (see Chapter 2 for adaptive hidden neuron algorithm).

The training matrix is prepared using uniform random numbers between 0 and 1. Each column in the training matrix represents one set of these random numbers with a length of 40. Note that the size of the training set could be any number. The bigger the number, the better estimation of target ranks is achieved by the neural networks. The training matrix is defined as

$$X = \begin{bmatrix} x_1 & x_{M+1} & \cdots & x_{M+K} \\ x_2 & x_{M+2} & \cdots & x_{M+K+1} \\ \vdots & \vdots & \cdots & \vdots \\ x_M & x_{2M} & \cdots & x_{2M+K-1} \end{bmatrix} \tag{4.24}$$

where one column of x_i's are chosen randomly from subsets of [0,1], and K could be any number. The values for the boundaries of these subsets are chosen randomly, and the total number of training vectors that are in each subset are also selected randomly. The training matrix is not adequate if it only holds numbers from one subset whose boundaries are equal to 0 and 1. Because if a testing vector that has numbers from certain subsets of [0, 1] is given to a neural network filter as an input, the outcome would be a failure. To overcome this inadequate training problem, the training matrix should have enough samples from distinct subsets of [0,1]. The output of the neural network (see Figure 4.5) can be found as

$$\vec{y} = W^o \tanh(W^h X),$$

$$\vec{y} = [y(1), \ldots, y(T)] \tag{4.25}$$

where T is the total number of columns in X, y(i) {i = 1,...,T} is the estimate of the rank, m, of i-th column in the training matrix, W^o, and W^h are weight matrixes for output and hidden layers respectively:

$$W^o = [w_{11}{}^o \cdots w_{1j}{}^O \cdots w_{1L}{}^o], \quad W^h = \begin{bmatrix} w_{11}{}^h & w_{12}{}^h & \cdots & w_{1M}{}^h \\ w_{21}{}^h & w_{22}{}^h & \cdots & w_{2M}{}^h \\ \cdots & \cdots & \cdots & \cdots \\ w_{L1}{}^h & w_{L2}{}^h & \cdots & w_{LM}{}^h \end{bmatrix}, \tag{4.26}$$

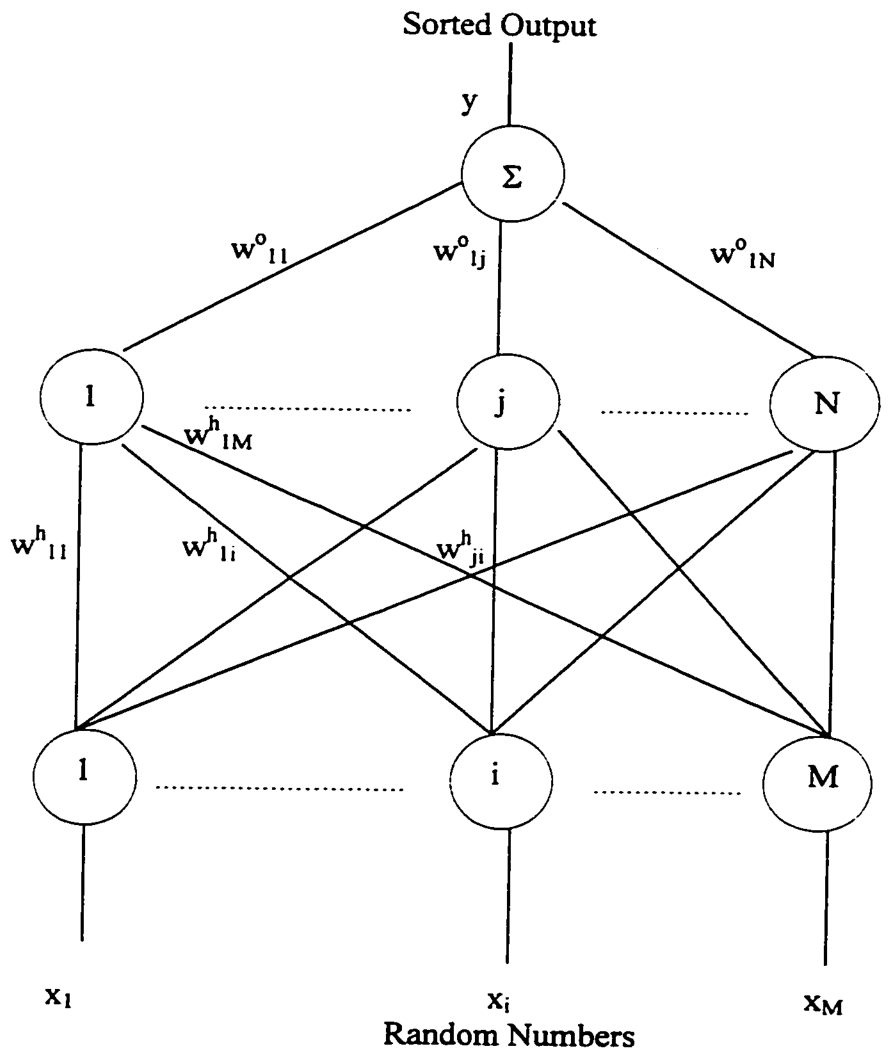


Figure 4.5 The Neural Network Order Statistic Filter

where L = 15, 10 and 16 for MinNNet, MedNNet, and MaxNNet respectively.

Training statistics of the neural network filters are given in Tables 4.1, 4.2 and 4.3. Note that MedNNet reaches a lower error (sum-square error, SSE, which is the sum of the square of differences between the actual and the desired ranks), 0.49, than that of the other two order statistic neural network filters MinNNet, and MaxNNet which are 2,44 and 2,98 respectively. And also MedNNet has less hidden neurons, 10, than those in MinNNet and MaxNNet which are 15 and 16 respectively. Total of 3615 sample vectors were applied to MinNNet, MedNNet, and MaxNNet 62225, 35685, and 65395 times respectively to update their neural network weights to find their optimal solutions. In the next paragraph, Hinton diagrams and pdf's are used to explain why MedNNet does a better sorting than the others.

Hinton diagrams of these filters are presented in Figures 4.6, 4.7 and 4.8 for MinNNet, MedNNet, and MaxNNet respectively to display their weights. When a column vector of random numbers are given to these neural network filters, the output is the estimate of the rank which is the closest number to one of their input random numbers. Note that the graphical distribution of weights in MinNNet and MaxNNet in Figures 4.6 and 4.8 display almost no random information (especially MaxNNet), but in Figure 4.7, the weights of MedNNet show some randomness buried in themselves. This might be the reason for the success of MedNNet over MinNNet and MaxNNet. The performance of these neural network filters can also be explained by examining the characteristics of the hyperbolic tangent transfer function. The MinNNet and MaxNNet

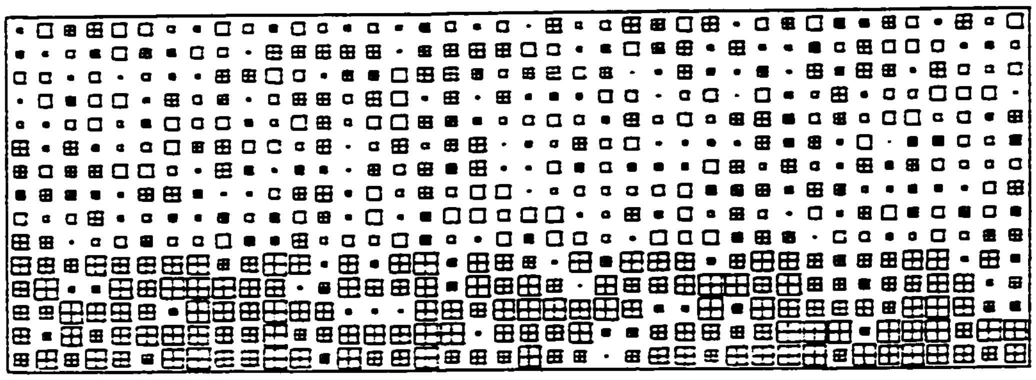


Figure 4.6 Hinton Diagrams for MinNNet

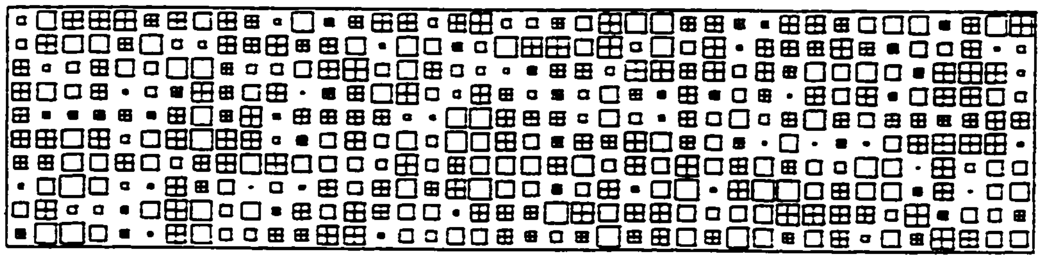


Figure 4.7 Hinton Diagrams for MedNNet

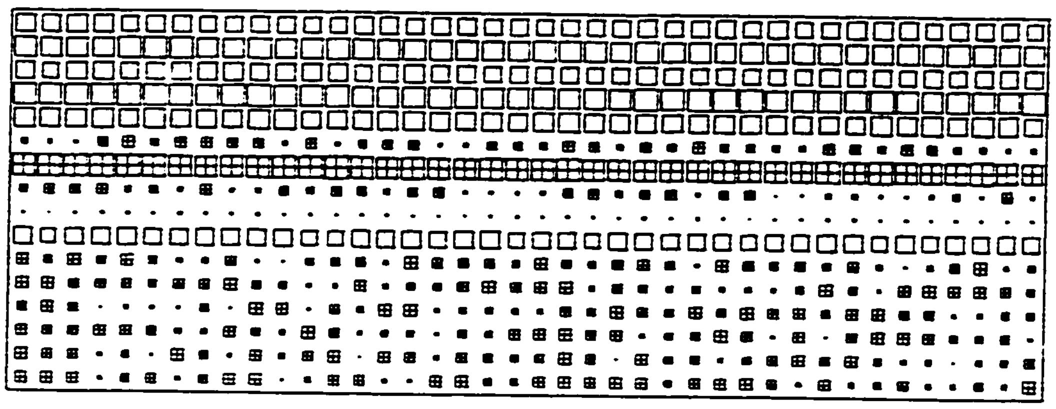


Figure 4.8 Hinton Diagrams for MaxNNet

Table 4.1 Training Statistics for MinNNet

Number of Sample Vectors	3615
Number of Inputs	40
Number of Outputs	1
Number of Hiddens	15
Epochs	62225
Error Goal	2.44

Table 4.2 Training Statistics for MedNNet

Number of Sample Vectors	3615
Number of Inputs	40
Number of Outputs	1
Number of Hiddens	10
Epochs	35685
Error Goal	0.49

have internal values that are on the theoretical minimum or maximum of the hyperbolic tangent functions. On these regions, there is less learning which means less contribution to the neural network weights. This might result in all the weights in MinNNet and MaxNNet filters to have similar magnitudes (see Figures 4.6 and 4.8). On the other hand, the MedNNet internal values fall into the middle region of the hyperbolic tangent function which is the linear region. In this region of hyperbolic tangent function, the weights are updated until they reach to the optimal weights.

Table 4.3 Training Statistics for MaxNNet

Number of Sample Vectors	3615
Number of Inputs	40
Number of Outputs	1
Number of Hiddens	16
Epochs	65395
Error Goal	2.98

4.5 The Performance of Neural Network Filters

Performance evaluation of the neural network filters is achieved by examining the pdf's of their hidden layer neurons. Probability density functions tell us that where the emphasize of the neural network hidden neurons is located at. If this emphasize is in the middle of the hyperbolic tangent transfer function for all the hidden neurons of MedNNet, we might say that the MedNNet does a good estimation on the median rank of the input signal. If the emphasize is in the minimum or the maximum region of the hyperbolic tangent transfer function for all the hidden neurons, then the MinNNet or the MaxNNet does a good estimation on finding their corresponding ranks. To find the output pdf of a neural network filter, we apply the simulated data and find the histogram of the actual pdf versus the desired pdf of the neural network. These histograms show us that how much the estimated rank is close to the expected rank and also help us

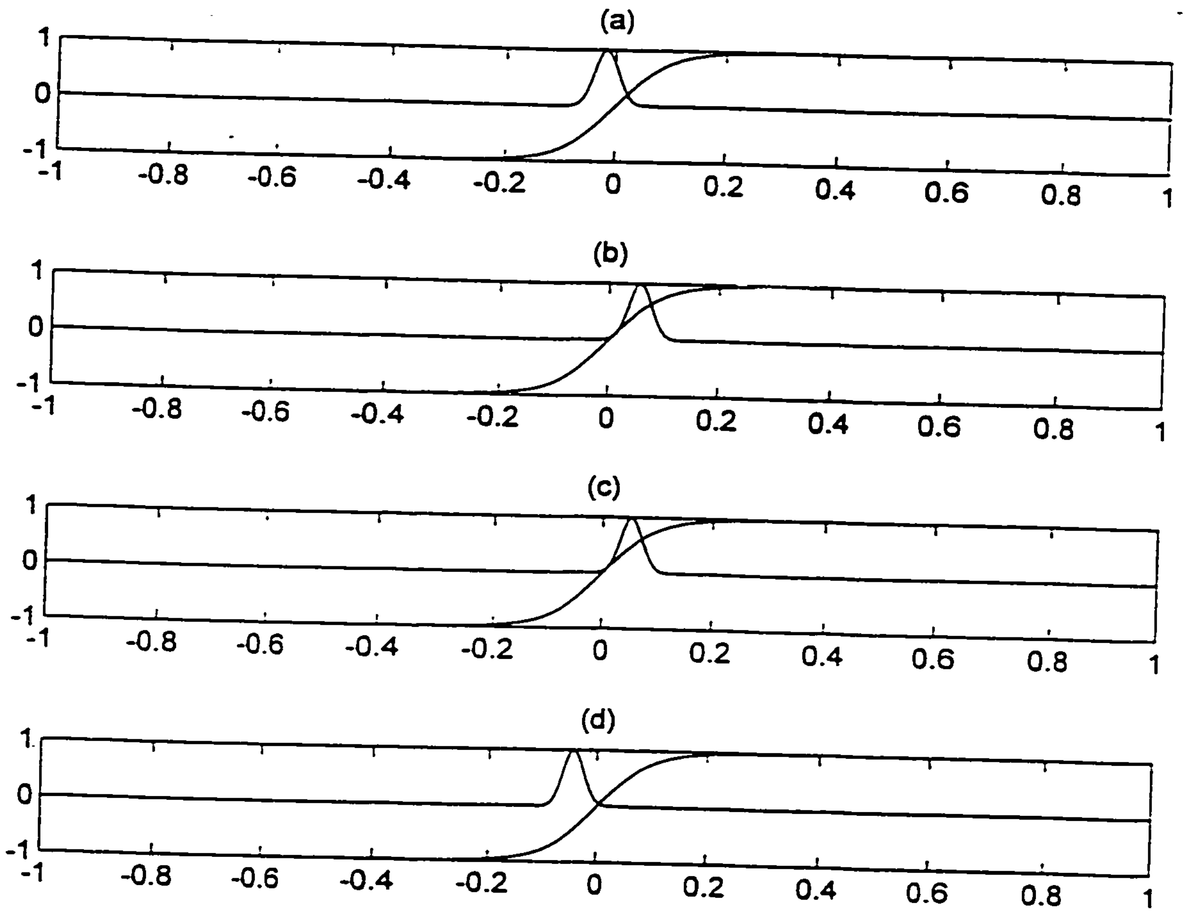


Figure 4.9 MinNNet: The normalized pdf's for Hidden Neurons (a) #1, (b) #2, (c) #3, and (d) #4 Before the application of Hyperbolic Tangent Transfer Function

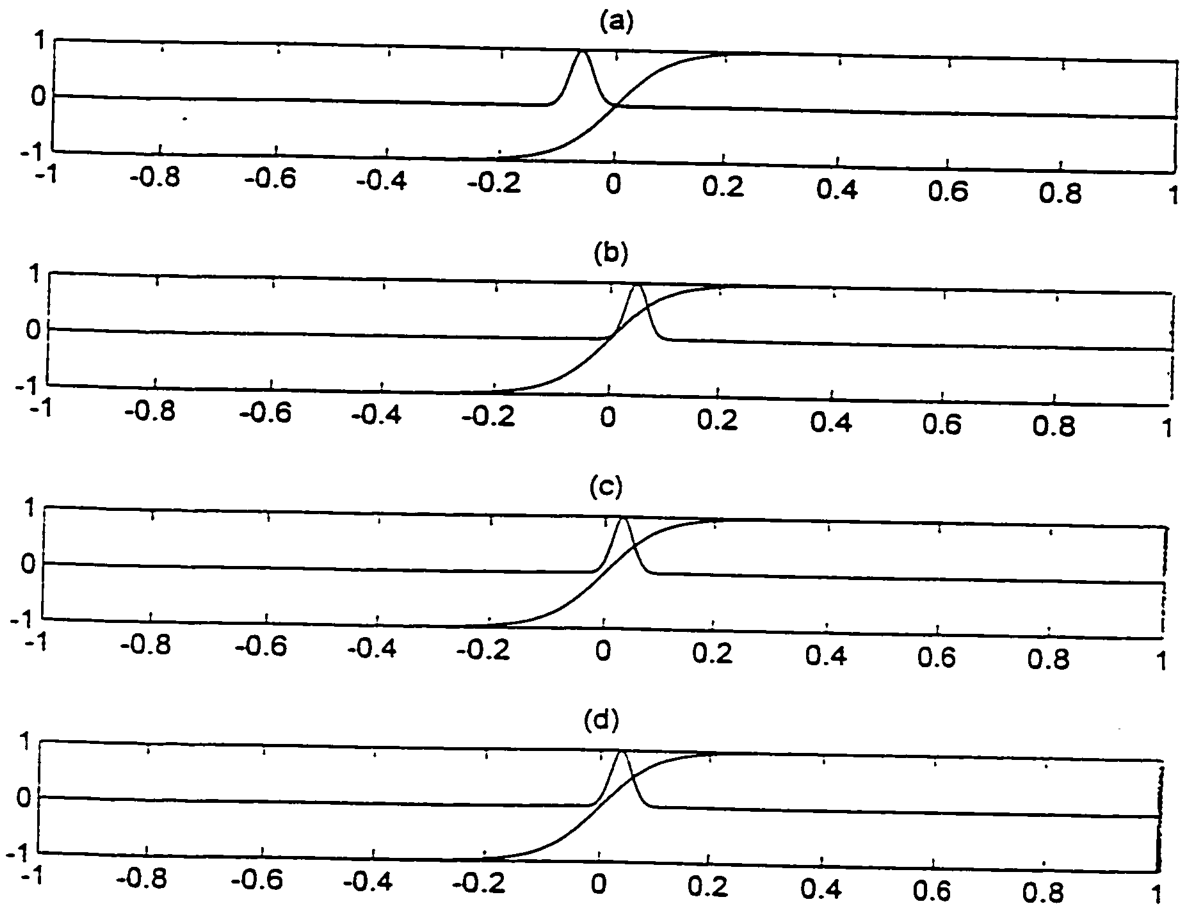


Figure 4.10 MinNNet: The normalized pdf's for Hidden Neurons (a) #5, (b) #6, (c) #7, and (d) #8 Before the application of Hyperbolic Tangent Transfer Function

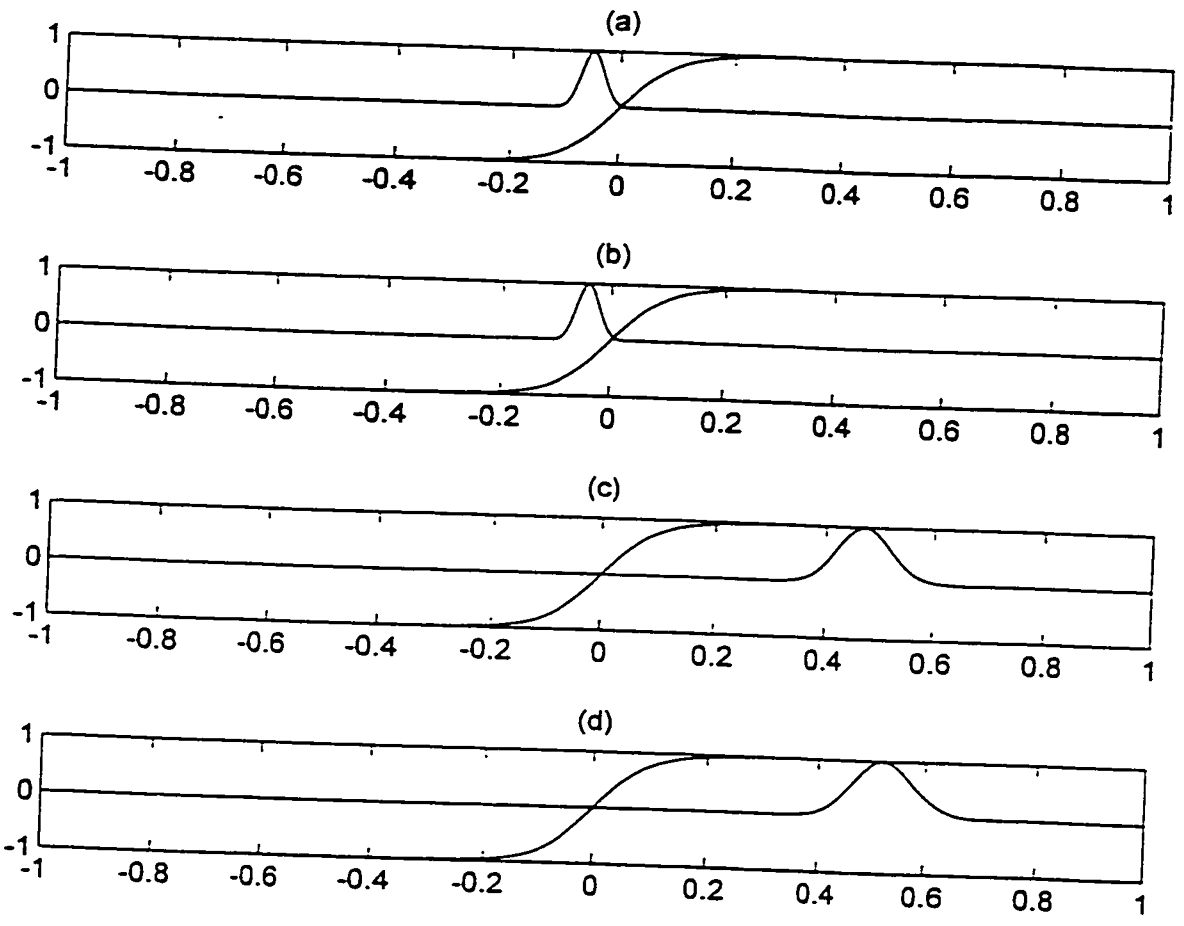


Figure 4.11 MinNNet: The normalized pdf's for Hidden Neurons (a) #9, (b) #10, (c) #11, and (d) #12 Before the application of Hyperbolic Tangent Transfer Function

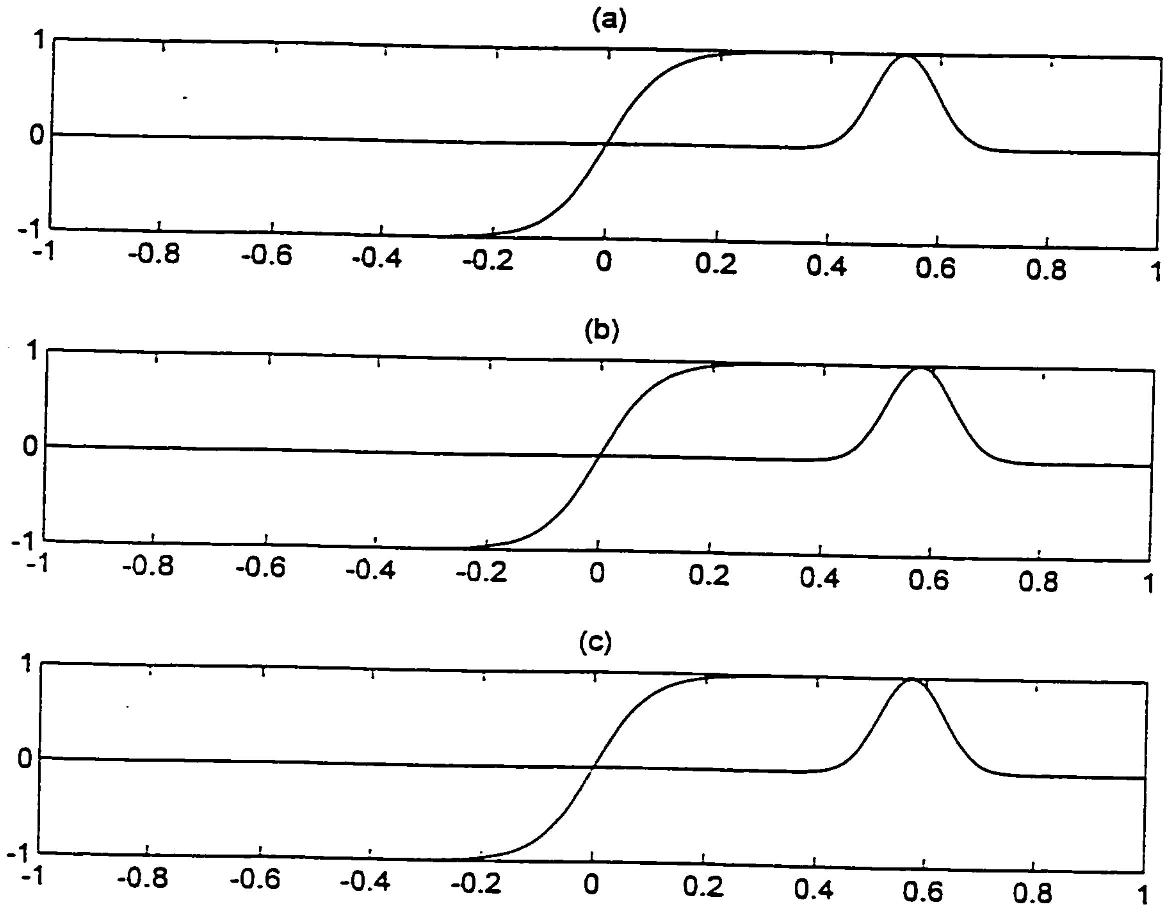


Figure 4.12 MinNNet: The normalized pdf's for Hidden Neurons (a) #13, (b) #14, and (c) #15 Before the application of Hyperbolic Tangent Transfer Function

predicting how well the outcomes are for the future testings of the neural network filters. The normalized pdf's that are given in Equation (4.19) are depicted for MinNNet in Figures 4.9 through 4.12. These are the normalized pdf's before the application of hyperbolic tangent transfer function. The normalized pdf's after the application of hyperbolic tangent function are presented in Figures 4.13 through 4.16 (see Equation (4.23)). Figure 4.17 shows the true histogram of the minimum rank for 2500 testing vectors as well as the actual histogram of the MinNNet along with the histogram of the input uniform random numbers. One can conclude from these figures that the ideal weights for the MinNNet were achieved. Same process was repeated for MedNNet. Figures 4.18 through 4.20 show the normalized pdf's of hidden neurons before the application of hyperbolic tangent function. It can be observed that the normalized pdf's are in the middle region of the hyperbolic tangent functions for all 10 hidden neurons. Figures 4.21 through 4.23 depict the figures of the normalized pdf's after the application of hyperbolic tangent. The influence is still in the middle region of the hyperbolic tangent transfer function for all the hidden neurons. Figure 4.24 shows the histograms of input, actual MedNNet and true median rank. The actual pdf is very close to that of true median rank. This shows that the MedNNet does a better sorting than that of MinNNet as well as MaxNNet. Figures 4.25 through 4.28 expose the normalized pdf's for the hidden neurons of MaxNNet before the application of hyperbolic tangent function. The effect of these normalized pdf's for most of the hidden neurons is close to the maximum region of the hyperbolic tangent function. Figures 4.29 through 4.32 display the

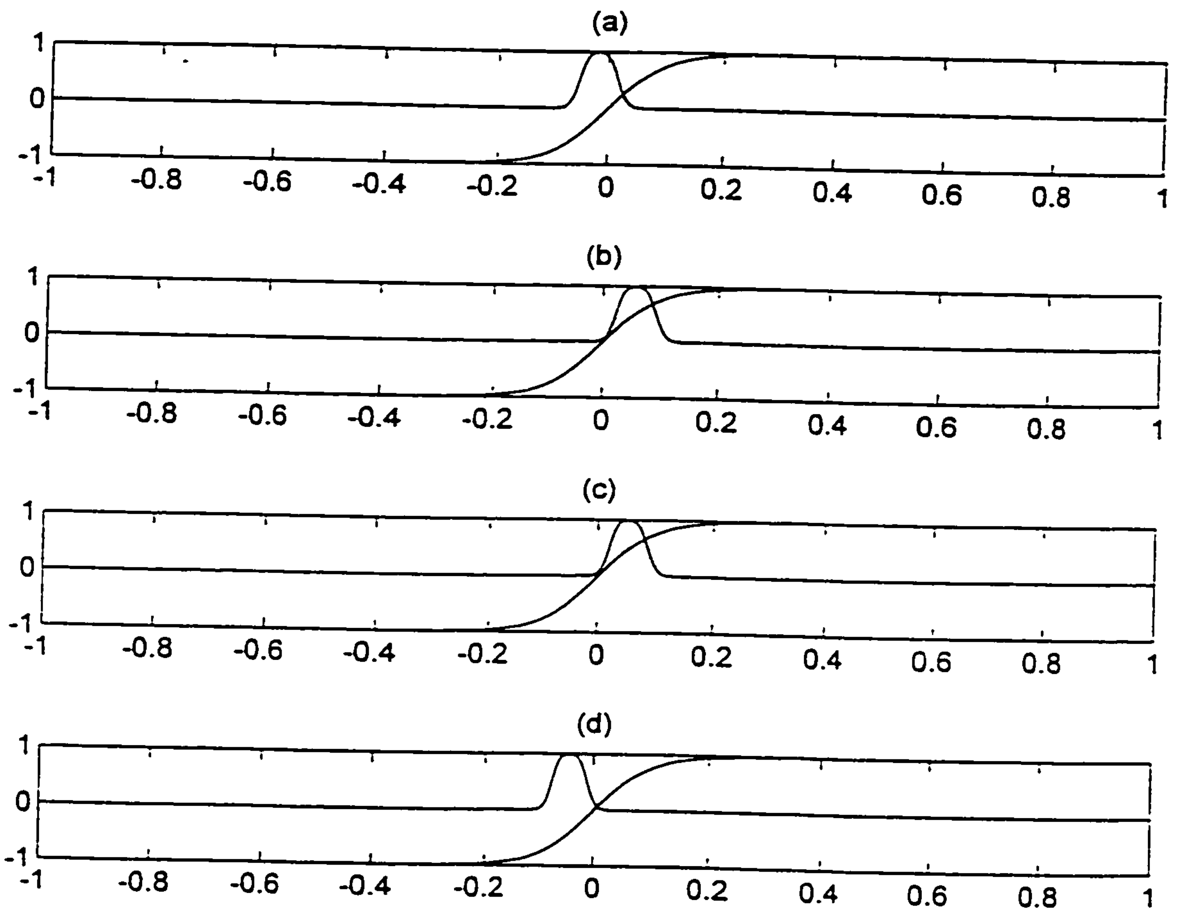


Figure 4.13 MinNNet: The normalized pdf's for Hidden Neurons (a) #1, (b) #2, (c) #3, and (d) #4 After the application of Hyperbolic Tangent Transfer Function

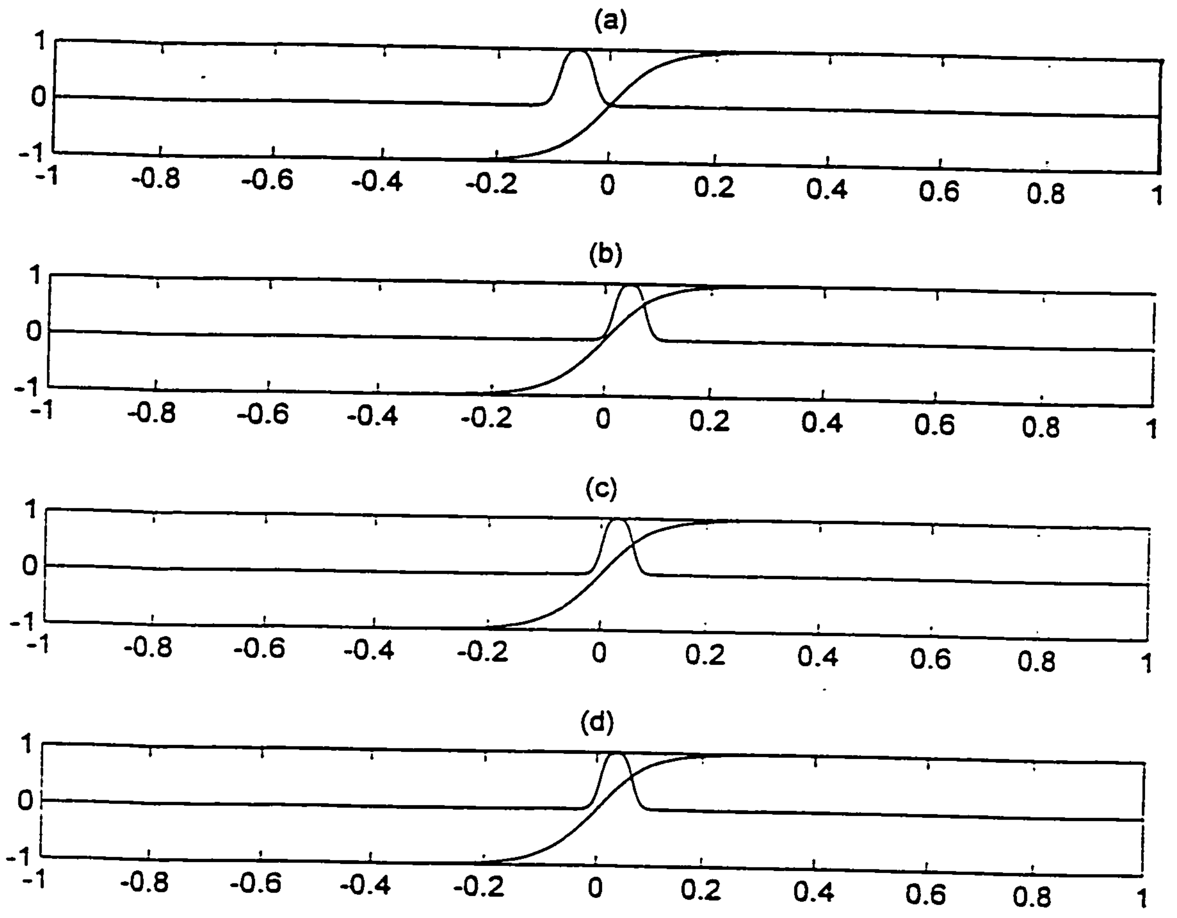


Figure 4.14 MinNNet: The normalized pdf's for Hidden Neurons (a) #5, (b) #6, (c) #7, and (d) #8 After the application of Hyperbolic Tangent Transfer Function

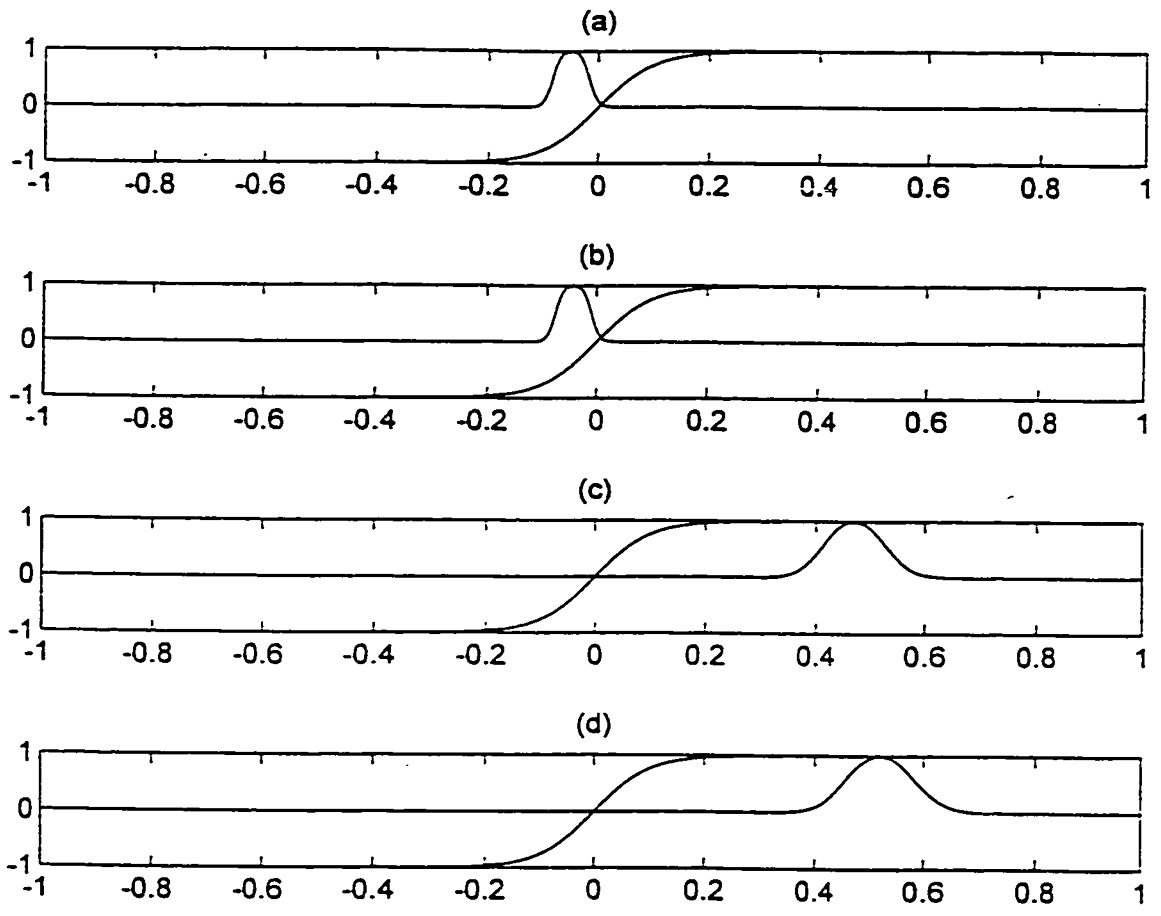


Figure 4.15 MinNNet: The normalized pdf's for Hidden Neurons (a) #9, (b) #10, (c) #11, and (d) #12 After the application of Hyperbolic Tangent Transfer Function

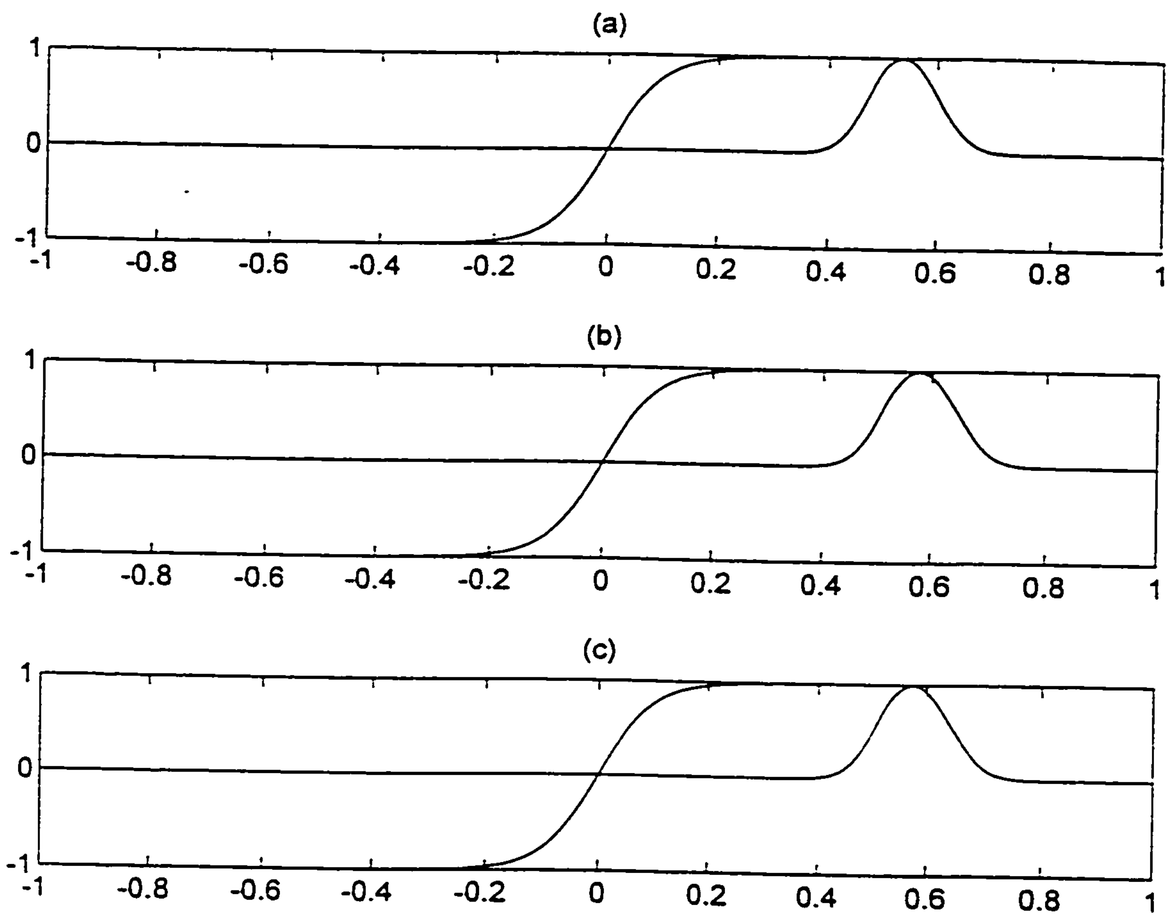


Figure 4.16 MinNNet: The normalized pdf's for Hidden Neurons (a) #13, (b) #14, and (c) #15 After the application of Hyperbolic Tangent Transfer Function

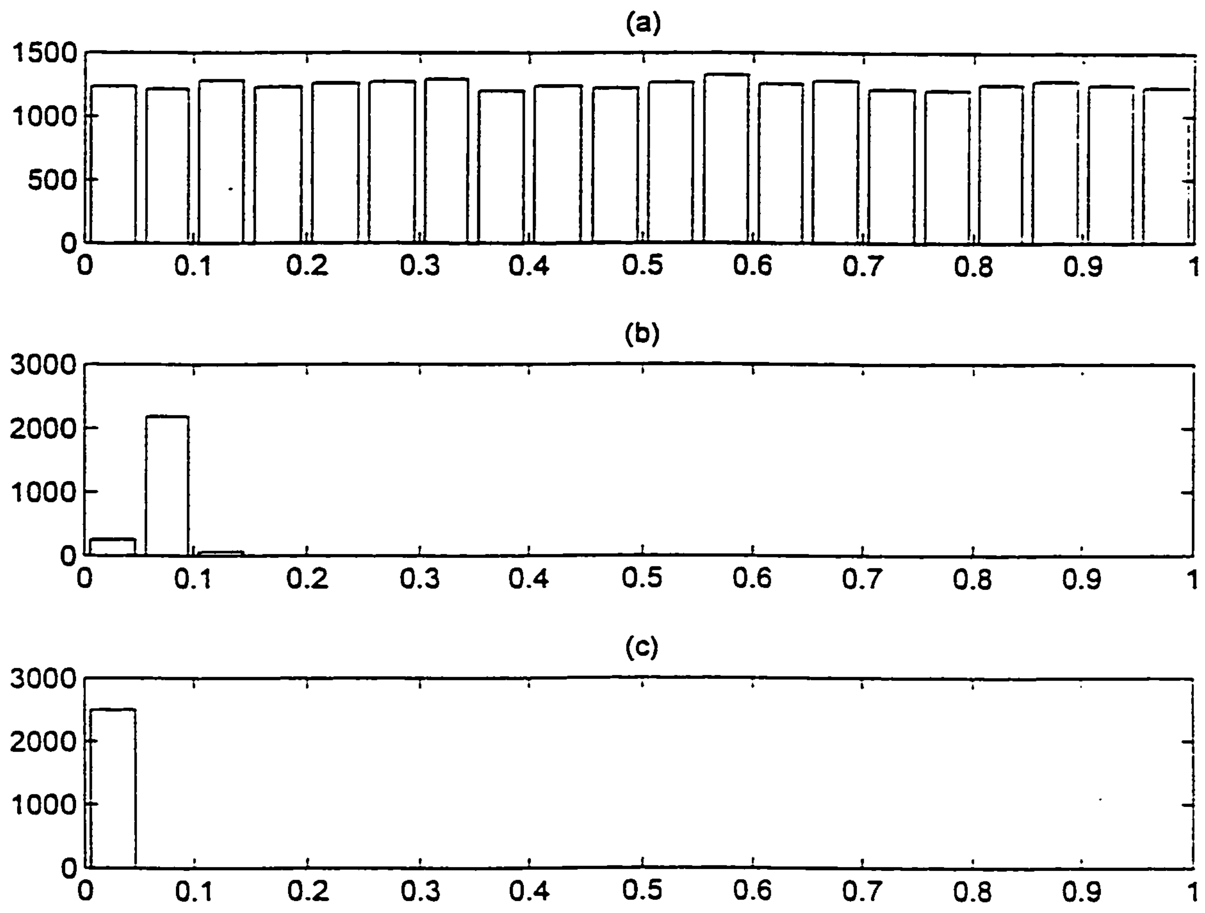


Figure 4.17 MinNNet: Histograms for (a) Input data, (b) MinNNet, and (c) Minimum

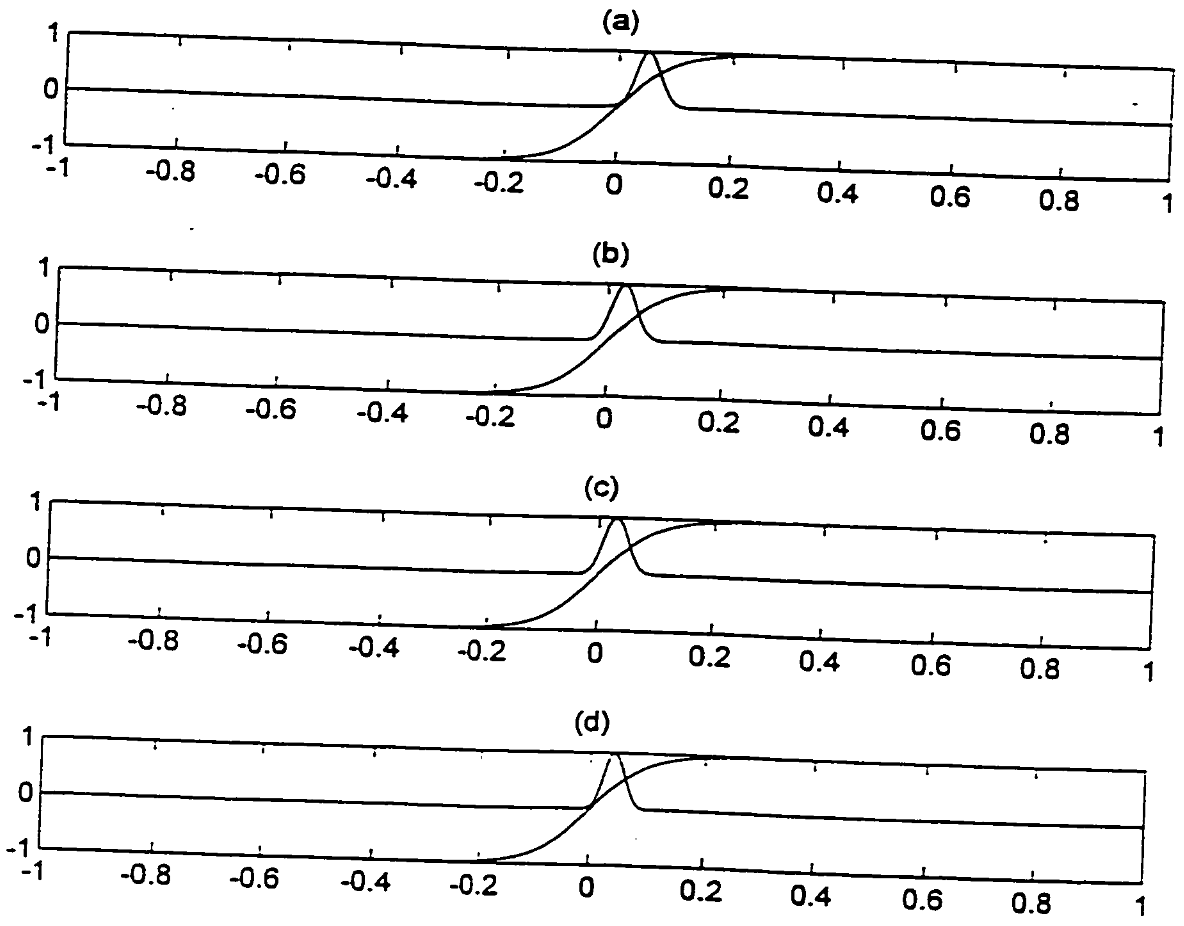


Figure 4.18 MedNNet: The normalized pdf's for Hidden Neurons (a) #1, (b) #2, (c) #3, and (d) #4 Before the application of Hyperbolic Tangent Transfer Function

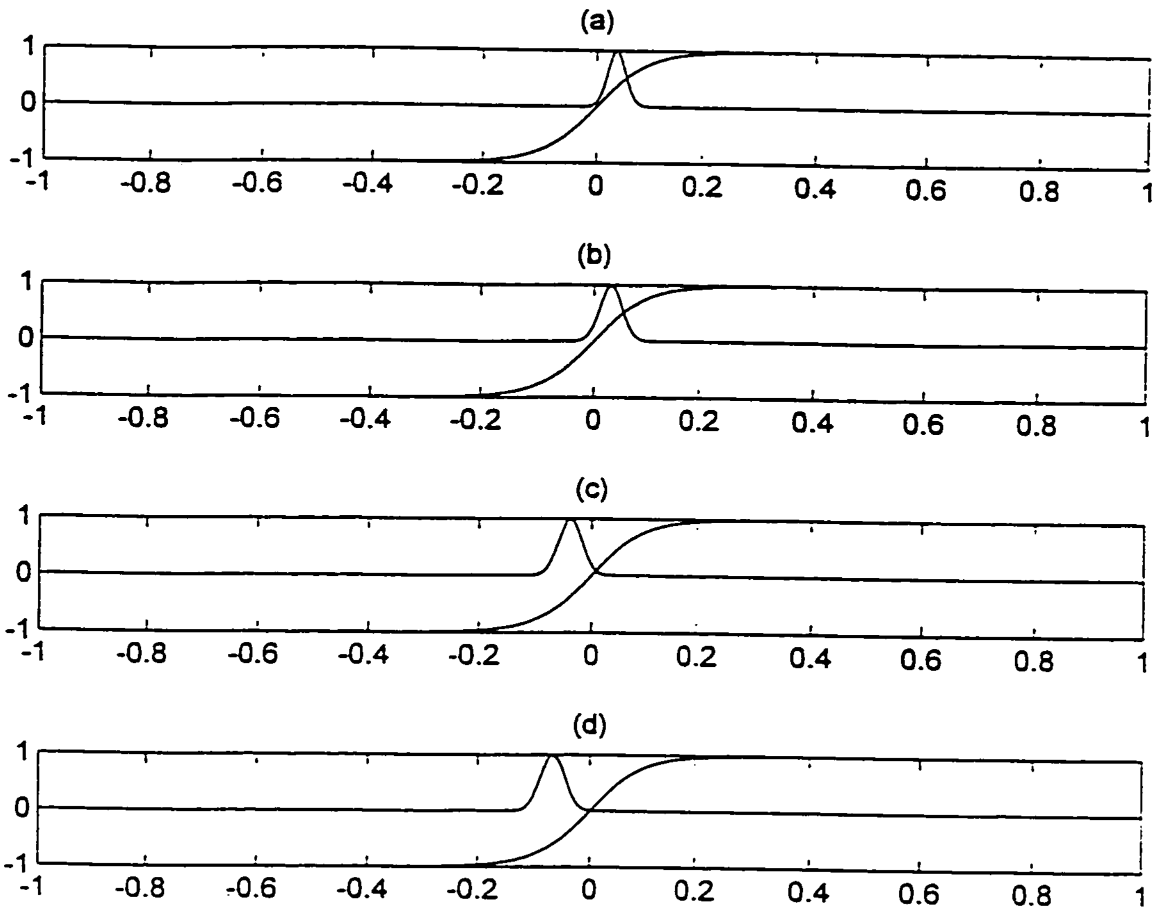


Figure 4.19 MedNNet: The normalized pdf's for Hidden Neurons (a) #5, (b) #6, (c) #7, and (d) #8 Before the application of Hyperbolic Tangent Transfer Function

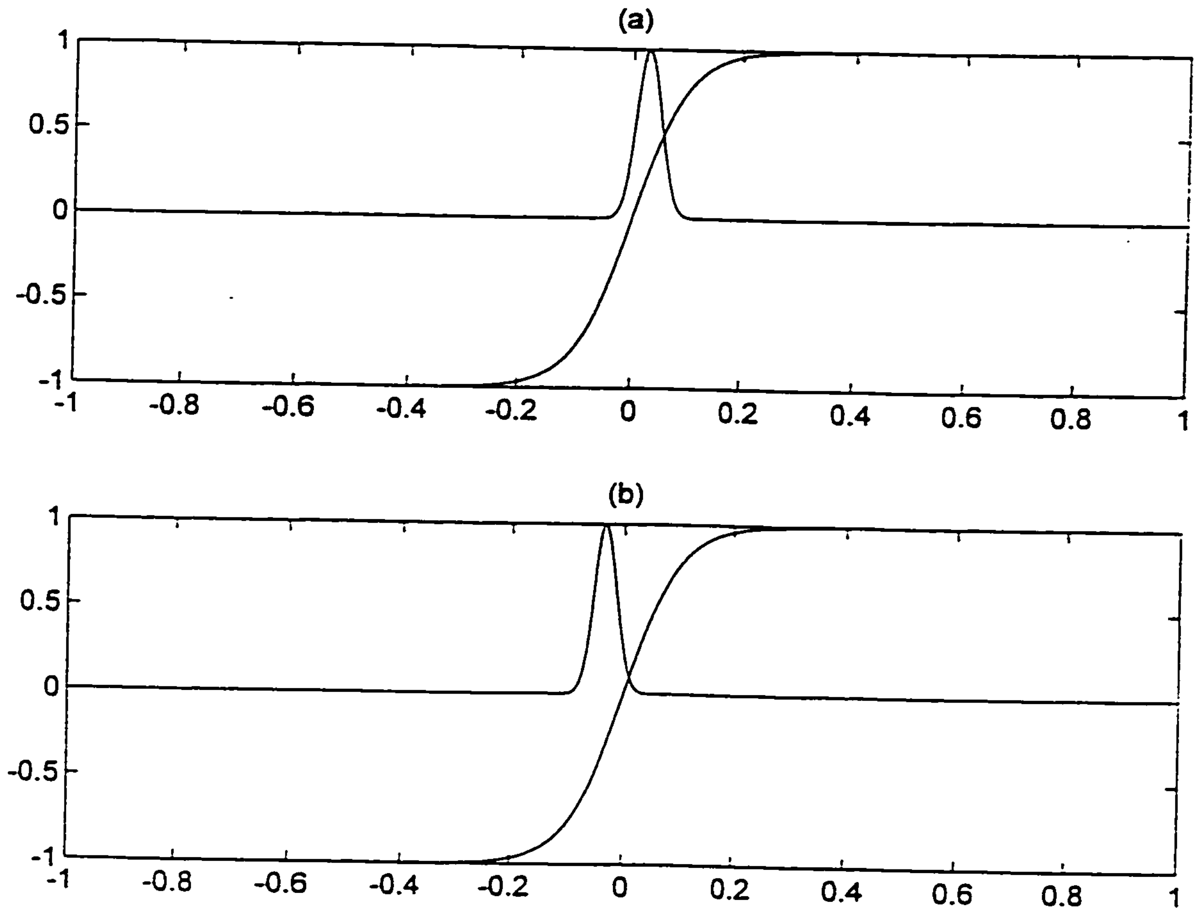


Figure 4.20 MedNNet: The normalized pdf's for Hidden Neurons (a) #9, and (b) #10 Before the application of Hyperbolic Tangent Transfer Function

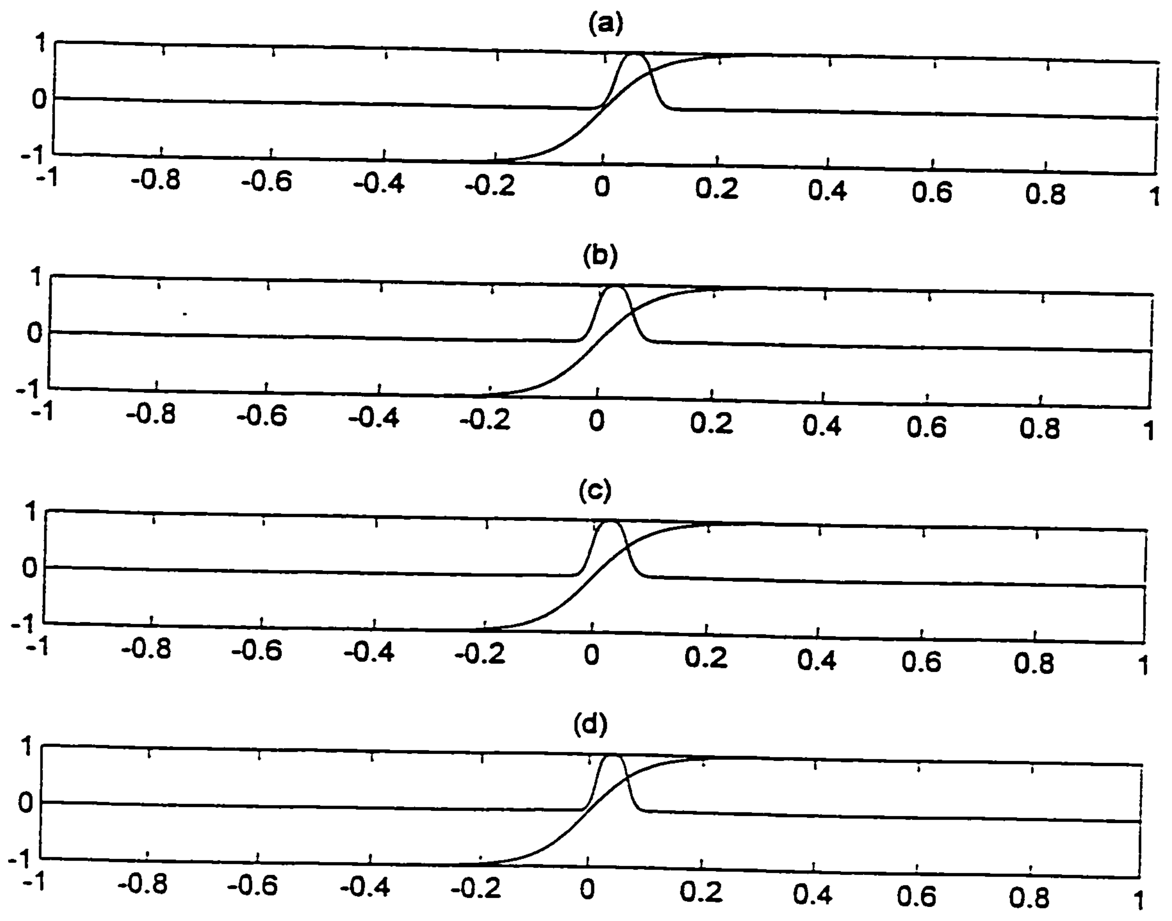


Figure 4.21 MedNNet: The normalized pdf's for Hidden Neurons (a) #1, (b) #2, (c) #3, and (d) #4 After the application of Hyperbolic Tangent Transfer Function

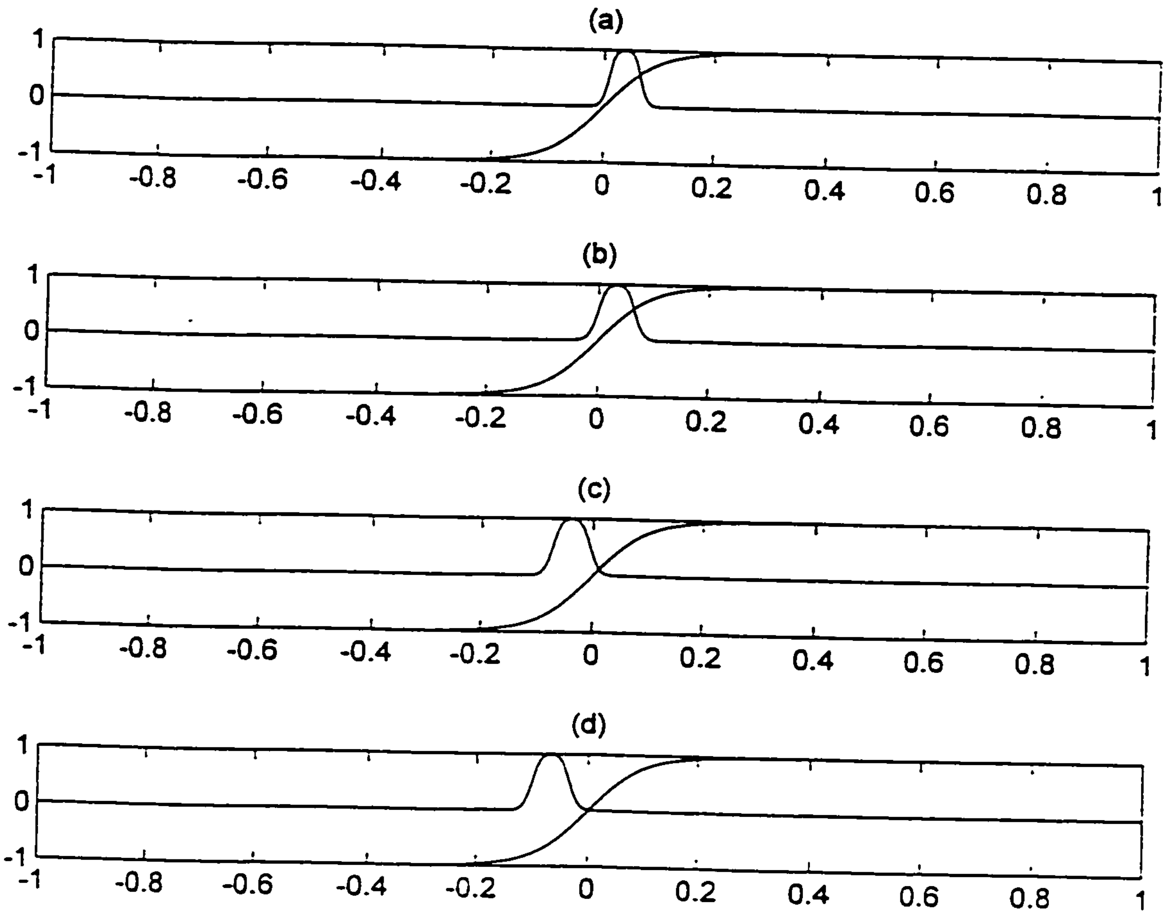


Figure 4.22 MedNNet: The normalized pdf's for Hidden Neurons (a) #5, (b) #6, (c) #7, and (d) #8 After the application of Hyperbolic Tangent Transfer Function

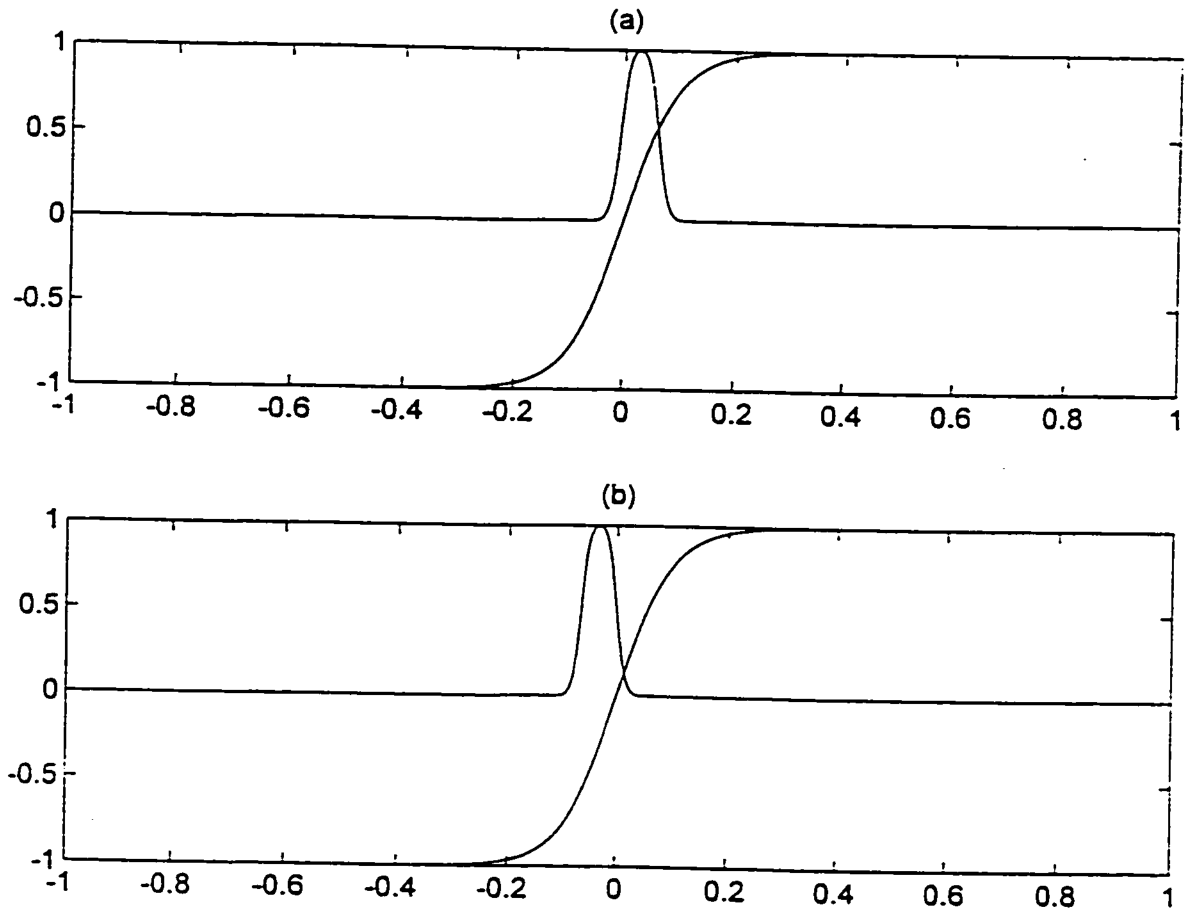


Figure 4.23 MedNNet: The normalized pdf's for Hidden Neurons (a) #9, and (b) #10 After the application of Hyperbolic Tangent Transfer Function

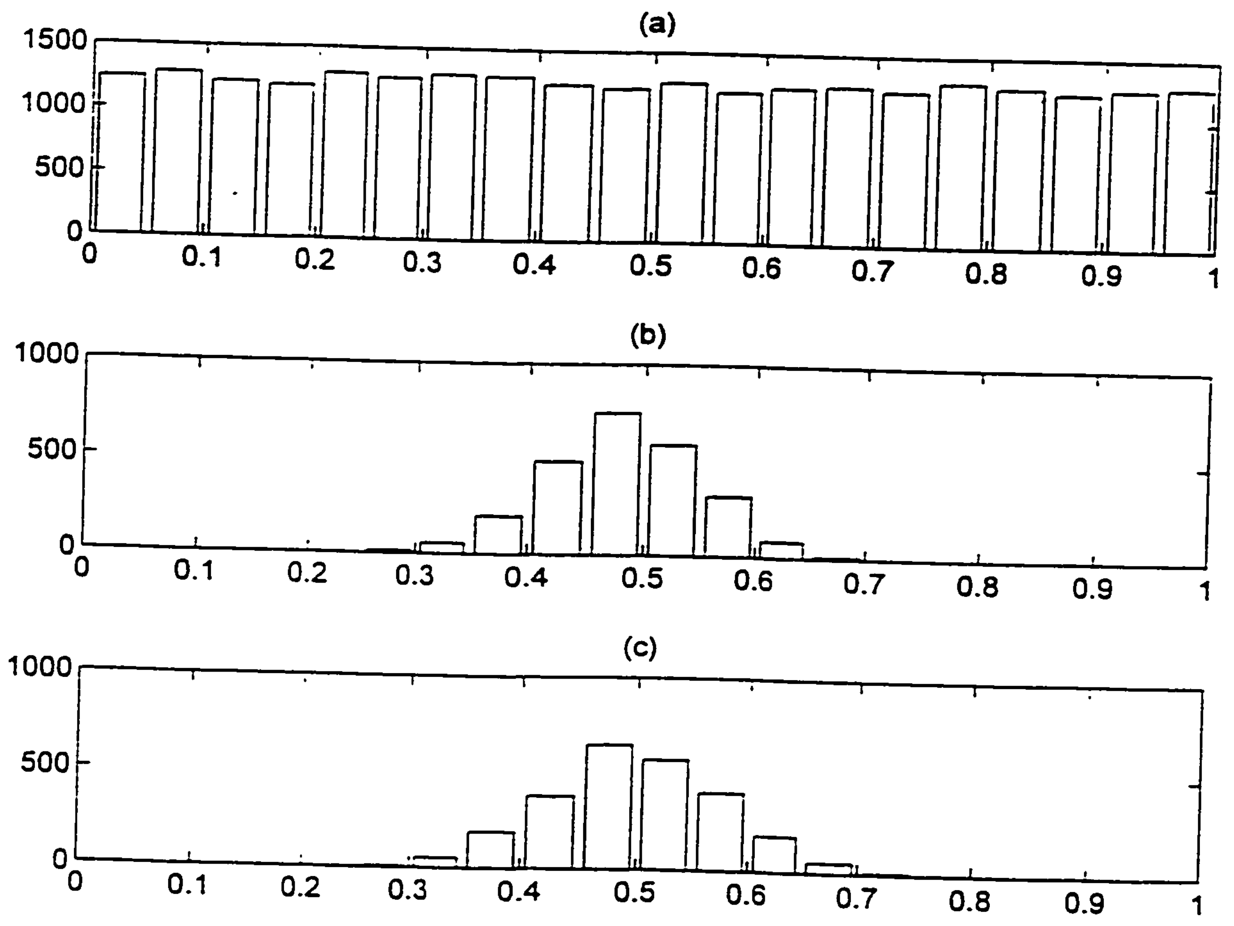


Figure 4.24 MedNNet: Histograms for (a) Input data, (b) MedNNet, and (c) Median Rank

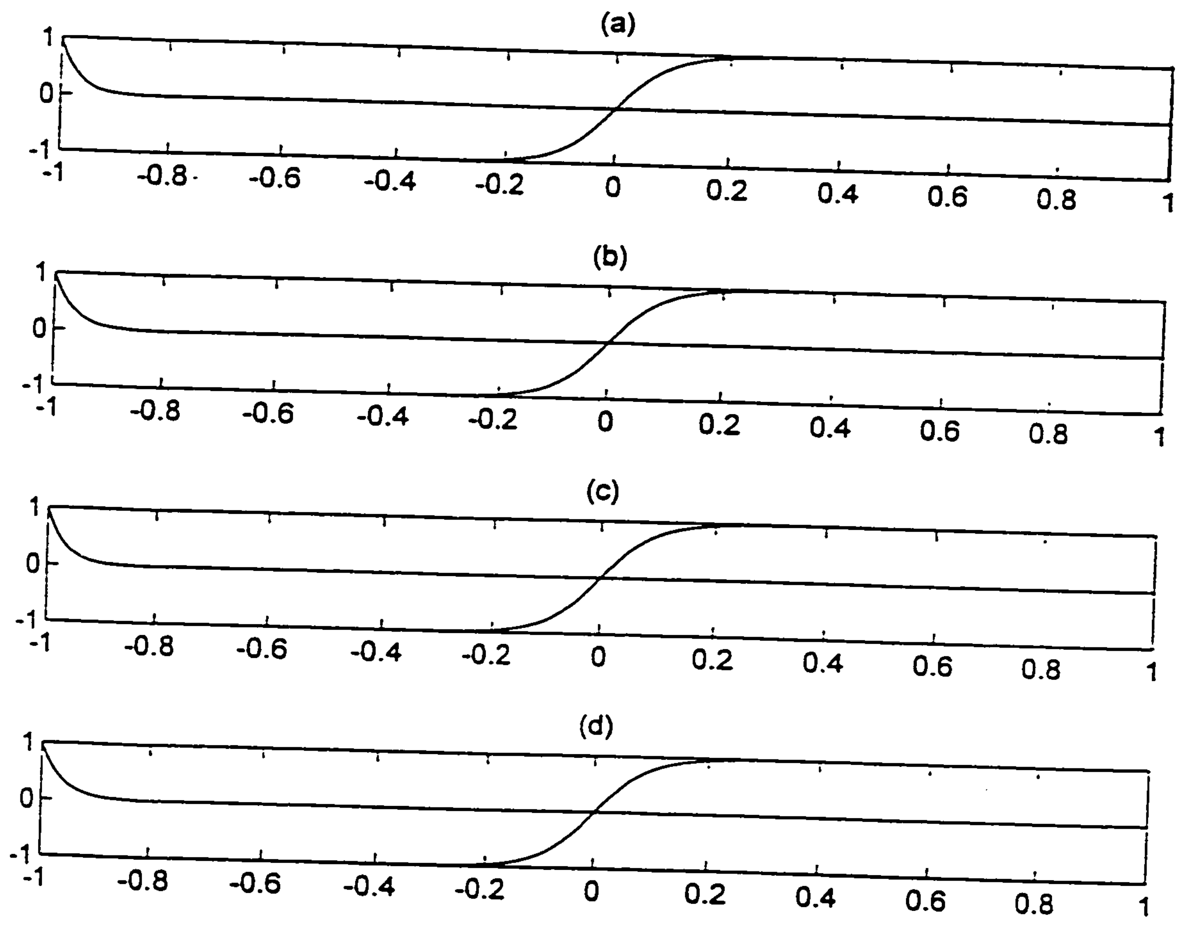


Figure 4.25 MaxNNet: The normalized pdf's for Hidden Neurons (a) #1, (b) #2, (c) #3, and (d) #4 Before the application of Hyperbolic Tangent Transfer Function

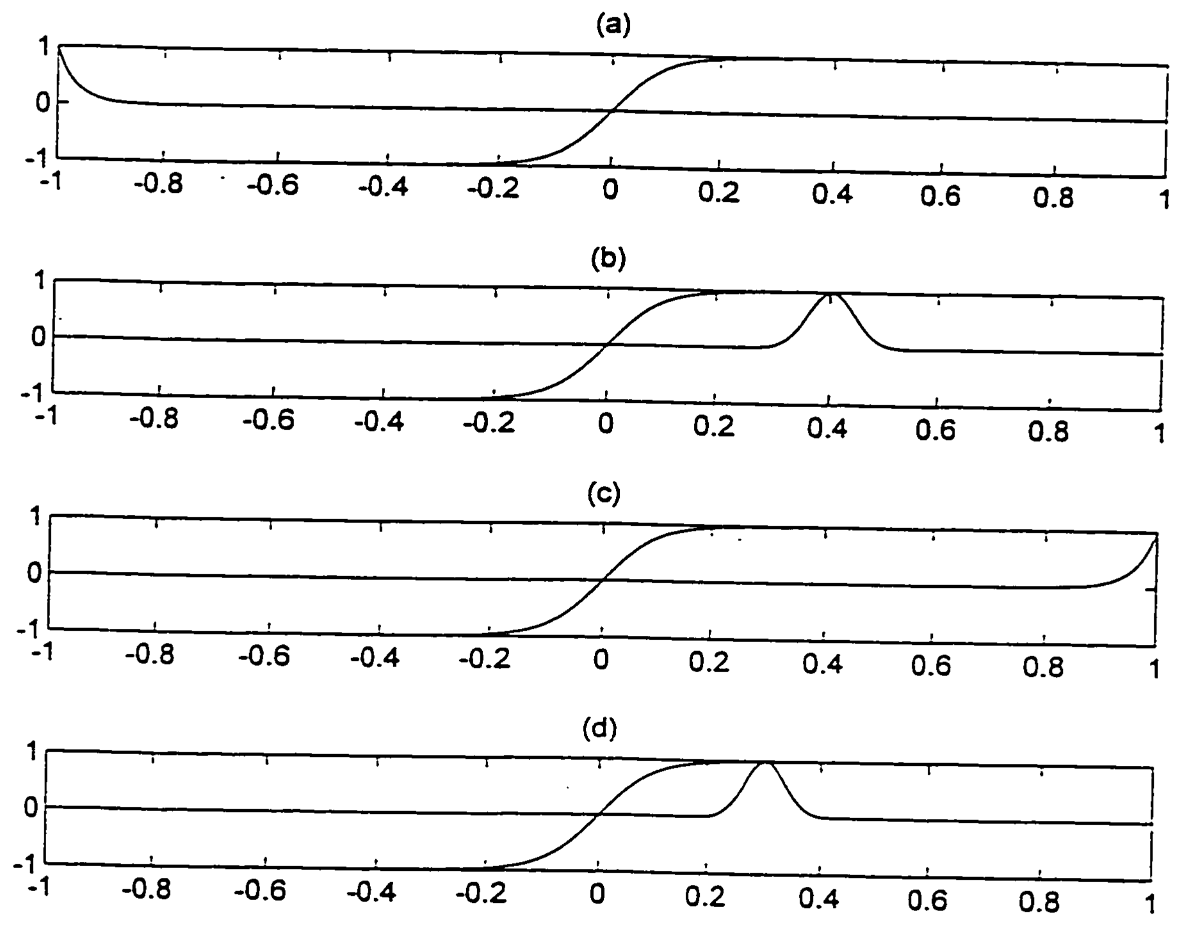


Figure 4.26 MaxNNet: The normalized pdf's for Hidden Neurons (a) #5, (b) #6, (c) #7, and (d) #8 Before the application of Hyperbolic Tangent Transfer Function

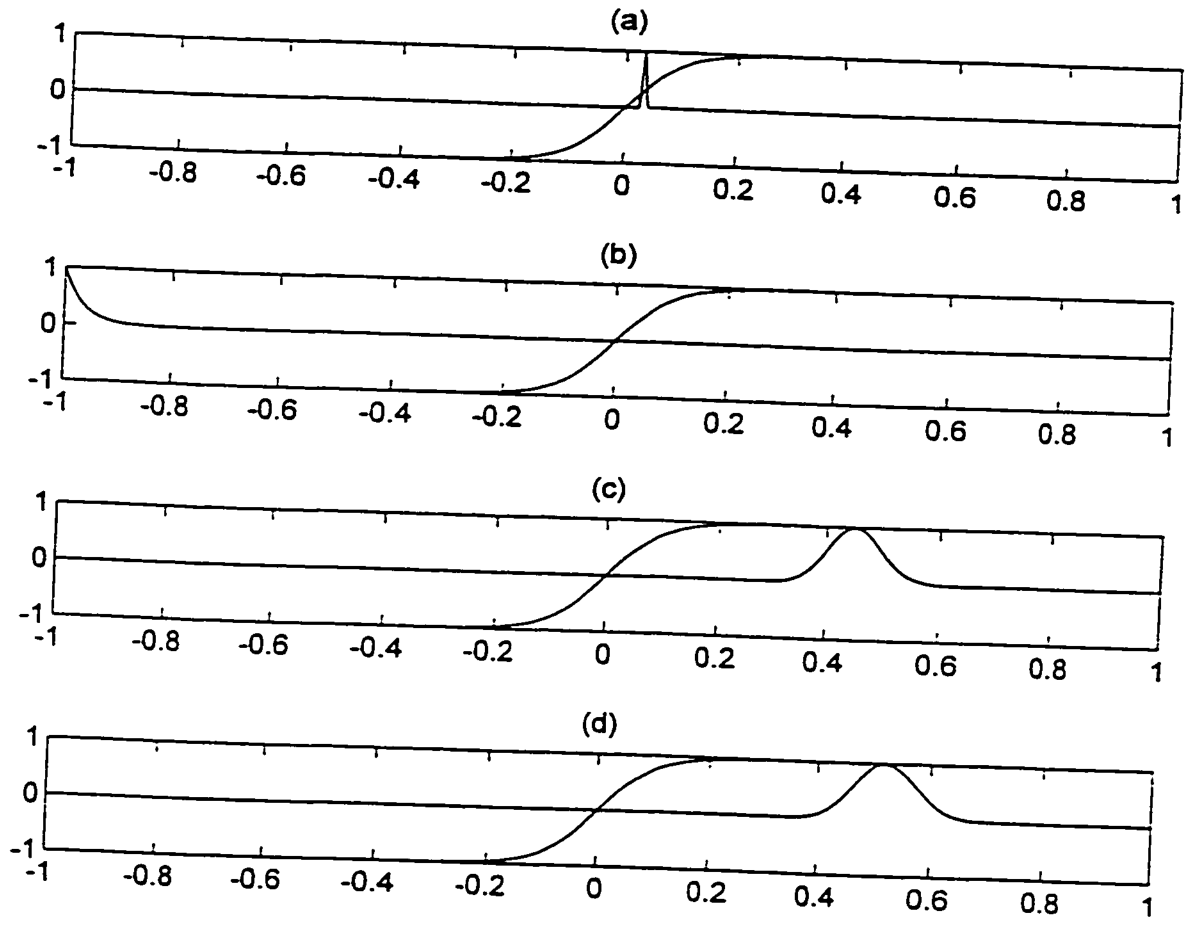


Figure 4.27 MaxNNet: The normalized pdf's for Hidden Neurons (a) #9, (b) #10, (c) #11, and (d) #12 Before the application of Hyperbolic Tangent Transfer Function

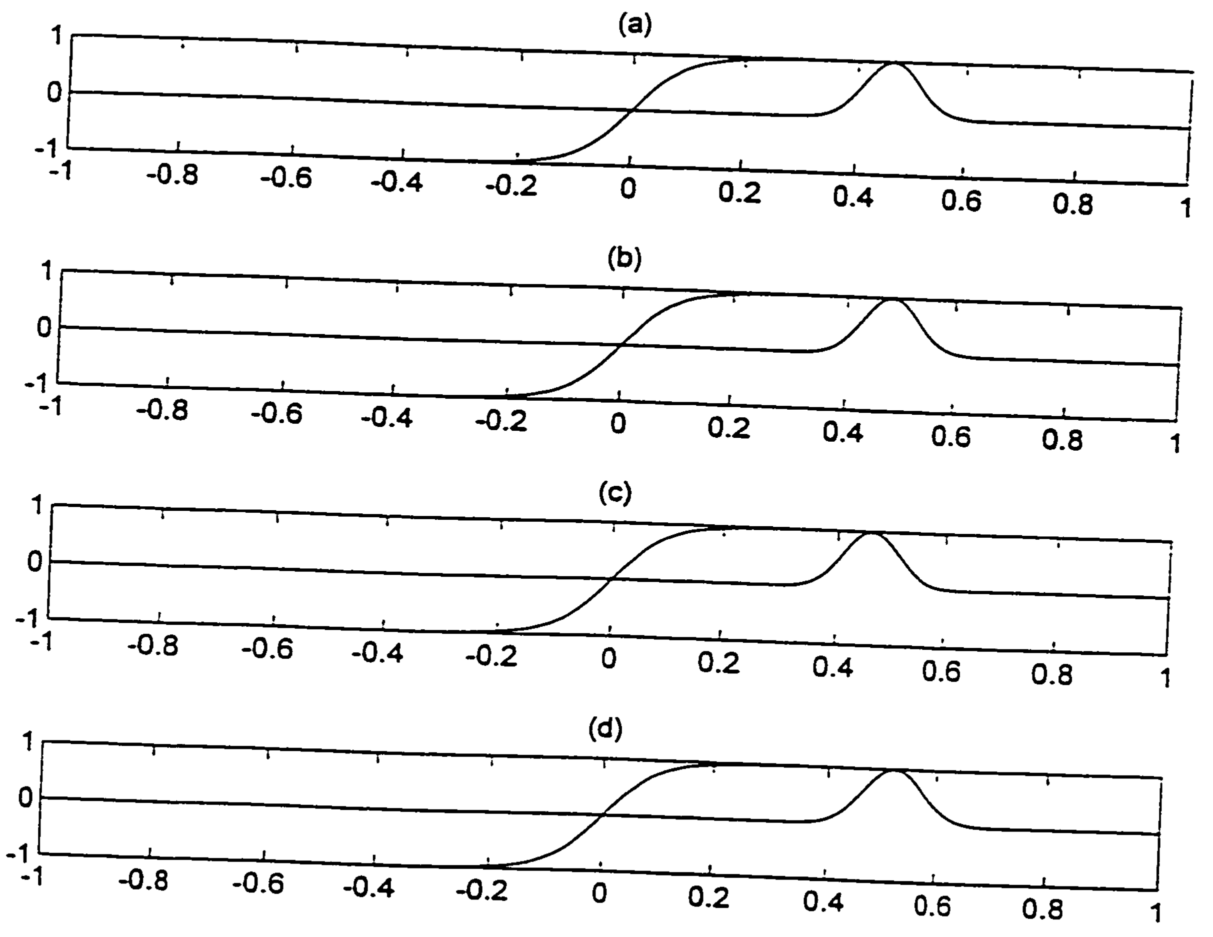


Figure 4.28 MaxNNet: The normalized pdf's for Hidden Neurons (a) #13, (b) #14, (c) #15, and (d) #16 Before the application of Hyperbolic Tangent Transfer Function

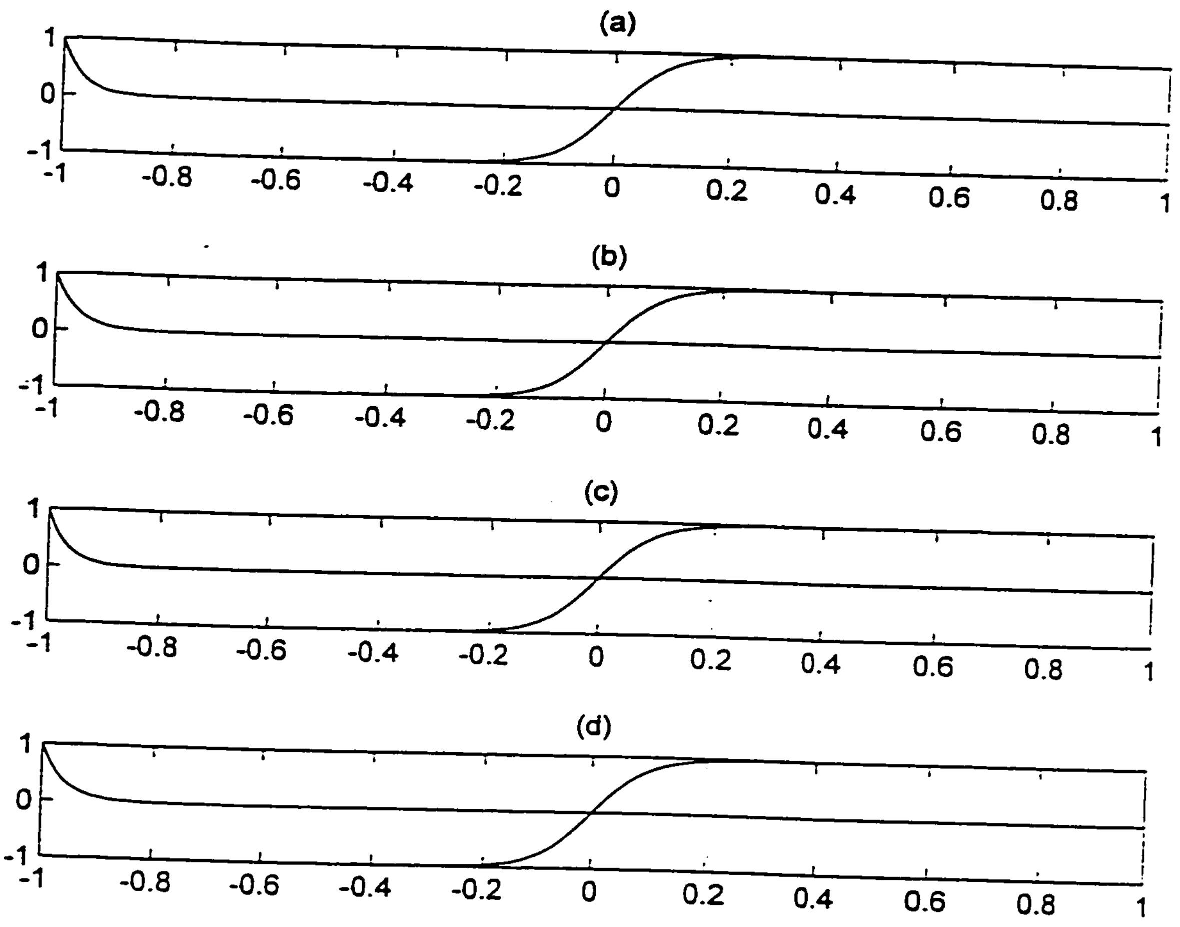


Figure 4.29 MaxNNet: The normalized pdf's for Hidden Neurons (a) #1, (b) #2, (c) #3, and (d) #4 After the application of Hyperbolic Tangent Transfer Function

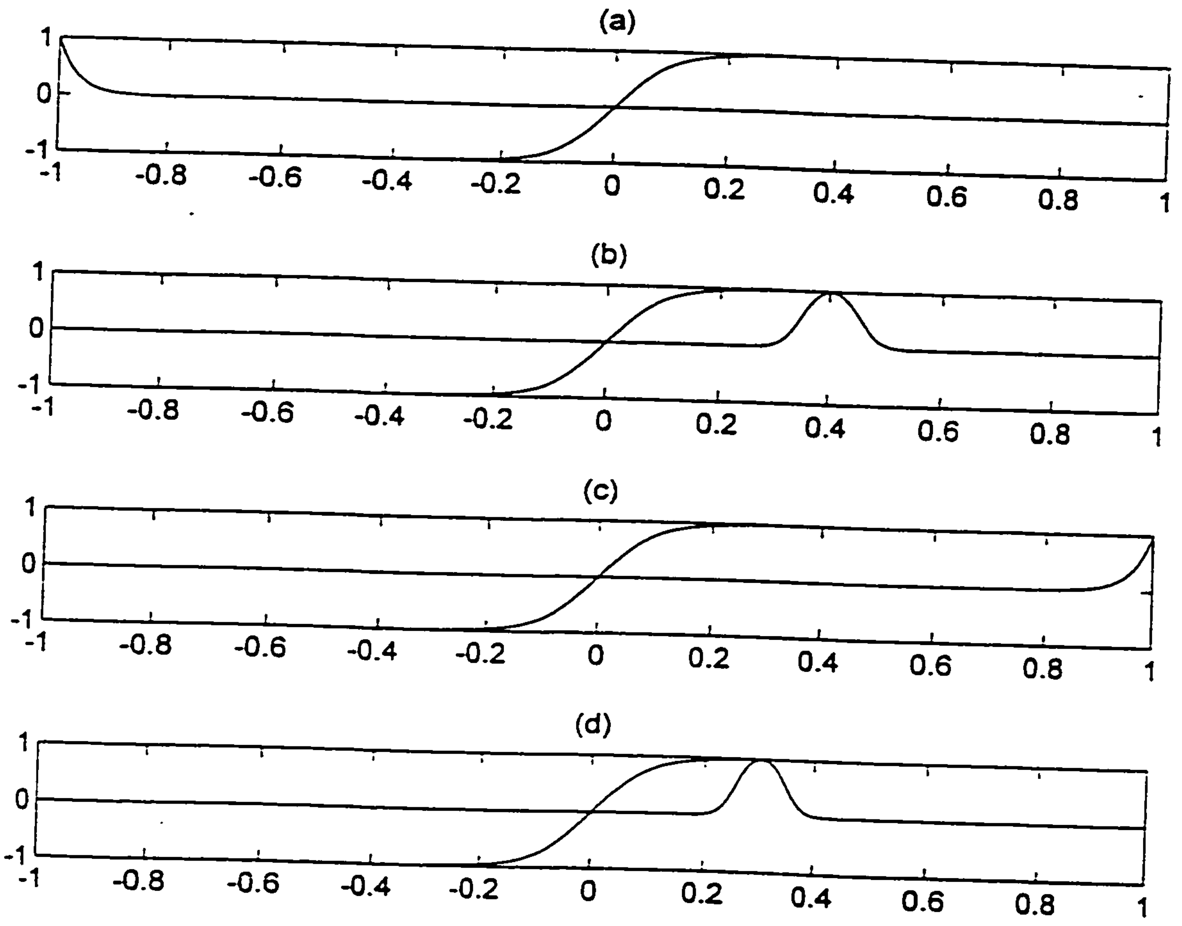


Figure 4.30 MaxNNet: The normalized pdf's for Hidden Neurons (a) #5, (b) #6, (c) #7, and (d) #8 After the application of Hyperbolic Tangent Transfer Function

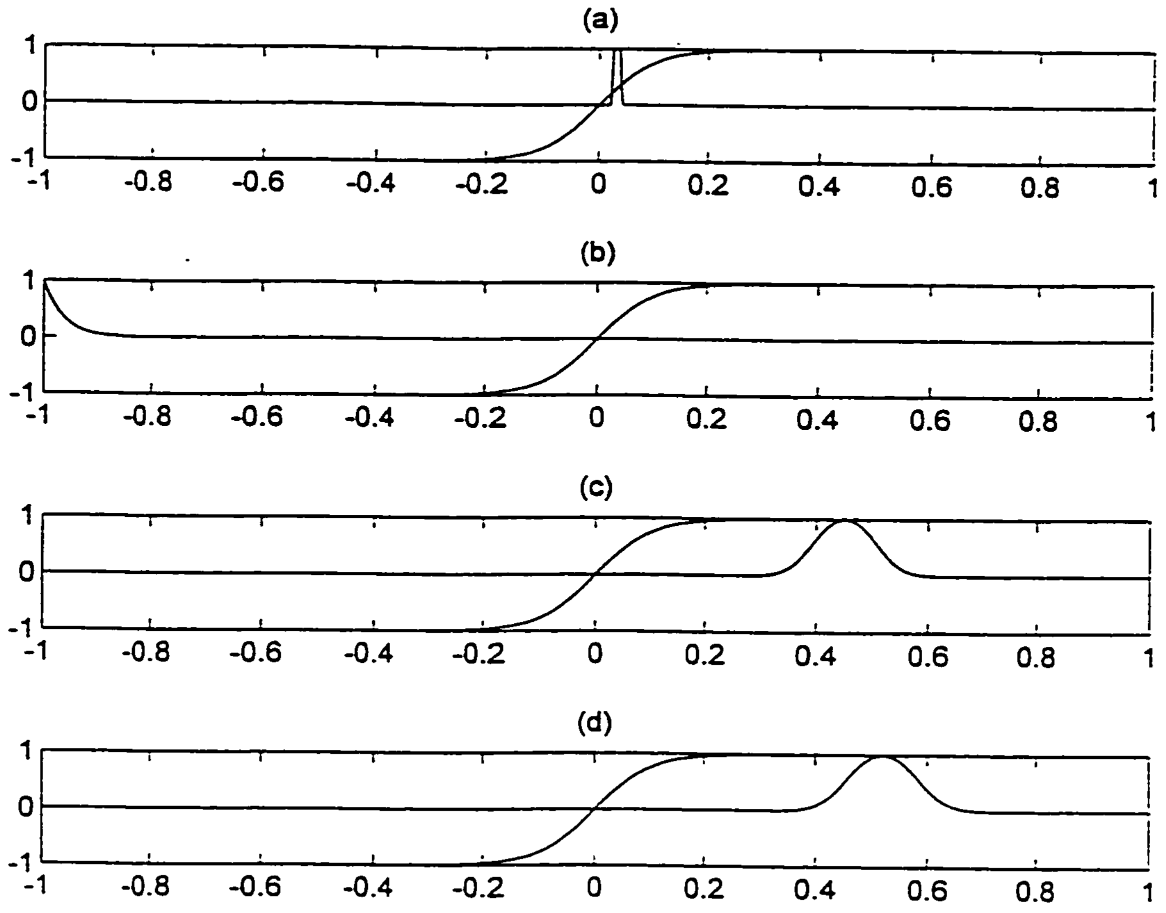


Figure 4.31 MaxNNet: The normalized pdf's for Hidden Neurons (a) #9, (b) #10, (c) #11, and (d) #12 After the application of Hyperbolic Tangent Transfer Function

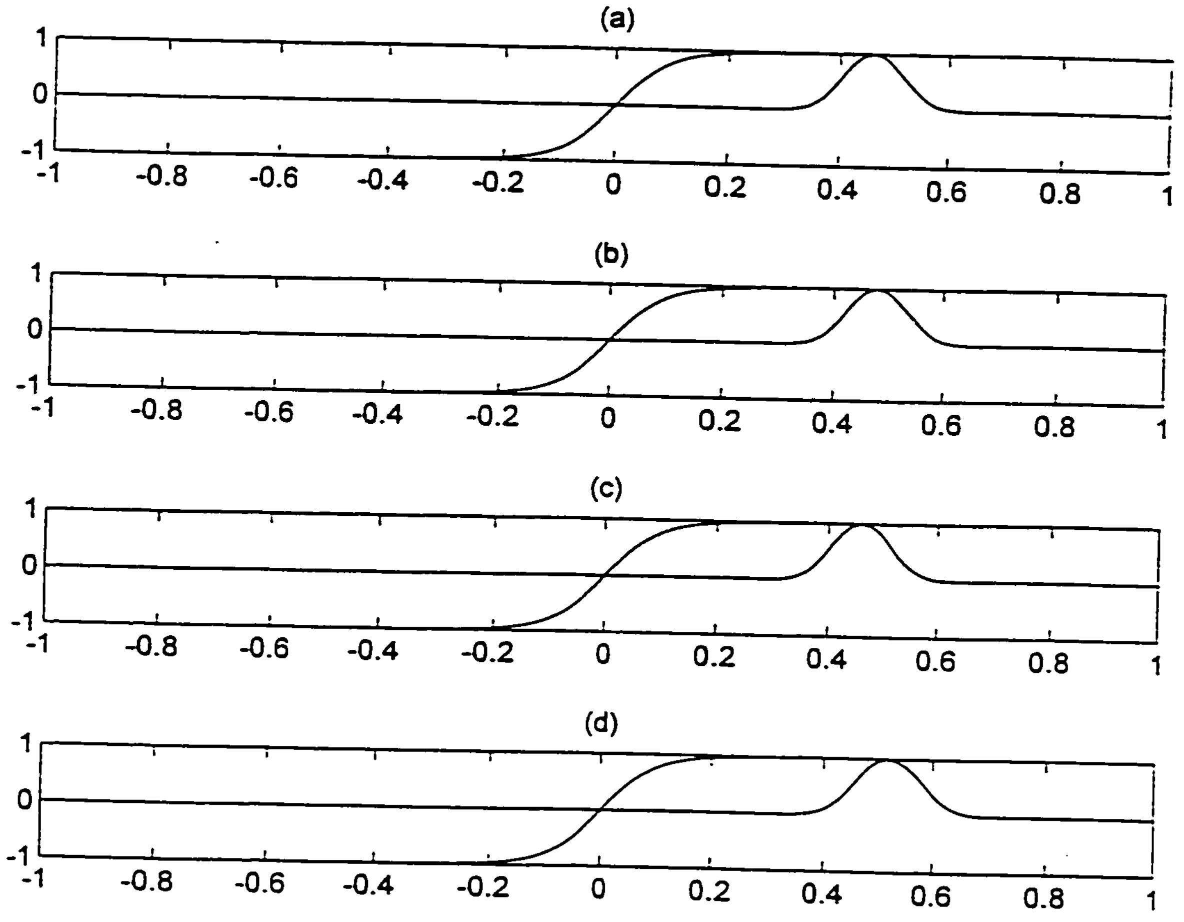


Figure 4.32 MaxNNet: The normalized pdf's for Hidden neurons (a) #13, (b) #14, (c) #15, and (d) #16 After the application of Hyperbolic Tangent Transfer Function

normalized pdf's of hidden neurons after the hyperbolic tangent transfer function. The function (see Figure 4.25) are populated into the maximum region after they are multiplied with the output layer weights and added together such a way that the estimate of the maximum rank of the input is sufficiently close enough to the desired maximum rank. Figure 4.33 depicts the histograms of the input, the actual MaxNNet output and the true maximum rank. By looking at Figure 4.33, one can see that the pdf of the actual MaxNNet is fairly close to that of the desired maximum rank.

Testing results are shown in Figure 4.34. In Figure 4.34, '+' sign shows the estimates for MaxNNet, 'x' sign displays the estimates for MedNNet, and 'o' sign depicts the estimates for MinNNet. Moreover, straight lines indicate the desired values. Random numbers in Figure 4.34a were generated between [0.8 1] and [0.1 0.3] for parts (a) and (b) respectively. The SSE of testing random numbers between [0.8 1] are 0.0038, 0.0042, and 0.0144 for MaxNNet, MedNNet, and MinNNet respectively. The MaxNNet does a better sorting than those of the other two neural networks. Since the input to the MaxNNet is close to 1 which moves the outputs of the hidden neurons to 1 and results in better sorting. On the other hand, the MedNNet still does as good as the MaxNNet does since MedNNet is taking the average of its inputs. The MinNNet is not as good as MedNNet, since the input to the MinNNet moves the outputs of the hidden neurons away from the theoretical minimum of the hyperbolic tangent transfer function. But the result of MinNNet is still promising. Furthermore, the SSE of testing random numbers between [0.1 0.3] are 0.0218, 0.0084, and 0.0066 for MaxNNet, MedNNet, and MinNNet respectively. In this part of the testing MinNNet does the best sorting.

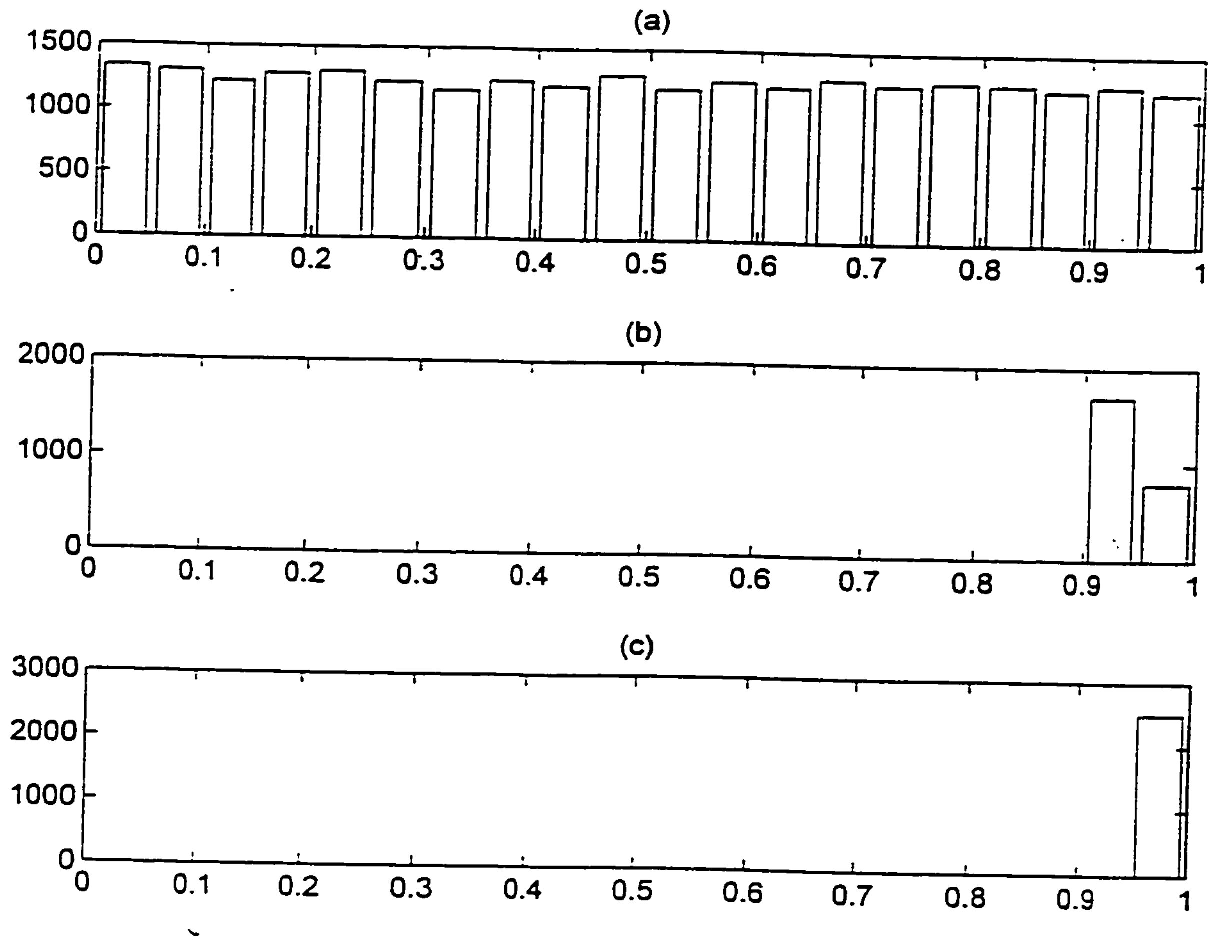


Figure 4.33 MaxNNet: Histograms for (a) Input data, (b) MaxNNet, and (c) Maximum

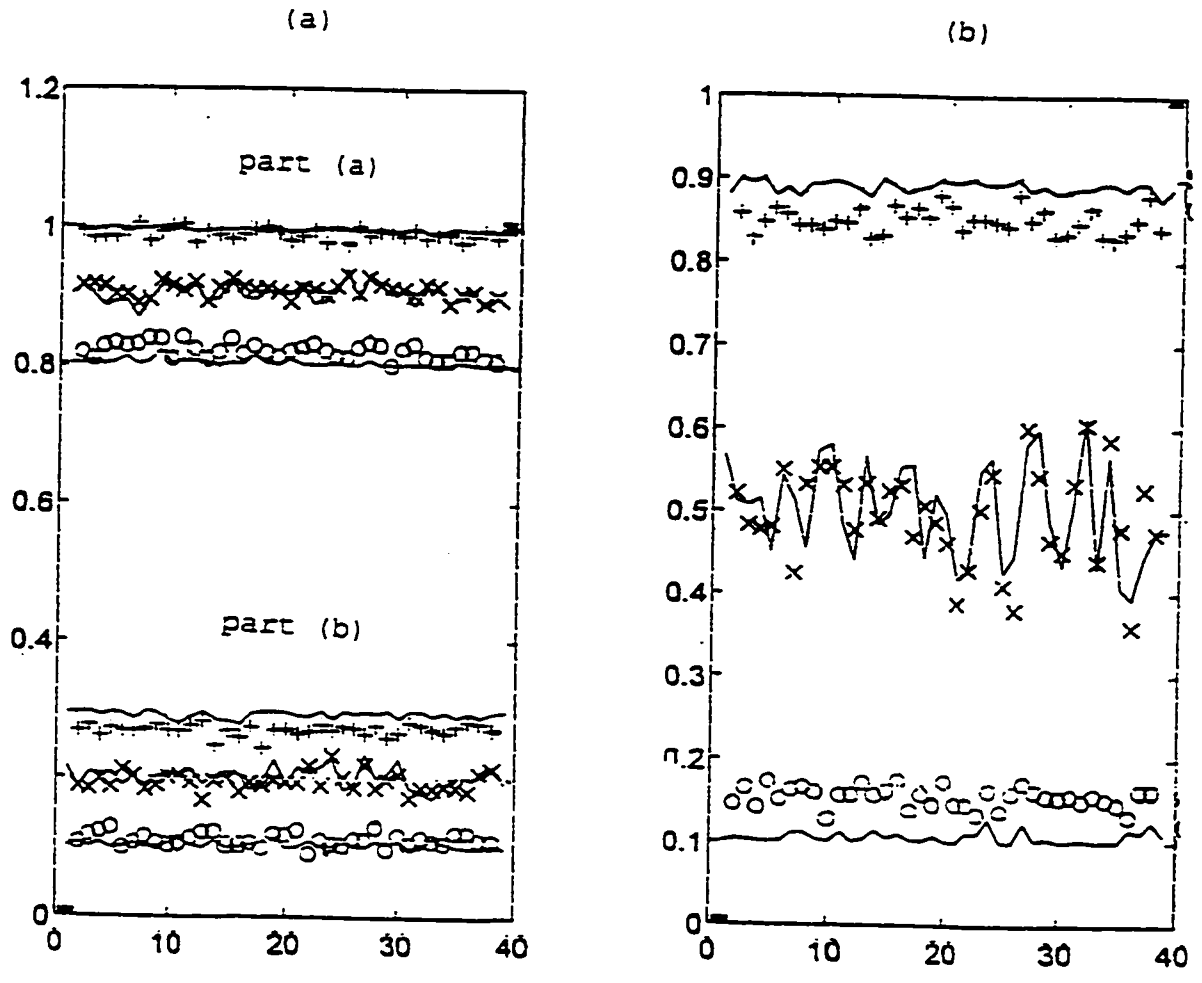


Figure 4.34 (a) Testing Results of MaxNNet, MedNNet and MinNNet for random numbers that were generated between, part(a) [0.8 1], part (b) [0.1 0.3], (b) Testing results of MaxNNet, MedNNet, and MinNNet for random numbers that were generated between [0.1 0.9]

Because the input to this neural network hold the outputs of the hidden neurons in the minimum region of the hyperbolic tangent function. On the other hand, the averaging process in the MedNNet still gets a better SSE than that of MaxNNet. Likewise, the MaxNNet's result is also as good as that of the other two neural network filters. In Figure 4.34b, the random numbers were generated between 0.1 and 0.9. The SSE's are 0.0739, 0.0665, and 0.0960 for MaxNNet, MedNNet and MinNNet respectively. In this case, MedNNet does a better testing than that of other two neural networks. However, the results from MinNNet and MaxNNet are still promising in the sense that they can be used effectively in finding the good estimates of the minimum and the maximum ranks of their input.

4.6 Neural Network Filters for Detection

Applying split-spectrum processing in ultrasonic signals combined with order statistic filters improves the signal-to-noise ratio of backscattered signals [63, 64, 82]. The performance of the order statistic filters is gotten better where signal and noise have good statistical separation representing a particular rank, such as minimum, median, or maximum. Our objective in this section is to replace the conventional order statistic filters with the neural network order statistic filters. Using neural network filters provides faster responses and eases the hardware implementation of these filters using VLSI technology [29-31]. The neural network structure that is given in Figure 4.5 can be developed as a hardware implementation of any member of the order statistic filter. Specific type of neural network filters can be created by simply creating the comprehensive input training matrix and the output target vector which consists of the desired ranks of each column of its training matrix. The neural network filter is called minimum detector when the rank is minimum, median detector when the rank is median, and the maximum detector when the rank is maximum.

The block diagram of split-spectrum processing is shown in Figure 4.35. The received broad-band signal is partitioned in several narrow-band channels as shown in Figure 4.35. The output of these channels are normalized at the end of each block referred as band-pass filters and sent to an order statistic filter. We normalized the output data between 0 and 1 because of the fact that the neural network filters were trained on the numbers which were randomly selected between 0 and 1. There are three important issues in split-spectrum processing. These are the number of partitions (how many band-pass filters are used), correlation amongst partitions, and statistical information in each partition. There is an upper limit on the number of partitions that can be chosen without a large amount of overlap amongst the partitions. The correlation between partitions can simply be reduced by reducing the size of the frequency band of each channel. This might also reduce the recovery of backscattered echoes. Correlation is not as critical to the performance of the split-spectrum processing as selecting the frequency range which contains the information of grain echoes, yet this knowledge is not generally known a priori.

MinNNet, MedNNet, and MaxNNet are replaced the conventional order statistic filters. The total number of bandpass filters was chosen as forty. Because of the fact that the neural network filters that were designed in this chapter have 40 input neurons. The total number of input neurons in a neural network filter can be selected as any number. They are tested using experimental data to show how a neural network filter can utilize the statistical information that is buried in different frequency bands to improve the signal-to-noise ratio in noisy environments.

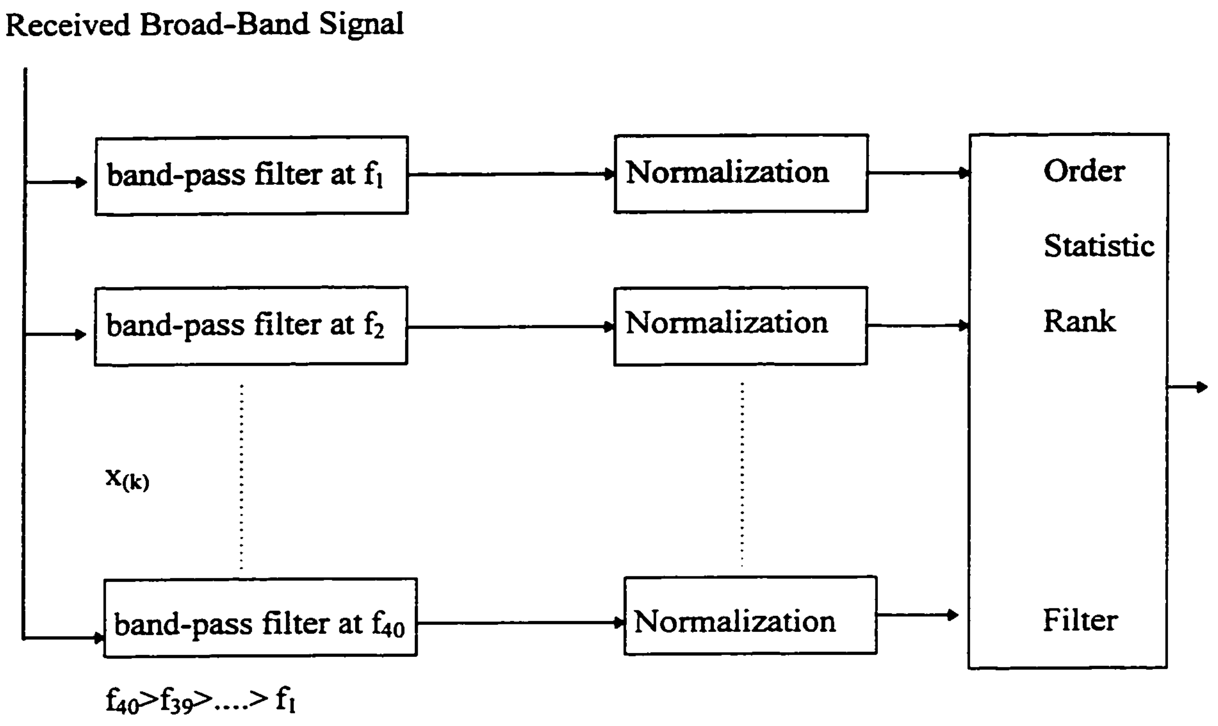

Received Broad-Band Signal
band-pass filter at f_1
band-pass filter at f_2
band-pass filter at f_{40}
$x_{(k)}$
$f_{40}>f_{39}>....>f_1$
Normalization
Normalization
Normalization
Order Statistic Rank Filter

Figure 4.35 The Block Diagram of SSP

Testing results are shown in Figure 4.36. As stated earlier, forty bandpass filters are used in the SSP. In Figure 4.36, the frequencies of the channels reside within the frequency range of 2.875-11.625 MHz and the bandwidth of the channels is 2 MHz. In addition, target echo is at location of 430 (SNR is 0 dB). In can be observed that the

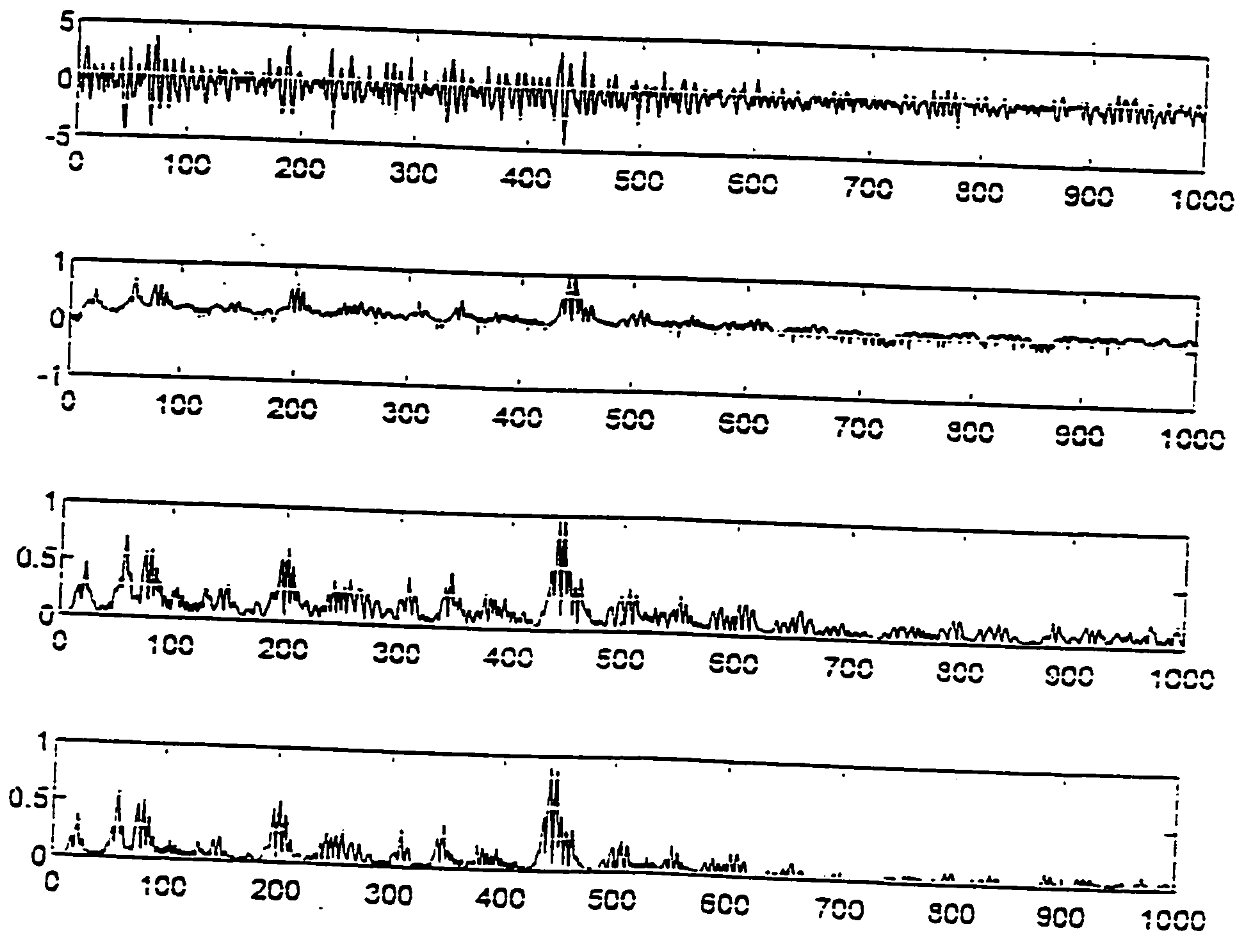


Figure 4.36 Frequency Range of Bandpass Filters is 2.875-11.625 MHz with the Bandwidth of 2 MHz. Target echo is at location 430 (a) Noisy Signal with a Single Echo, (b) The Output of MaxNNet, (c) The Output of MedNNet, (d) The Output of MinNNet

MedNNet and the MinNNet do a better target detection than that of MaxNNet because the training of MaxNNet causes the MaxNNet to have similar weights at the end of its training phase (see Figure 4.8). On the contrary, when we look at the output of MaxNNet, the target echo is still detected successfully.

4.7 Conclusion

It has been shown how to implement order statistic filters using neural networks. Examples of neural network filters are given. These filters are MinNNet, MedNNet, and MaxNNet. Testing results show that the MedNNet does a better sorting than that of MinNNet and MaxNNet. Furthermore, we examined the Hinton diagrams and the probability density functions of each hidden neuron. Analytical results reveal that the hidden neurons in MedNNet active around the middle region of their hyperbolic tangent functions where the learning takes place. As a result, MedNNet is faster in training and offers desirable testing results in sorting. On the other hand, MinNNet, and MaxNNet active either at the beginning or at the end of their hyperbolic tangent functions where there is less learning. This results in poor performance in both training and testing phases of these neural network filters.

The training matrix for each type of filter is prepared using random numbers between 0 and 1. Each column of this training matrix consists of numbers from certain subsets of [0,1]. The total number of training vectors in each subset and the values for the boundaries of these subsets are chosen randomly. The training output vector consists of the minimum, or the median, or the maximum rank of each columns. The performance is not 100% accurate. However, experimental results show that utilizing neural network filters in split-spectrum processing achieves a desirable sorting in noisy environments.

In summary, the neural network filters have been presented to replace the conventional order statistic filters in the field of signal and image processing. These neural network filters have the following properties: 1) the response is real-time once they are trained, 2) all hidden neurons have the same structure which makes its hardware implementation easy using VLSI design techniques. Based on experimental observations, neural network order statistic filters can be efficiently used in split-spectrum processing in order to detect flaw echoes in grain scattering noise.

CHAPTER V

NEURAL NETWORKS FOR ULTRASONIC GRAIN SIZE DISCRIMINATION

In this study, the grain power spectrum neural network (GPSNN) has been developed to classify the ultrasonic backscattered grain signals for material characterization [53, 61, 63, 64, 66, 82, 83, 86]. The GPSNN has 32 input nodes, 13 hidden neurons determined adaptively, and one summing output node. A set of 4490 training sets is used to train the neural network. A new set of 12572 testing sequences is utilized to test the GPSNN performance. The samples tested for grain size discrimination are steel with grain sizes of 14 and 50 microns. GPSNN achieves a recognition performance of over 98%. This high level of recognition suggests that the GPSNN is a promising method for ultrasonic nondestructive testing.

5.1 Introduction

The importance of evaluating the microstructure of materials ultrasonically has been long recognized [53, 62]. In particular, it is high interest to estimate grain size or classify materials based on the scattering properties of their microstructure. Backscattered grain echoes are random signals that bear information related to both the grain size and frequency of sound. In ultrasonic grain size characterization a model for the grain signal consists of the convolution of components representing the contribution of the measuring system impulse response (i.e., the interrogating ultrasonic wavelet) and the grain scattering function. This function contains information related to many random physical parameters such as grain size, shape, orientation, boundary characteristics, and chemical constituents. Consequently, the grain scattering signal becomes random and exhibits a great deal of variability in the time domain. Therefore, spectral analysis is often adopted as an alternate method for signal characterization [63, 64, 82, 83, 86].

In the Rayleigh scattering region (the wavelength, λ, is larger than the average grain diameter, A) the scattering coefficients vary with the third power of the grain diameter and the fourth power of the frequency, while the absorption coefficient increases linearly with frequency [53]. The model for attenuation coefficient for a given frequency, f, and at a distance, z, can be modeled as

$$\alpha(z,f) = b_a(z)f + b_s(z)A^3 f^4 \tag{5.1}$$

where $b_a(z)$ is the absorption constant and $b_s(z)$ is the scattering constant. Inspection of the Equation (5.1) suggests that the high frequency component of the interrogating ultrasonic wavelet backscatters with higher intensity than the lower frequency components. This situation results in a higher expected frequency than that of the original interrogating wavelet.

In this study, we have developed the design procedure for a neural network to discriminate the frequency signatures inherent in ultrasonic grain scattering signals. This method, called the grain power spectrum neural network (GPSNN), offers practical advantages such as real-time processing, adaptability and training capability [13, 21, 27, 45, 54, 70, 88]. GPSNN deduces the relationship between the measurement power spectrum and the classification output without knowing the scattering model, physical parameters, or the solution methodology. With the neural network, as each set of input vectors is applied to the neural network, the hidden layers configure themselves to recognize certain frequency features of the input vectors related to the scattering properties of the materials. After the GPSNN is fully trained, each hidden neuron will represent certain frequency characteristics of the total input space. Therefore, when the power spectrum of a new grain scattering signal is applied to GPSNN, each neuron is able to respond to the presence of a particular subset of frequency information which it was trained to recognize.

5.2 Design of GPSNN

To discriminate the frequency signatures inherent to grain signals, the backpropagation algorithm [21] is used to design the GPSNN. The block diagram of GPSNN is shown in Figure 5.1. The input data is normalized and segmented where it represents information pertaining to a predefined region of materials. This data is used as the input to the power spectrum processor using the Fast Fourier transform algorithm. Then, the power spectrum of segmented data is applied to a three layer fully interconnected neural network for classification. A set of desired output values (0 for grain Type-1 and 1 for grain Type-2) is then compared to the estimated outputs of the neural network for every set of input values of the power spectrum of the backscattered grain echoes. The weights are appropriately updated by backpropagating the gradient of the output error through the entire neural network. In this study, an adaptive hidden neuron algorithm is utilized in the design of the GPSNN (see Chapter 2). This adaptive hidden neuron algorithm is promising for determining the optimal number of hidden neurons.

The experimental data, used for both training and testing the GPSNN, is obtained using a broadband transducer with a 6.22 MHz center frequency and a 3 dB bandwidth of 2.75 MHz. A total of 38 experimental data sets were measured. Each experimental data set is composed of 512 points sampled at 25 MHz. This sampling frequency satisfies the Nyquist rate and reduces the correlation among the data points. Furthermore, the sparse sampling is also beneficial in reducing the neural network size.

The experimental data sets are normalized by removing the mean and dividing it with standard deviation. This normalization is highly desirable because it desensitizes the neural network to the signal offset and/or signal gain. Normalization is given in Equation (2.2) (see Chapter 2).

The block diagram of the GPSNN is shown in Figure 5.1. The input signal to the power spectrum block is created using a sliding window. The size of the sliding window is 64 samples, and the step between two successive windows is one sample. The first set is taken from the beginning of the experimental data. The second set is taken from the second sample of the experimental data, and this is repeated until the window covers the entire 512 samples of the measured signal. The power spectrum, $X_p(k\Omega)$, of each input set is calculated by

$$X_p(k\Omega) = FFT\{x(nT)\}.Conj\{FFT\{x(nT)\}\} \tag{5.2}$$

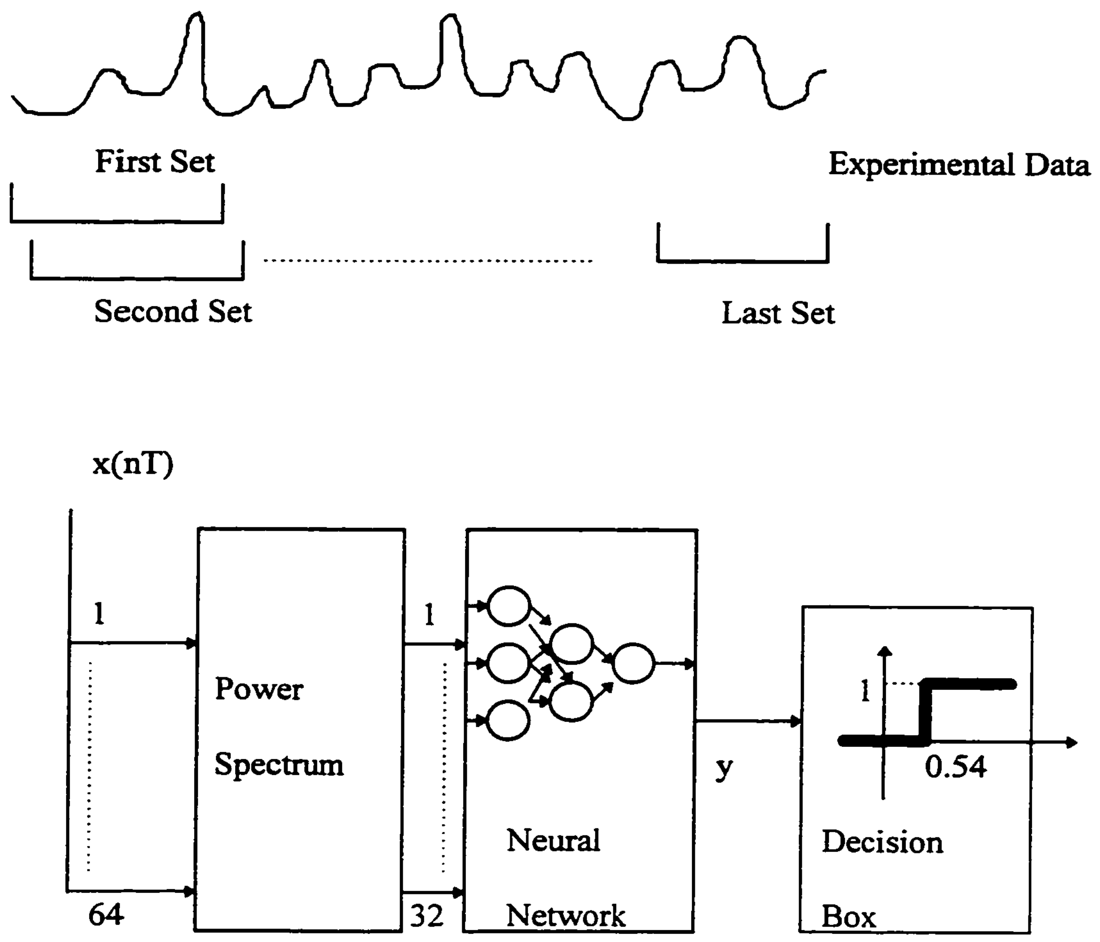


Figure 5.1 Block Diagram of the Ultrasonic Grain Power Spectrum Neural Network (GPSNN)

where *x(nT)* is the normalized sampled value of the grain echoes, *T* is the time sampling interval, FFT is the Fast Fourier transform, and Ω is the frequency sampling interval. The first 32 samples of $X_p(k\Omega)$ which span the entire grain frequency range are taken into consideration as an input to the neural network for signal classification. The output of this neural network, y, is given as

$$\begin{aligned} &y = W^{o} \tanh(W^{h} X), \\ &y \le \eta \rightarrow \text{Type - 1 Grain Signal} \\ &y > \eta \rightarrow \text{Type - 2 Grain Signal} \end{aligned} \tag{5.3}$$

where W^h , W^o are the weight matrixes for the hidden layer and the output layer respectively, and X is the input vector. Then, the output, y, is applied to the decision block in order to classify the grain size. A value of 0.54 is chosen for the threshold, η, to help decide whether the input grain signal is Type-1 or Type-2. This value is found using the output density functions (see Figure 5.2) of the training grain signals for Type-1 and Type-2. The output density functions are estimated using the Parzen method. Hence, an estimate of the density function from samples can be obtained as:

$$f_j(y) = \frac{1}{n\sigma_j} \sum_i \phi\left(\frac{y - y_i}{\sigma_j} \right), j = Type - 1 \text{ or Type - 2} \tag{5.4}$$

where $\{y_i, i = 1, 2, ..., n\}$ is the neural network output for Type-1 or Type-2, $\phi()$ is the Gaussian density function (i.e., $\phi(y) = e^{-y^2}$), and constant σ_j is chosen to be 0.25 for our experimental data ($j = 1$ for Type-1 and $j = 2$ for Type-2).

The key issue in the design of a neural network is determining the number of hidden neurons. The improper selection of hidden neurons may perform satisfactory for design data, but fails significantly for test data or causes unsatisfactory convergence (i.e., neural network is not fully trainable. To avoid these problems an adaptive hidden neuron algorithm for determining the number of hidden neurons is required [27, 65, 78, 79].

This adaptive technique starts with three hidden neurons and then adaptively increases the number of hidden neurons until convergence is guaranteed. Sum-square-error (SSE) is used as a measure of performance for the neural network. This error is used in updating the neural network weights using a backpropagation algorithm. When the SSE does not meet the appropriate criterion after 5000 epochs (an epoch is defined as one sweep through all training samples), the number of hidden neurons is increased by one (see Chapter 2).

5.3 Experimental Results

The experimental data, applied for both training and testing the GPSNN, is obtained using a broadband transducer with a 6.22 MHz center frequency and a 3 dB bandwidth of 2.75 MHz. A total of 38 experimental data sets was measured. The material applied for the microstructure is two steel blocks, type 1018, with two different grain sizes, 15 microns and 50 microns. Each experimental data set is composed of 512 points sampled at 25 MHz. This sparse sampling frequency satisfies the Nyquist rate and offers reduced correlation among the data points. Sparse sampling is also beneficial in reducing the size of the neural network an improving the overall efficiency of the grain size classification system.

From measured experimental data, a set of 4490 training sequences was assembled to train the grain power spectrum neural network. A new set of 12572 testing sequences was utilized to test the GPSNN performance. Figure 5.3 shows a sample of 4 A-scans (i.e., amplitude scan) of Type-1 and 4 A-scans of Type-2 measured grain signals.

The corresponding power spectra of these A-scans are shown in Figure 5.4. Both the A-scans and their power spectra exhibit random pattern and a set of features that can be used directly for classification is not recognizable. However, due to the scattering theory of the microstructure, the grain signal backscattered from larger grains is expected to display lower frequency content than the grain signal backscattered from smaller grains. But this trend in frequency content is not readily quantifiable due to random peaks and dips in the power spectra. Therefore, a trained neural network is conceivable to recognize the microstructure signals backscattered from materials with different grain sizes.

Table 5.1 shows the training statistics of GPSNN. The number of inputs to the neural network is 32 which represents a power spectrum of grain spanning frequency 0-12.5 MHz. It should be noted that 4490 training data sets were applied 52225 times to estimate the optimal number of hidden neurons and their weights in order to satisfy the SSE criterion. Table 5.2 presents testing results for the GPSNN. Note that 28 additional experimental grain signals (14 measurements for Type-1: gr106-gr119; 14 measurements for Type-2: gr206-gr219) are used to test the trained neural network grain classifier. Since all these signals are segmented, a total of 12572 testing sequences is used. As shown in Table 5.2, the GPSNN achieves an average recognition of 98%. This performance is impressive and statistically reliable since 17062 data segments are used in training and testing the neural network. Furthermore, this high level recognition is desirable and practical since it is applied to a short data segment which represents information pertaining to a small depth of about 6.5 mm of steel samples.

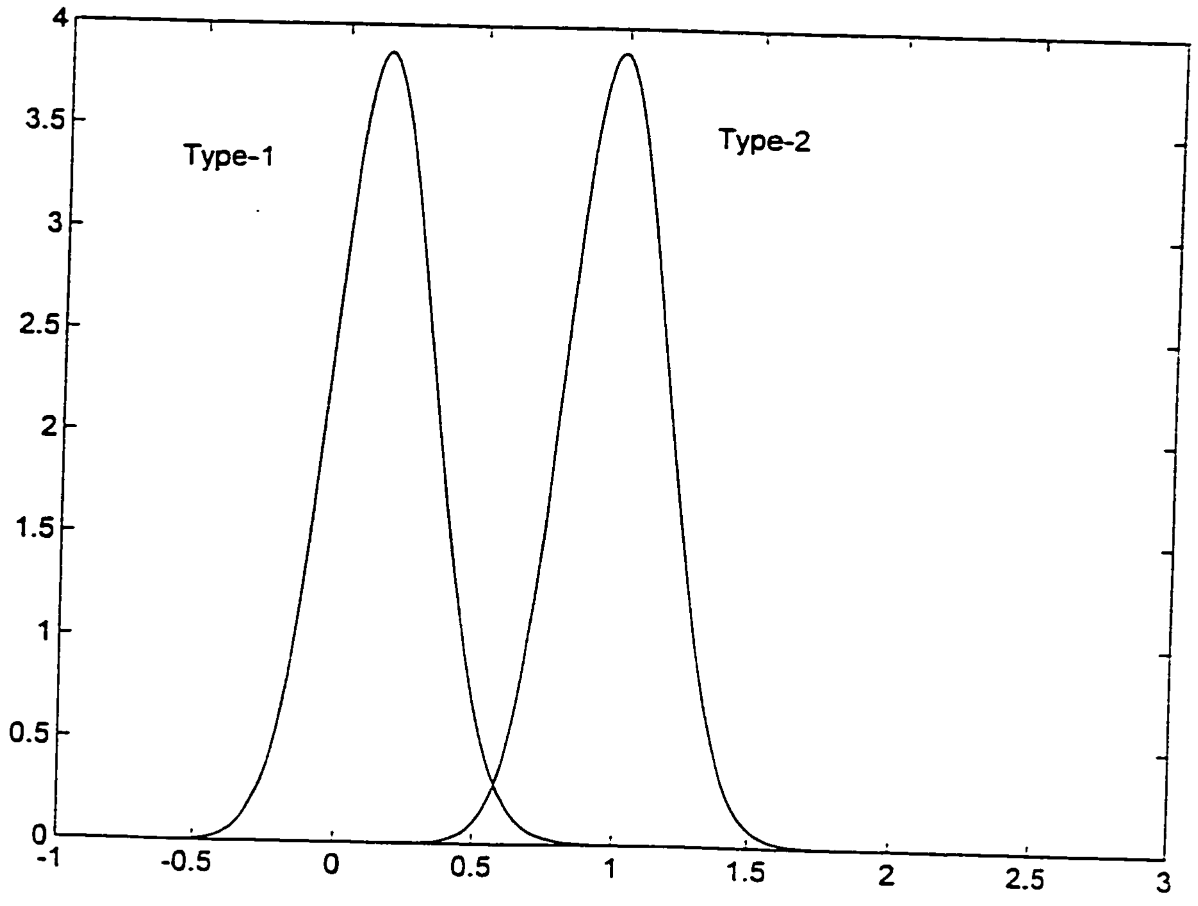


Figure 5.2 Density Functions of the Output of GPSNN for Type-1 and Type-2 Grain Signals

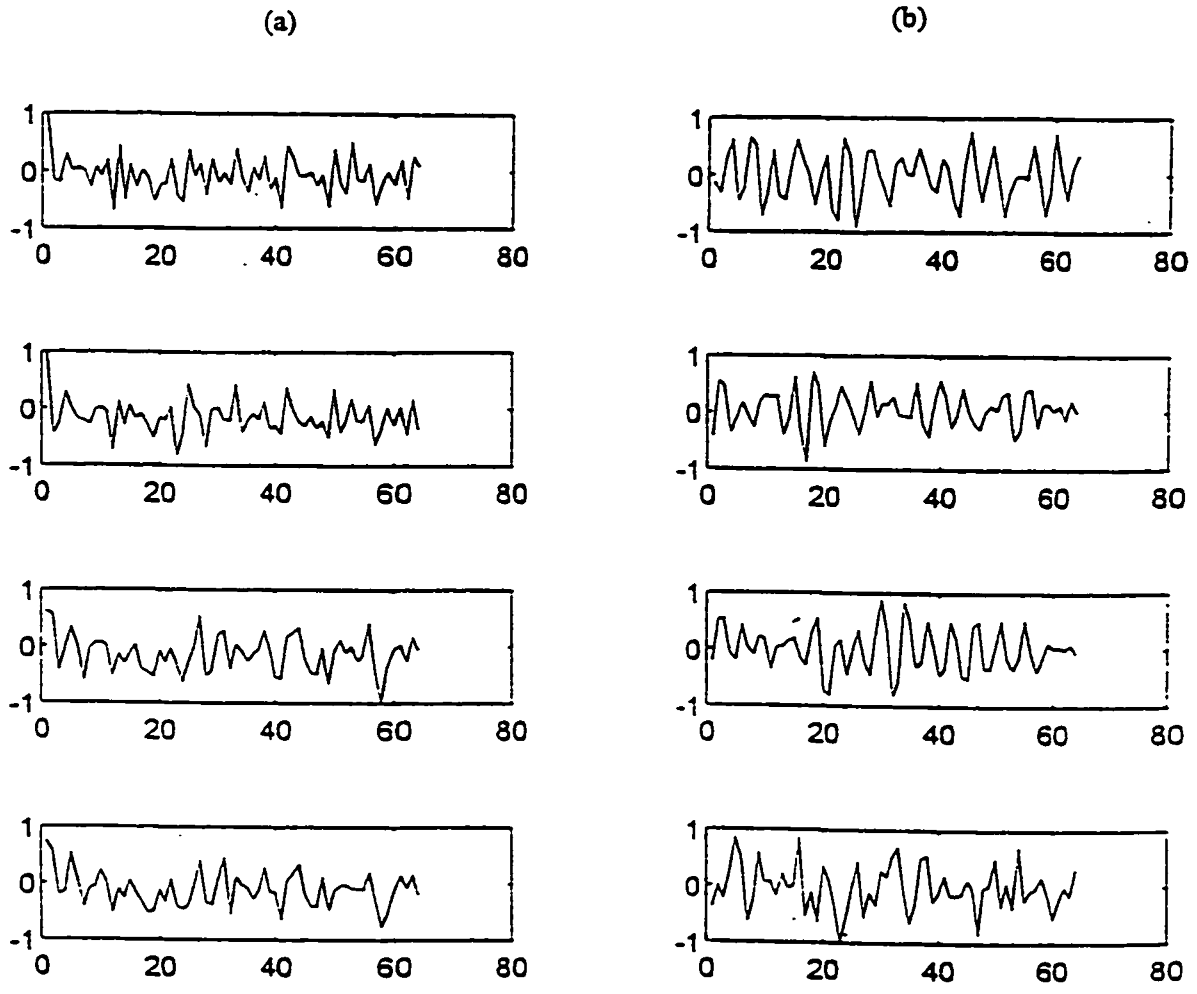


Figure 5.3 Examples of Backscattered Grain Signals, (a) Type-1 Signals, and (b) Type-2 Signals

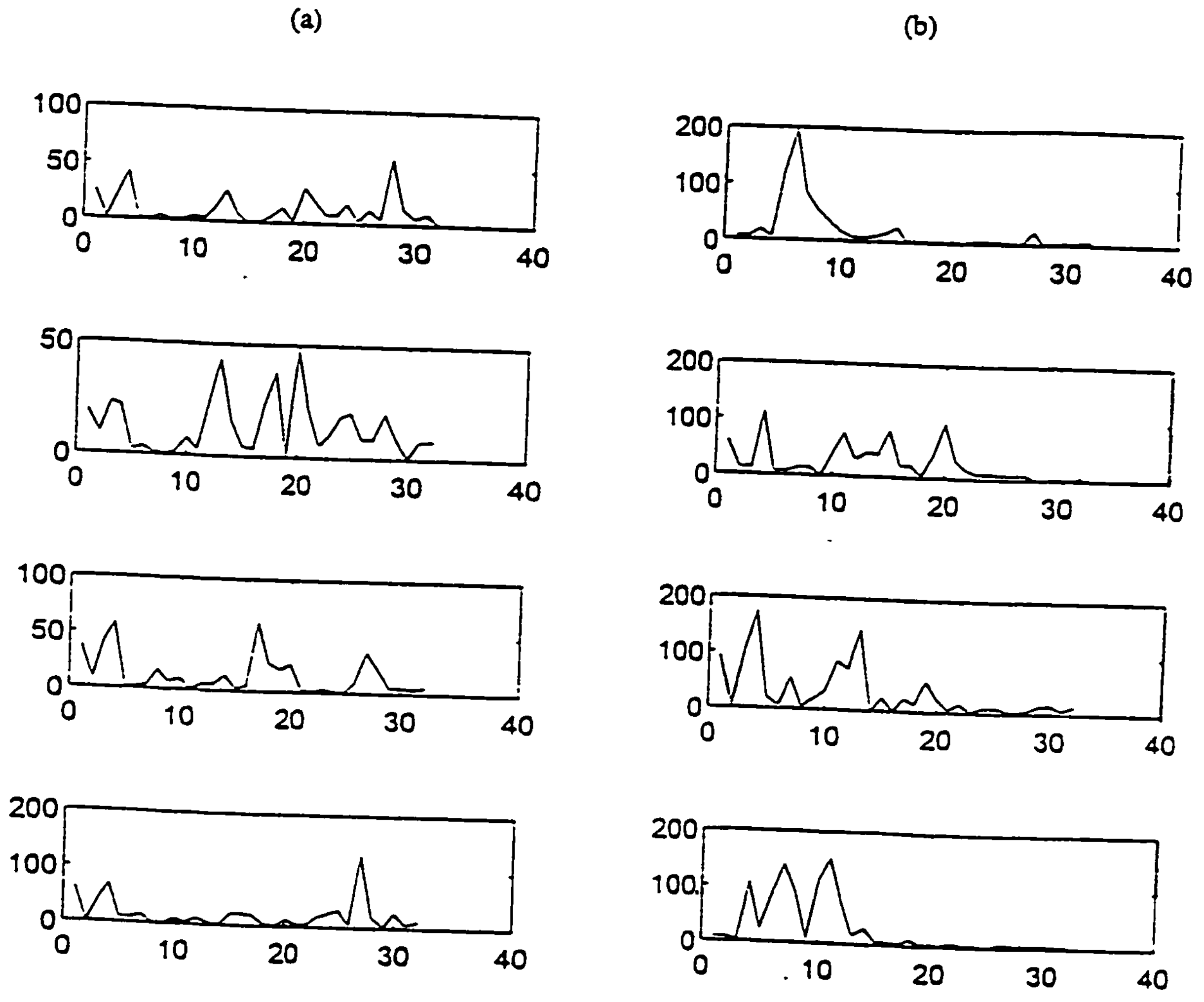


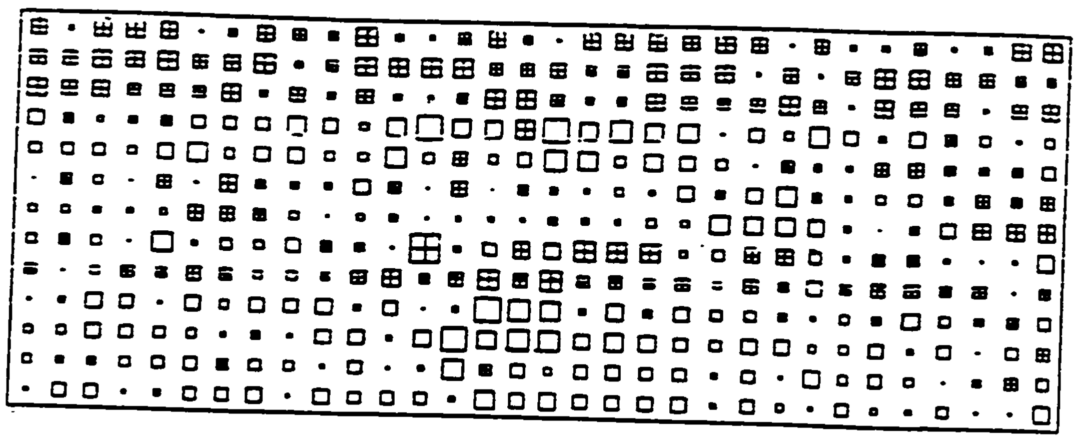
Figure 5.4 Power Spectrum of Grain Signals Shown in Figure 5.3

Table 5.1. Training statistics for GPSNN using Experimental Data

	GPSNN
Number of Sample Vectors	4490
Number of Inputs	32
Number of Outputs	1
Number of Hidden Neurons	13
Epoch	52225

The success of GPSNN may be obscured if there is no understanding of the optimality of the neural network. In this study, Hinton diagrams are used to explain the optimal characteristics of the neural network. Hinton diagrams are named for Geoffrey Hinton who used this method to display neural network weights. Figure 5.5a depicts the Hinton diagram of the GPSNN. The first row shows the weights from the first hidden neuron to the input nodes, the second row depicts the weights from the second hidden neuron to the input nodes, and so on. Note that the number of rows is equal to the number of hidden neurons. Figure 5.5b depicts the weights from the hidden neurons to the output neuron. Since there is only one output neuron, the number of rows in the Hinton diagram is one. Inspection of Hinton diagrams reveals that all hidden nodes contribute to the decision process and the optimal design of the neural network has been realized. Furthermore, there is no situation suggesting that a set of weights yields very large values, very low values, or predictable values. Such observations indicate that

(a)



(b)

Figure 5.5 Hinton diagram Displaying the Estimated Weights of GPSNN, (a) W^h weights, and (b) W^o weights

overfitting is avoided since an adaptive search for the number of nodes in the hidden layer is implemented.

5.4 Time Signature Recognition

An alternative to using the GPSNN for grain signal classification is to train the neural network directly using A-scan data. In this study we have repeated training the neural network by applying the A-scan (64 samples) to the neural network directly. This method is called the grain amplitude neural network (GANN). As discussed earlier, no notable feature can be recognized in A-scan signals due to the random amplitude and phase of the backscattered echoes. In spite of this randomness, it is desirable to probe A-scans using GANN in order to detect any unforeseeable grain scattering characteristics that may exist in the time domain.

To achieve the above objective, an adaptive hidden neuron algorithm is used to determine the number of nodes in the hidden layer. Similar to the training technique of GPSNN, a total of 4,490 training sequences are assembled to train the GANN. A new set of 12,572 testing sequences are utilized to test the GANN performance. Table 5.3 presents the training statistics of GANN. A suboptimal network is obtained by applying 4,490 data sets using 77,875 training iterations. The density function of the GANN output for the Type-1 grain and Type-2 grain signals significantly overlaps. These density functions are shown in Figure 5.6. Consequently, a major difficulty is encountered using GANN for classifying A-scans. The testing results offer a far from desirable 71% correct classifying. This would indicate that there is no significant manifestation of an underlying pattern in the grain A-scan that can be detected by GANN for classification. What appears to be inconsistent is that the neural network performed successfully when the power spectrum of the same grain A-scan was used for classification. This superb performance of GPSNN can be attributed to adequate differences in the power spectrum of the signal, governed by frequency dependent attenuation and scattering (see Equation 5.1) that allows the neural network to adapt for recognition. On the contrary, the A-scan contains and displays information related not only to the power spectrum but also to the random phase spectrum. This random phase interferes with and obscures the inherent frequency characteristics of grain A-scans needed for properly training GANN.

To probe the evaluation of the ultrasonic grain signals even further, the autocorrelation of the A-scans (i.e., the inverse Fourier transform of $X_p(k\Omega)$) was used to train the neural network for recognition. The grain autocorrelation neural network is called GACNN. Table 5.4 presents the training statistics of GACNN. The density function of the GACNN outputs are shown in Figure 5.7. Like the performance of GPSNN, the GACNN offers an excellent grain size recognition performance of 92.75% (see Table 5.5). This is expected of GACNN since the autocorrelation of the grain signal conveys the same frequency characteristics as the power spectrum itself.

5.5 Conclusion

In this study we have developed a neural network that is designed to classify the power spectrum of the backscattered grain signals (i.e., the A-scan) using ultrasound.

The backscattered grain echoes are random signals that bear information related to the grain size and frequency of sound. However this information is not readily quantifiable and lacks uniquely recognizable features. Therefore, the neural network becomes appealing for classifying these signals because they are trainable. In this study, an adaptive hidden neuron algorithm is used to determine the optimal number of neurons both in the hidden layer. The optimal values for neural network weights are estimated using the backpropagation algorithm. Experimental measurements of steel grains are utilized to train and test the grain power spectrum neural network. This network shows a remarkable 98% classification performance. Parallel classification performance is also achieved when training the neural network using the autocorrelation of grain signals. These results are encouraging and suggest that neural networks are potentially useful for nondestructive testing and quality control. Furthermore, the grain power spectrum neural network renders practical advantages such as real-time processing, adaptability and training capability. It is important to point out that similar neural network designs can be used in medical ultrasonic imaging for tissue characterization and diagnosis.

Table 5.2 Summary of the Testing Results for the GPSNN

TYPE-1 Grain Signal	Percent Recognition	TYPE-2 Grain Signal	Percent Recognition
gr106	98.44%	gr206	98.21%
gr107	100%	gr207	93.98%
gr108	100%	gr208	95.99%
gr109	94.20%	gr209	100%
gr1010	97.99%	gr210	95.99%
gr111	92.20%	gr211	100%
gr112	100%	gr212	93.98%
gr113	99.77%	gr213	98.66%
gr114	99.55%	gr214	100%
gr115	97.55%	gr215	98.21%
gr116	94.20%	gr216	100%
gr117	96.88%	gr218	100%
gr118	99.55%	gr219	100%
gr119	100%	gr220	100%

Table 5.3. Training statistics for GANN using Experimental Data

	GPSNN
Number of Sample Vectors	4490
Number of Inputs	64
Number of Outputs	1
Number of Hidden Neurons	18
Epoch	77875

Table 5.4. Training statistics for GACNN using Experimental Data

	GPSNN
Number of Sample Vectors	4490
Number of Inputs	32
Number of Outputs	1
Number of Hidden Neurons	10
Epoch	36545

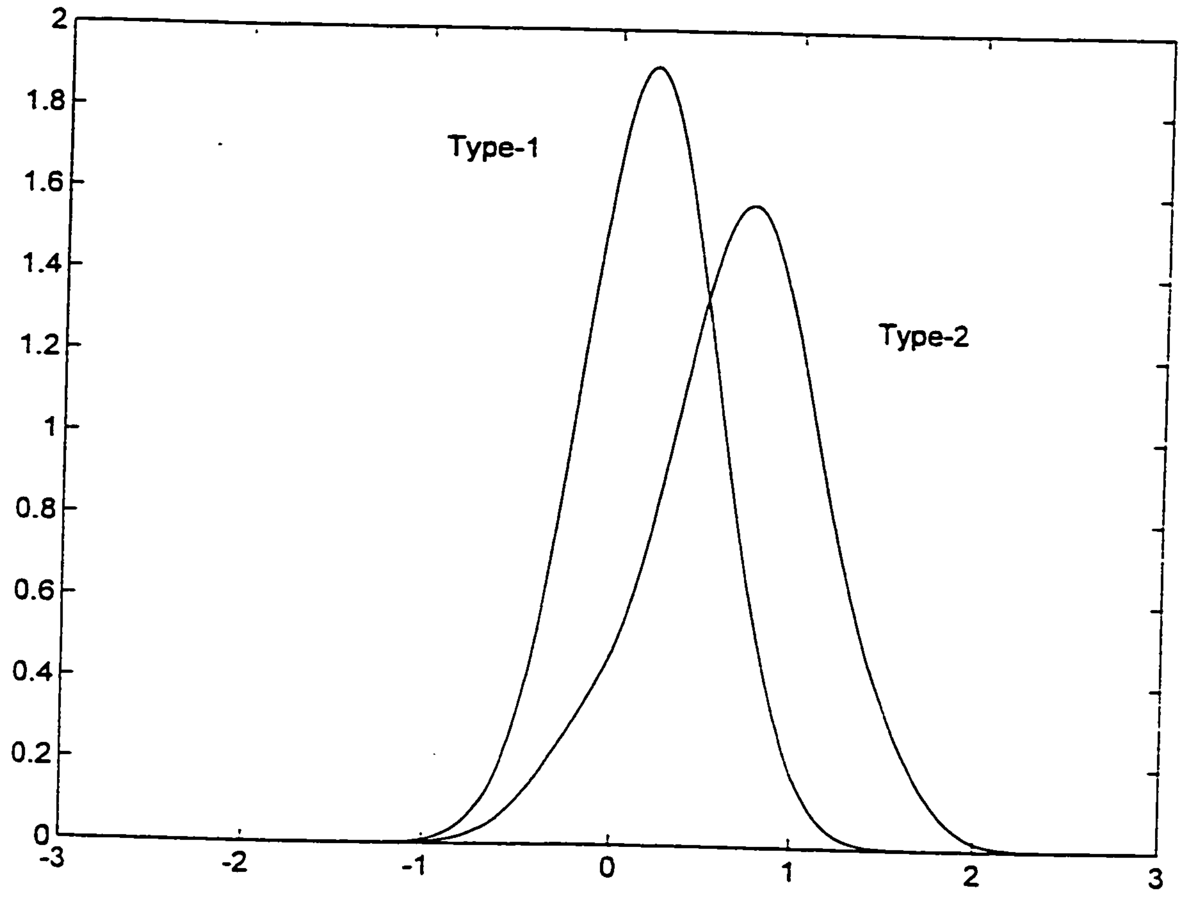


Figure 5.6 Density Functions of the Output of GANN for Type-1 and Type-2 Grain Signals

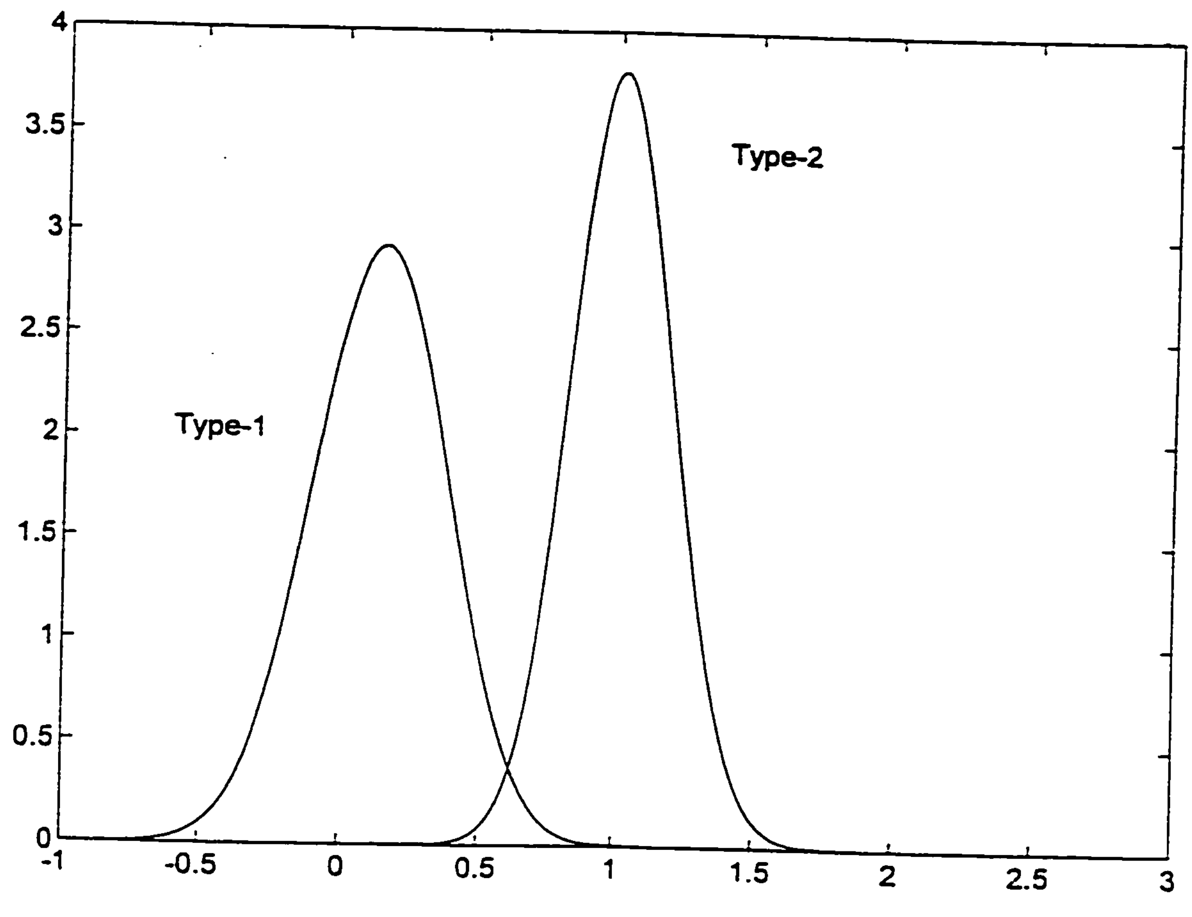


Figure 5.7 Density Functions of the Output of GACNN for Type-1 and Type-2 Grain Signals

Table 5.5 Summary of the Testing Results for the GACNN

TYPE-1 Grain Signal	Percent Recognition	TYPE-2 Grain Signal	Percent Recognition
gr106	98.66%	gr206	83.29%
gr107	96.21%	gr207	75.27%
gr108	94.20%	gr208	87.08%
gr109	86.19%	gr209	84.63%
gr110	99.33%	gr210	95.99%
gr111	88.41%	gr211	94.65%
gr112	84.85%	gr212	85.30%
gr113	87.08%	gr213	98.21%
gr114	87.30%	gr214	98.88%
gr115	98.21%	gr215	100%
gr116	97.10%	gr216	100%
gr117	94.87%	gr217	100%
gr118	90.86%	gr218	100%
gr119	91.09%	gr219	100%

CHAPTER VI
SUMMARY AND CONCLUSION

In this study, three novel deconvolution neural network models (i.e., DNN, ADNN, and PDNN) have been developed in order to detect and resolve multiple interfering target echoes in noisy environments. In Chapter 2, DNN was introduced as the first neural network model. Experimental target echoes with/without noise were used to train it. We got very promising results where obtained in detecting and resolving multiple echoes in the presence of noise (SNR=8 dB). We developed Autoassociative neural network to improve the SNR in the input signal. Then, the improved signal is applied to DNN. By utilizing ANN and DNN, the problem of detection in scattering noise is implemented using two processing stages which are removing the noise and detecting the target echoes. ADNN was able to detect echoes with SNR of 4 dB. An alternative neural network model, PDNN, has been developed in Chapter 3. PDNN allows to estimate statistical parameters of target and noise classes and this is advantageous for PDNN which can use these parameters for echo detection. It has been observed that the PDNN achieves the same performance as ADNN does with less number of weights. PDNN uses the Gram-Charlier coefficients describing the target echoes and the distribution of the scattering noise. Flaws are detected successfully with SNR of 3 dB. These deconvolution techniques are very effective for the problem of detection in the field of ultrasonic signal processing. Results show that the deconvolution neural networks recognize multiple target echoes and lock into the signature of these echoes. Overall, the deconvolution neural networks offer a high probability of detection for a reasonably low signal-to-noise ratio signals.

We improved the backpropagation learning algorithm by utilizing the adaptive hidden neuron algorithm. In the design of neural networks, the unknowns are the number of input neurons, and hidden neurons. The number of input neurons are equal to the length of the target echo in case of DNN and ANN. In case of PDNN, it is equal to the number of Gram-Charlier coefficients which is 4. The number of hidden neurons is problem dependent and has to be changed adaptively during the training phase. Adaptive hidden neuron algorithm increases the number of hidden neurons as needed and causes the neural network solutions to escape from local minimum and reach the global minimum. At the end of each training phase, we used the Hinton diagrams of the neural network weights to evaluate the optimality of the solution.

Another approach to detect the flaws in grain scattering noise is to utilize a method called split-spectrum processing. We developed three order statistic neural network filters to replace the conventional ones in the split-spectrum processing detection algorithm. These methods are minimum, median, and maximum order statistic neural network filters. Testing results show that the output of these neural networks are not 100% accurate. On the other hand, these neural networks provide good estimates of the input rank, reliability, and usefulness to achieve the sorting goal. In addition, all hidden neurons have the same structure which makes its hardware implementation easy using VLSI design techniques. Testing results show that the performance of these filters is improved when signal and noise have good statistical separation representing a particular rank, such as minimum, median, and maximum. We estimated the probability density function of the hidden neurons and the output neuron to analyze the performance of the order statistic neural network filters. We further employed the Hinton diagrams of the neural network weights to investigate their optimality. And we found out that the MedNNet does a superior sorting than that of MinNNet and MaxNNet. The activation levels for the hidden neurons of MedNNet are in the middle region of hyperbolic tangent function where the learning takes place. On the other hand, for MinNNet and MaxNNet, the activations are either at the end or at the beginning of hyperbolic tangent functions where there is less significant learning and adaptation.

In this study, we have developed a neural network that is designed to classify the backscattered grain signals (i.e., the A-scan) based on their power spectrums. The backscattered grain echoes are random signals that bear information related to both the grain size and the frequency of ultrasound. However, this information is not readily quantifiable and lacks uniquely recognizable features. Therefore, the neural network becomes appealing for classifying these signals because they are trainable. In this thesis an adaptive hidden neuron algorithm is used to determine the optimal number of hidden neurons in the hidden layer. The optimal values for neural network weights are estimated using the backpropagation learning algorithm. Experimental measurements of steel grains are utilized to train and test the grain power spectrum neural network. This network shows a remarkable 98 % classification performance.

BIBLIOGRAPHY

[1] Alexandre F., Guyot F. and Haton J. P., "The Cortical Column: A New Processing Unit for Multilayered Networks," Neural Networks, Vol. 4, pp. 15-25, 1991.

[2] Anaya J. J., Ullate L. G., and Fritsch C., "A Method for Real-Time Deconvolution," IEEE Transactions On Instrumentation and Measurement, Vol. 41, No. 3, 1992.Company Limited, 1963.

[3] Atiya A. F., "An Unsupervised Learning Technique for Artificial Neural Networks," Neural Networks, Vol. 3, pp. 707-711, 1990.

[4] Baba N., "A New Approach for Finding the Global Minimum of Error Function of Neural Networks," Neural Networks, Vol. 2, pp. 367-373, 1989.

[5] Bichsel M. and Seitz P., " A Maximum Information Approach to Layered Networks," Neural Networks , Vol. 2, pp. 133-141, 1989.

[6] Blum E. K. and Li L. K., "Approximation Theory and Feedforward Networks," Neural Networks, Vol. 4, pp. 511-515, 1991.

[7] Bounds D. G., Lloyd P. J. and Mathew B. G., "A Comparison of Neural Network and Other Pattern Recognition Approaches to the Diagnosis of Low Back Disorders," Neural Networks, Vol. 3, pp. 583-591, 1990.

[8] Bressloff P. C. and Taylor J. G., "Discrete Time Leaky Integrator Network With Synaptic Noise," Neural Networks, Vol. 4, pp. 789-801, 1991.

[9] Buonomano D. V., Baxter D. A. and Byrne J. H., "Small Networks of Emprically Derived Adaptive Elements Simulate Some Higher-Order Features of Classical Conditioning," Neural Networks, Vol. 3, pp. 507-523, 1990.

[10] Carpenter G. A., "Neural Network Models for Pattern Recognition and Associative Memory," Neural Networks, Vol. 2, pp. 243-257, 1989.

[11] Carpenter G. A., Grossberg S. and Reynolds J. H., "ARTMAP: Supervised Real-Time Learning Classification of Nonstationary Data by a Self-Organizing Neural Network," Neural Networks, Vol. 4, pp. 565-588, 1991.

[12] Chen S., Mulgrew B., and Grant P. M., “A Clustering Technique for Digital Communications Channel Equalization Using Radial Basis Function Networks,” IEEE Transactions On Neural Networks, Vol. 4, No. 4, pp. 570-579, July 1993.

[13] Cherkassky V. and Lari-Najafi H., "Constrained Topological Mapping for Nonparametric Regression Analysis," Neural Networks, Vol. 4, pp. 27-40, 1991.

[14] Chiou C. and Schmerr L. W., "A Neural Network Model for Ultrasonic Flaw Sizing," Nondestr. Test. Eval., Vol.10, pp. 167-182.

[15] Cichocki A. and Unbehauen R., Neural Networks for Optimization and Signal Processing, John Wiley and Sons, 1992.

[16] Daunicht W. J., "DEFAnet- A Deterministic Neural Network Concept for Function Approximation," Neural Networks, Vol. 4, pp. 839-845, 1991.

[17] Deprit E., "Implementing Recurrent Back-Propagation on the Connection Machine," Neural Networks, Vol. 2, pp. 295-314, 1989.

[18] Doya K. and Yoshizawa S., "Adaptive Neural Oscillator Using Continuous-Time Back-Propagation Learning," Neural Networks, Vol. 2, pp. 375-385, 1989.

[19] Eberhardt S. P., Daud T., Kerns D. A., Brown T. X. and Thakoor A. P., "Competitive Neural Architecture for Hardware Solution to the Assignment Problem," Neural Networks, Vol. 4, pp. 431-442, 1991.

[20] Farhat N. H. and Bai B., "Echo Inversion and Target Shape Estimation by Neuromorphic Processing," Neural Networks, Vol. 2, pp. 117-125, 1989.

[21] Freeman J. A. and Skapura D. M., Neural Networks Algorithms, Applications, and Programming Techniques, Addison-Wesley Publishing Company, October 1991.

[22] Fujita O., "A Method for Designing the Internal Representation of Neural Networks and Its Application to Network Synthesis," Neural Networks, Vol. 4, pp. 827-837, 1991.

[23] Giraud B., Liu L. C., Axelrad C. B. and Axelrad H., "Optimal Approximation of Square Integrable Functions by a Flexible One-Hidden-Layer Neural Network of Excitatory and Inhibitory Neuron Pairs," Neural Networks, Vol. 4, pp. 803-815, 1991.

[24] Grossberg S. and Schamajuk N. A., "Neural Dynamics of Adaptive Timing and Temporal Discrimination During Associative Learning," Neural Networks, Vol. 2, pp. 79-102, 1989.

[25] Guez Y., Donohue K. and Bilgutay N. M., "A Neural Network Architecture for Ultrasonic Nondestructive Testing," IEEE Ultrasonics Symposium Proceedings, pp. 777-780, 1991.

[26] Hepp D. J., "An Application of Backpropagation to the Recognition of Handwritten Digits Using Morphologically Derived Features," SPIE, Vol. 1451 Nonlinear Image Processing II, pp. 228-233, 1991.

[27] Hirose Y., K. Yamashita and Hijiya S., "Back-Propagation Algorithm Which Varies the Number of Hidden Units," Neural Networks, Vol. 4, pp. 61-66, 1991.

[28] Ho K. C., Chan Y. T. and Ching P.C., "Adaptive Time-Delay Estimation in Nonstationary Signal and/or Noise Power Environments," IEEE Transactions On Signal Processing, Vol. 41, No. 7, 1993.

[29] Hollis P. W. and Paulos J. J., "Artificial Neural Networks Using MOS Analog Multipliers," Int. Conf. Neural Networks, 1988.

[30] Hollis P. W. and Paulos J. J., "Artificial Neural Networks Using MOS Analog Multipliers," Journal of Solid-State Circuits, Vol. 25, No. 3, pp. 849-855, 1990.

[31] Hollis P. W. and Paulos J. J., "A Neural Network Learning Algorithm Tailored for VLSI Implementation,"IEEE Transactions On Neural Networks, Vol. 5, No. 5, 1994.

[32] Hornik K., "Approximation Capabilities of Multilayer Feedforward Networks," Neural Networks, Vol. 4, pp. 251-257, 1991.

[33] Hornik K., Stinchcombe M. and White H., "Multilayer Feedforward Networks are Universal Approximators," Neural Networks, Vol. 2, pp. 359-366, 1989.

[34] Hornik K., Stinchcombe M. and White H., "Universal Approximation of an Unknown Mapping and its Derivatives Using Multilayer Neural Networks," Neural Networks, Vol. 3, pp. 551-560, 1990.

[35] Hush D. and Horne B. G.,"Progress in Supervised Neural Networks: What's New Since Lippman? " IEEE Signal Processing Magazine, pp. 8-39, January 1993.

[36] Jain K. A., Fundamentals of Digital Image Processing, Prentice Hall, 1989.

[37] Joerding W. H. and Meador J. L., "Encoding A Priori Information in Feedforward Networks," Neural Networks, Vol. 4, pp. 847-856, 1991.

[38] Kammerer B. R. and Kupper W. A., "Experiments for Isolated-Word Recognition with Single and Two-Layer Perceptrons," Neural Networks, Vol.3, pp. 693-706, 1990.

[39] Kechriotis G., Zervas E., and Manolakos E. S., "Using Recurrent Neural Networks for Adaptive Communication Channel Equalization," IEEE Transactions On Neural Networks, Vol. 5, No. 2, pp. 267-278, March 1994.

[40] Kendall M.G. and Stuart A., The Advanced Theory of Statistics, Charles Griffin & Company Limited, 1963.

[41] King T., "Using Neural Networks for Pattern Recognition: Recognizing and Learning Patterns is One Thing Neural Nets Do Best," Dr. Dobb's Journal, pp. 14-20, January 1989.

[42] Klimasauskas C., "Neural Nets and Noise Filtering," Dr. Dobb's Journal, pp. 32-48, January 1989.

[43] Kobuchi Y., "State Evaluation Functions and Lyapunov Functions for Neural Networks," Neural Networks, Vol. 4, pp. 505-510, 1991.

[44] Korn G. A., "A New Environment for Interactive Neural Network Experiments," Neural Networks, Vol. 2, pp. 229-237, 1989.

[45] Kulkarni A. D., "Solving Ill-Posed Problems With Artificial Neural Networks," Neural Networks, Vol. 4, pp. 477-484, 1991.

[46] Lang K. J., Waibel A. H. and Hinton G. E., "A Time-Delay Neural Network Architecture for Isolated Word Recognition," Neural Networks, Vol. 4, pp. 23-43, 1991.

[47] Lee Y. H. and Fam A. T., "An Edge Gradient Enhancing Adaptive Order Statistic Filter," IEEE Trans. Acoust., Speech, Signal Processing, Vol. ASSP-35, no. 5, pp. 680-695, May 1987.

[48] Levin E., "A Recurrent Neural Network: Limitations and Training," Neural Networks, Vol. 3, pp. 641-650, 1991.

[49] Levin E., Gewirtzman R. and Inbar G. F., "Neural Network Architecture for Adaptive System Modelling and Control," Neural Networks, Vol. 4, pp. 185-191, 1991.

[50] Masters T., Practical Neural Network Recipes in C++, Academic Press Inc., 1993.

[51] Moore K. L., "Artificial Neural Networks: Weighing the Different Ways to Systematize Thinking," IEEE Potentials, pp. 27-28, 1992.

[52] Mougeot M., Azencott R. and Angeniol B., "Image Compression With Back Propagation: Improvement of the Visual restoration Using Different Cost Functions," Neural Networks, Vol. 4, pp. 467-476, 1991.

[53] Nguyen D. D. and Lee J. S. J., "A New LMS-Based Algorithm for Rapid Adaptive Classification in Dynamic Environments," Neural Networks, Vol. 2, pp. 215-228, 1989.

[54] Papadakis E. P., "Ultrasonic Attenuation Caused by Scattering in Polycrystalline Metals," Journal of the Acoustical Society of America, Vol. 37, pp. 703-710, 1965.

[55] Papoulis A., Signal Analysis, Mc Graw-Hill Book Company, 1977.

[56] Perlovsky L. I. and McManus M. M., "Maximum Likelihood Neural Networks for Sensor Fusion and Adaptive Classification," Neural Networks, Vol. 4, pp. 89-102, 1991.

[57] Rabiner L. R., Sambur M. R., and Schmidt C. E., "Applications of a Nonlinear Smoothing Algorithm to Speech Processing," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-23, pp. 552-557, Dec. 1975.

[58] Rajavelu A., Musavi M. T. and Shirvaikar M. V., "A Neural Network Approach to Character Recognition," Neural Networks, Vol. 2, pp. 387-393, 1989.

[59] Rigler A. K., Irvine J. M. and Vogl T. P., "Rescaling of Variables in Back Propagation Learning," Neural Networks, Vol. 4, pp. 225-229, 1991.

[60] Sachse W., Sribar I.Grabec, "Intelligent Processing of Ultrasonic Signals for Quantative Material Testing," IEEE Ultrasonics Symposium Proceedings, pp. 767-776, 1991.

[61] Samad T., "Back Propagation With Expected Source Values," Neural Networks, Vol. 4, pp. 615-618, 1991.

[62] Sanger T. D., "Optimal Unsupervised Learning in a Single-Layer Linear Feedword Neural Network", Neural Networks, Vol. 2, pp. 459-473, 1989.

[63] Saniie Jafar, Ultrasonic Signal Processing: System Identification and Parameter Estimation of Reverberant and Inhomogeneous Targets, August 1981.

[64] Saniie J. and Bilgutay N. M., "Quantitative Grain Size Evaluation Using Ultrasonic Backscattered Echoes," Journal of the Acoustical Society of America, Vol. 80, pp. 1816-1824, Dec. 1986.

[65] Saniie J. and Jin X. M., "Spectral Analysis for Ultrasonic NDE Applications Using AR, Prony and MUSIC Methods," Journal of the Acoustical Society of America, pp. tbd, October 1996.

[66] Saniie J. and Nagle D. T., "Analysis of Order-Statistic CFAR Threshold Estimators for Improved Ultrasonic Flaw Detection," IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, Vol. 39, No. 5, September 1992.

[67] Saniie J., Nagle D. T., and Donohue K. D., "Analysis of Order Statistic Filters Applied to Ultrasonic Flaw Detection Using Split-Spectrum Processing," IEEE Trans. on ultra., ferro., and freq. control, Vol. 38, no. 2, pp. 133-140, March 1991.

[68] Saniie J., Unluturk M. and Chu T., "Frequency Discrimination Using Neural Networks with Applications in Ultrasonics Microstructure Characterization," IEEE Ultrasonics Symposium Proceedings, pp. 1195-1199, 1992.

[69] Saniie J., Wang T., and Bilgutay N. M., "Analysis of Homomorphic Processing for Ultrasonic Grain Signal Characterizations", IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control, pp. 365-375, May 1989.

[70] Saniie J., Wang T., and Jin X., "Performance Evaluation of Frequency Diverse Bayesian Ultrasonic Flaw Detection," J. Accoust. Soc. Am. 91 (4), Pt. 1, pp. 2034- 2041, April 1992.

[71] Shawe J. S. and Cohen D. A., "Linear Programming Algorithm for Neural Networks," Neural Networks, Vol. 3, pp. 575-582, 1990.

[72] Shoemaker P. A., Carlin M. J. and Shimabukuro R. L., "Back Propagation Learning With Trinary Quantization of Weight Updates," Neural Networks, Vol. 4, pp. 231-241, 1991.

[73] Sietsma J. and Dow R. J. F., "Creating Artificial Neural Networks That Generalize," Neural Networks, Vol. 4, pp. 67-79, 1991.

[74] Silverman R. H. and Noetzel A. S., "Image Processing and Pattern Recognition in Ultrasonograms by Backpropagation," Neural Networks, Vol. 3, pp. 593-603, 1990.

[75] Sontag E. D. and Sussmann H. J., "Back Propagation Seperates Where Perceptrons Do," Neural Networks, Vol. 4, pp. 243-249, 1991.

[76] Specht D. F., "Probabilistic Neural Networks," Neural Networks, Vol. 4, pp. 109-118, 1990.

[77] Takadoya M., Notake M., Yabe Y., Ogi T., Kitahara M. and Achenbach J. D., "Quantitative Evaluation Of Defects By Neural Network," Nondestr. Test. Eval., Vol. 8-9, pp. 443-451, 1992.

[78] Tao Y., "Approximation of Functions on a Compact Set by Finite Sums of a Sigmoid Function Without Scaling," Neural Networks, Vol. 4, pp. 817-826, 1991.

[79] Tenorio D. and Tenorio M. F., "Short Utterance Recognition Using Network With Minimum Training," Neural Networks, Vol. 4, pp. 711-722, 1991.

[80] Tollenaere T., "SuperSAB: Fast Adaptive Back Propagation with Good. Scaling Properties," Neural Networks, Vol. 3, pp. 561-573, 1990.

[81] Unluturk M. S. and Saniie J., "Deconvolution Neural Networks for Ultrasonic Testing," 1995 IEEE Ultrasonics Symposium, pp. 715-719, 1995.

[82] Unluturk M. and Saniie J., "Neural Networks for Ultrasonic Grain Size Discrimination," 1996 IEEE Ultrasonics Symposium, 1996.

[83] Van Der Maas H. L. J., Verschure P. F. M. J. and Molenaar P. C. M., "A Note on Chaotic Behavior in Simple Neural Networks," Neural Networks, Vol. 3, pp. 119-122, 1990.

[84] Van Hulle M. M. and Orban G. A., "Representation and Processing in a Stochastic Neural Network: An Integrated Approach," Neural Networks, Vol. 4, pp. 643-655, 1991.

[85] Wang T., Ultrasonic Signal Processing and Pattern recognition in Evaluating the Microstructure of Materials, December 1987.

[86] Wang T., Saniie J., and Jin X., "Analysis of Low-Order Autoregressive Models for Ultrasonic Grain Signal Characterization," IEEE Transactions on Ultrasonics, Ferroelectrics, and frequency control, vol. 38, No. 2, March 1991.

[87] Watanabe S. and Yoneyama M., "An Ultrasonic Visual sensor Using Neural Network and its Application to Automatic Object Recognition," IEEE Ultrasonics Symposium Proceedings, pp. 781-784, 1991.

[88] White H., "Connectionist Nonparametric Regression: Multilayer Feedforward Networks Can Learn Arbitrary Mappings," Neural Networks, Vol. 3, pp. 535-549, 1990.

[89] Winters J. H. and Rose C., "Minimum Distance Automata in Parallel Networks for Optimum Classification," Neural Netwoks, Vol. 2, pp. 127-132, 1989.

[90] Wong K. M. and Chen S., "Detection of Narrow-band Sonar Signals Using Order Statistical Filters," IEEE Trans. Acoust., Speech, Signal Processing, vol. ASSP-35, no. 5., pp 597-613, May 1987.

[91] Yao Y., Freeman W., Burke J., B. and Yang Q., "Pattern Recognition by a Distributed Neural Network: An Industrial Application," Neural Networks, Vol. 4, pp. 103, 121, 1991.

[92] Youn D.H., Ahmed N., and Carter G. C., "On Using the LMS Algorithm for Time Delay Estimation,"IEEE Transactions On Acoustics, Speech, and Signal Processing, Vol. ASSP- 30, No. 5, 1982.