

**Neural Network based studies on  
Spectroscopic Analysis  
and  
Image Processing**

**Saritha M**

International School of Photonics  
Cochin University of Science and Technology  
Kochi- 682022, Kerala, India

Ph. D. Thesis submitted to  
Cochin University of Science and Technology  
in partial fulfillment of the requirements for the  
Degree of Doctor of Philosophy  
**February, 2010**

**Neural Network based studies on Spectroscopic Analysis and  
Image Processing**

*Ph. D. Thesis*

**Author:**

Saritha M

Research Fellow, International School of Photonics

Cochin University of Science and Technology

Kochi – 682 022, India

Email: saritha\_madhu@rediffmail.com

**Research Advisors:**

Dr. V M Nandakumaran

Professor, International School of Photonics

Cochin University of Science and Technology

Kochi – 682 022, India

Email: nandak@cusat.ac.in

Dr. V P N Nampoori

Professor, International School of Photonics

Cochin University of Science and Technology

Kochi – 682 022, India

Email: vpnnampoori@cusat.ac.in

International School of Photonics,

Cochin University of Science and Technology

Kochi – 682 022, India

URL: [www.photonics.cusat.edu](http://www.photonics.cusat.edu)

February, 2010.

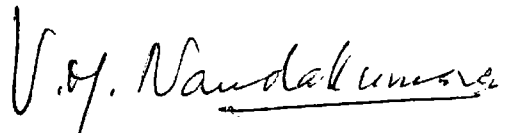
Cover design: Praveen N, Arun S Nair

## CERTIFICATE

Certified that the work presented in the thesis entitled “**Neural Network based studies on Spectroscopic Analysis and Image Processing**” is based on the original work done by Mrs. Saritha M under my guidance and supervision at the International School of Photonics, Cochin University of Science and Technology, Kochi-22, India and has not been included in any other thesis submitted previously for the award of any degree.

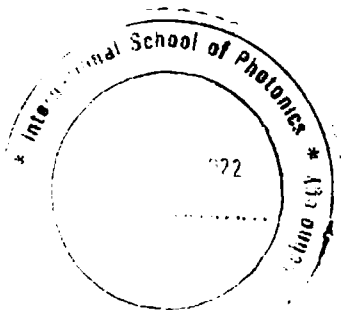
Kochi – 682022

5<sup>th</sup> February 2010.



Prof. V M Nandakumaran

(Supervising Guide)



## DECLARATION

Certified that the work presented in the thesis entitled “**Neural Network based studies on Spectroscopic Analysis and Image Processing**” is based on the original work done by me under the guidance of Dr. V M Nandakumaran, Professor, International School of Photonics, Cochin University of Science and Technology, Kochi-22, India and the co-guidance of Dr. V P N Nampoori, Professor, International School of Photonics, Cochin University of Science and Technology, Kochi-22, India and it has not been included in any other thesis submitted previously for the award of any degree.

Kochi – 682 022

5<sup>th</sup> February 2010

  
Saritha M

## **Acknowledgements**

I express my gratitude and heartfelt thanks to Prof. V M Nandakumaran and V P N Nampoori for the supervision, guidance and support, without which I could not have completed this work. I sincerely thank them for their valuable suggestions and encouragement given to me.

I am grateful to Prof. P Radhakrishnan for encouraging me throughout the period of my research

I also thank Mr. Kailasnath M for his valuable support and inspiration.

I thank Dr. Dann V J, Manu Punnen John and Lyjo Joseph for the timely help extended to me throughout my research period.

I am extremely thankful to all my friends in ISP for their invaluable help extended to me. Without their help it would not have been possible for me to complete the work in time

I have no words to express my gratitude to my family whose encouragement and support helped me in the successful completion of this work. I bow my head to my mother, the real source of inspiration.

Thanks to all and everyone around me

Saritha M

# Preface

Artificial Neural Networks (ANNs) are computational modeling tools that have found extensive acceptance in many disciplines for modeling complex real-world problems. ANNs may be defined as structures comprised of densely interconnected adaptive simple processing elements (called artificial neurons or nodes) that are capable of performing massively parallel computations for data processing and knowledge representation. Although ANNs are drastic abstractions of the biological counterparts, the idea of ANNs is not to replicate the operation of the biological systems but to make use of what is known about the functionality of the biological networks for solving complex problems. The attractiveness of ANNs comes from the remarkable information processing characteristics of the biological system such as nonlinearity, high parallelism, robustness, fault and failure tolerance, learning capability, ability to handle imprecise and fuzzy information, and their ability to generalize. One of the recently emerged applications of ANN is digital image processing. Interest in digital image processing stems from two principal application areas: improvement of pictorial information for human interpretation; and processing of image data for storage, transmission, and representation for autonomous machine perception.

Chapter 1 gives the introduction to artificial neural network and digital image processing. In this chapter, the definition of neural network, the comparison of ANN with human brain, the infinite of neural networks, the various activation functions used, the different learning processes, a brief history and the various learning algorithms like perceptron and backpropagation algorithms are dealt with. This chapter also gives a brief

introduction to image processing also. Here a definition of the digital image is given. Also the two-dimensional DFT and its inverse is discussed as a tool for digital image processing and the various interpolation techniques like nearest neighbor, bilinear interpolation, bicubic and spline techniques are introduced.

**Chapter 2** gives an idea about the development of a successful artificial neural network. It gives a detailed discussion of the six phases of development of a ANN project ranging from the problem definition to the implementation of the network. Also here a discussion is done on the general issues of ANN development like the data size and partitioning, data preprocessing, data normalization, input/output representation, network weight initialization, determination of parameters like learning rate, momentum coefficient and transfer function, the convergence criteria, number of training cycles, hidden layer size etc.

**Chapter 3** gives an application of the neural network. Here a technique to automate the spectrum identification is given. The different modeling issues are dealt. Also a system is developed to identify elements like Ca, Cd, Fe, Li, Hg, K and Sr in a given sample. After the successful development of such a system, attempt is done to automate the spectrum identification. For that a system is developed to identify elements like Ti, Ca, Al and Sn. The system was able to identify the elements present in the spectrum obtained using a CCD camera coupled to a spectrograph having a grating blazed at 750nm with 1200grooves/mm and using the fundamental emission of Nd:YAG laser having 10ns pulse width.

**Chapter 4** gives another application of neural network. Here neural network is employed in the digital image processing field. The super resolution of binary image with discrete cosine transform (DCT) is

done with ANN. An introduction to DCT is done. In this chapter a discussion is done on the variation of the neural network output with the number of hidden layer neurons, the input weight initialization and the number of iterations is discussed. A neural network is trained to super-resolve a 16x16 binary image to a 32x32 binary image. The binary images considered are images of numbers ranging from 0-9. The output of the neural network is compared to the output obtained using the existing methods.

Chapter 5 gives a discussion on the restoration of gray level images with DCT. Here the variation of neural network output with variations in the activation function, the selection in the input data given for training, the selection of proper training algorithms etc are discussed. An ANN is trained to enlarge a 128x128 image to 256x256. The performance of the network is appreciable. The same network was used to enlarge a 256x256 image to 512x512 with good performance.

Chapter 6 gives a discussion on the various noises affecting the digital image. It also gives an introduction to the noise immunity capability of the ANNs. The network developed to restore the binary images and the gray level images are tested with the various noises like Gaussian noise and impulse noise. The performances of these systems with the existing methods are evaluated.

Chapter 7 gives a brief discussion on the future scope of ANNs.



## **List of Publications**

### **In Journals**

- [1]** Saritha M and V.P.N. Nampoorei, 2009. Identification of spectral lines of elements using artificial neural networks Microchemical Journal 91 pp. 170-175
- [2]** Saritha M, V. M. Nandakumaran and V.P.N. Nampoorei Learning based super resolution of binary Images using Discrete Cosine Transforms Submitted to Journal of Experimental and Theoretical Artificial Intelligence
- [3]** Saritha M, V. M. Nandakumaran and V.P.N. Nampoorei Interpolation of Gray level Images using Discrete Cosine Transform Submitted to Journal of Experimental and Theoretical Artificial Intelligence

### **In Conferences**

- [1]** Saritha M and V. P. N. Nampoorei, 2002. Peak Identification in Optical Spectrum using Artificial Neural Networks. Proc. of DAE BRNS National Laser Symposium, pp. 578-580.

## ABBREVIATIONS

2-D	Two Dimension
3-D	Three Dimensions
AI	Artificial Intelligence
ANN	Artificial Neural Network
ASCE	American Society of Civil Engineers
BP	Backpropagation
BPANN	Backpropagation Artificial Neural Network
BT	Batch Training
CCD	Charged Coupled Devices
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
EET	Example-by-example Training
EWR	Example-to-weight ratio
FFT	Fast Fourier Transform
HN	Hidden Neurons
IDCT	Inverse Discrete Cosine Transform
IEEE	Institute of Electrical and Electronics Engineers
INNS	International Neural Network Society
LAM	Linear Associative Memory
LR	Low-resolution
MAP	Maximum a Posteriori
MDCT	Modified Discrete Cosine Transform
MLP	Multilayer Perceptron
Nd:YAG	Neodymium doped Yttrium Aluminum Garnet
NHN	Number of Hidden Neurons
$N_{INP}$	Number of nodes in input
$N_{OUT}$	Number of nodes in output
$N_{TRN}$	Number of Training patterns
$N_{TST}$	Number of testing patterns
$N_w$	Number of weights
OLAM	Optimal Linear Associative Memory
PDF	Probability Density Function

P-MAP	Poisson-Maximum a Posteriori
PSNR	Peak Signal to Noise ratio
SR	Super- resolution
SSE	Sum of Squared Errors
VQ	Vector Quantization

## Contents

CHAPTER 1 .....	1
INTRODUCTION .....	1
NEURAL NETWORKS AND DIGITAL IMAGE PROCESSING.1	
1.1 Introduction .....	1
1.2 What is a Neural Network? .....	3
1.2.1 Human Brain .....	6
1.3 Models of a Neuron .....	12
1.4 Types of activation functions .....	17
1.4.1 A step function .....	17
1.4.2 Piecewise-Linear Function .....	18
1.4.3 Sigmoid Function .....	19
1.5 Perceptrons .....	22
1.6 Learning Processes .....	24
1.7 A Brief History .....	25
1.8 Learning Rules .....	28
1.9 Learning Algorithms .....	32
1.9.1 The Perceptron Algorithm .....	33
1.9.2 The Backpropagation Algorithm .....	35
1.10 Digital Image Processing: .....	38
1.10.1 Image Representation, Sampling, Quantization .....	38
1.11 Various tools for Digital Image Processing .....	44
1.11.1 The Two-Dimensional DFT and its Inverse .....	44
1.12 Image Interpolation techniques .....	48
1.12.1 Nearest Neighbour Interpolation .....	48
1.12.2 Bilinear Interpolation .....	48
1.12.3 Bicubic Interpolation .....	52
1.12.3a Bicubic spline interpolation .....	53
1.12.3b Bicubic convolution algorithm .....	55
1.12.4 Spline Interpolation .....	56
Summary .....	58
References .....	59
CHAPTER 2 .....	63
DEVELOPMENT OF A SUCCESSFUL ARTIFICIAL NEURAL NETWORK .....	63
2.1 Introduction .....	63

2.2	Backpropagation networks.....	63
2.3	BP Algorithm .....	66
2.4	ANN Development Project .....	70
2.5	General issues in ANN development .....	72
2.5.1	Database size and partitioning .....	73
2.5.2	Data preprocessing, balancing, and enrichment .....	74
2.5.3	Data normalization.....	76
2.5.4	Input /output representation .....	77
2.5.5	Network weight initialization .....	78
2.5.6	BP learning rate ( $\eta$ ) .....	79
2.5.7	BP momentum coefficient ( $\mu$ ) .....	80
2.5.8	Transfer function, $\sigma$ .....	81
2.5.9	Convergence criteria .....	82
2.5.10	Number of training cycles.....	84
2.5.11	Training modes .....	85
2.5.12	Hidden layer size .....	86
2.5.13	Parameter optimization .....	88
	<b>Summary</b> .....	89
	<b>References</b> .....	91
<b>CHAPTER 3</b> .....		97
<b>IDENTIFICATION OF SPECTRAL LINES OF ELEMENTS WITH ARTIFICIAL NEURAL NETWORKS</b> .....		97
3.1	Introduction .....	97
3.2	Modelling Issues .....	99
3.3	The Results.....	106
3.4	An Automated System .....	111
3.5	The Approach.....	114
3.6	The Output .....	117
	Summary .....	124
	References.....	125
<b>CHAPTER 4</b> .....		129
<b>LEARNING BASED SUPER-RESOLUTION OF BINARY IMAGES WITH DISCRETE COSINE TRANSFORMS</b> .....		129
4.1	Introduction .....	129
4.2	Discrete Cosine Transforms .....	134
	DCT-I.....	136

DCT-II .....	136
DCT-III .....	137
DCT-IV .....	138
DCT V-VIII .....	138
Inverse transforms.....	139
Multidimensional DCTs .....	140
4.3 Super Resolution .....	141
4.4 Network Design and Training.....	142
4.5 The Output .....	153
Summary.....	157
References:.....	158
CHAPTER 5 .....	163
RESTORATION OF GRAY LEVEL IMAGES WITH DISCRETE COSINE TRANSFORMS .....	163
5.1 Introduction .....	163
5.2 Shrinking and Zooming of image .....	163
5.3 Neural Networks and Image Interpolation.....	169
5.4 Neural Network Design and Training .....	171
5.5 Variations in Backpropagation Algorithms .....	177
5.6 Simulation Results .....	181
Summary.....	193
Reference: .....	193
CHAPTER 6 .....	199
RECONSTRUCTION OF IMAGES FROM NOISE EMBEDDED DATA .....	199
6.1 Introduction .....	199
6.2 Noise Immunity of Multilayer Perceptrons.....	204
6.3 Effect of Noise on Binary Images.....	206
6.3.1 Effect of Gaussian Noise .....	206
6.3.2 Effect of salt and pepper noise.....	210
6.4 Effect of Noise on Gray level Images .....	214
6.4.1 Effect of Gaussian Noise .....	214
6.4.1 Effect of Salt and Pepper Noise.....	217
Summary.....	219
References:.....	220
FUTURE SCOPE .....	223

# CHAPTER 1

## INTRODUCTION

# NEURAL NETWORKS AND DIGITAL IMAGE PROCESSING

## 1.1 Introduction

Artificial Neural Networks (ANNs) are computational modeling tools that have found extensive acceptance in many disciplines for modeling complex real-world problems. ANNs may be defined as structures comprised of densely interconnected adaptive simple processing elements (called artificial neurons or nodes) that are capable of performing massively parallel computations for data processing and knowledge representation. Although ANNs are drastic abstractions of the biological counterparts, the idea of ANNs is not to replicate the operation of the biological systems but to make use of what is known about the functionality of the biological networks for solving complex problems. The attractiveness of ANNs comes from the remarkable information processing characteristics of the biological system such as nonlinearity, high parallelism, robustness, fault and failure tolerance, learning capability, ability to handle imprecise and fuzzy information, and their ability to generalize. Artificial models possessing such characteristics are desirable because (i) nonlinearity allows better fit to the data, (ii) noise-insensitivity provides accurate prediction in the presence of uncertain data

and measurement errors, (iii) high parallelism implies fast processing and hardware failure-tolerance, (iv) learning and adaptivity allow the system to update (modify) its internal structure in response to changing environment, and (v) generalization enables application of the model to unlearned data. The main objective of ANN-based computing (neurocomputing) is to develop mathematical algorithms that will enable ANNs to learn by mimicking information processing and knowledge acquisition in the human brain. ANN-based models are empirical in nature, however they can provide practically accurate solutions for precisely or imprecisely formulated problems and for phenomena that are only understood through experimental data and field observations. In microbiology, ANNs have been utilized in a variety of applications ranging from modeling, classification, pattern recognition, and multivariate data analysis (Basheer and Hajmeer,2000).

One of the recently emerged applications of ANN is digital image processing. Interest in digital image processing stems from two principal application areas: improvement of pictorial information for human interpretation; and processing of image data for storage, transmission, and representation for autonomous machine perception. An image may be defined as a two dimensional function,  $f(x,y)$ , where  $x$  and  $y$  are spatial coordinates, and the amplitude of  $f$  at any pair of coordinates  $(x, y)$  is called the intensity or gray level of the image at that point. When  $(x,y)$  and the amplitude values of  $f$  are all finite, discrete quantities, it is called as a digital image. The field of digital image processing refers to processing digital images by means of a digital computer. A digital image is composed of a finite number of elements each having a particular location



and value. These elements are referred to as picture elements, image elements, pels and pixels. The areas of application of digital image processing are wide and varied (Gonzalez and Woods, 2002).

## 1.2 What is a Neural Network?

Work on artificial neural networks has been motivated right from its inception by the recognition that the human brain computes in an entirely different way from the conventional digital computer. The brain is a highly complex, nonlinear and parallel information processing system. It has the capability to organize its structural constituents, known as neurons, so as to perform certain computations many times faster than the fastest digital computer in existence today. At birth, a brain has great structure and the ability to build its own rules through experience. One of the best examples is the acquiring of specific natural language as the mother tongue. Indeed, experience is built up over time, with the most dramatic development of the human brain taking place during the first two years from birth; the development continues well beyond that stage (Haykin,2003).

A developing neuron is synonymous with a plastic brain: Plasticity permits the developing nervous system to adapt to its surrounding environment. Just as plasticity appears to be essential to the functioning of neurons as information-processing units in the human brain, so it is with neural networks made up of artificial neurons. In its most general form, neural network is a machine that is designed to model the way in which the brain performs a particular task or function of interest; the network is usually implemented by using electronic

components or is simulated in software on a digital computer. To achieve good performance, neural networks employ a massive interconnection of simple computing cells referred to as neurons or processing units. A neural network can be considered as a massively distributed processor made up of simple processing units, which has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:

- Knowledge is acquired by the network from its environment through a learning process
- Interneuron connection strengths, known as synaptic weights, are used to store acquired knowledge (Haykin, 2003).

It is apparent that a neural network derives its computing power through, (i) its massively parallel distributed structure and (ii) its ability to learn and therefore to generalize. Generalization refers to the neural network producing reasonable outputs for inputs not encountered during training (learning). These two information processing capabilities make it possible for neural networks to solve complex problems that are currently intractable (Haykin,2003).

The use of neural networks offers the following properties and capabilities (Hagan et. al., 2002). An artificial neuron can be linear or nonlinear. A neural network, made up of an interconnection of nonlinear neurons, is itself nonlinear. Another capability of the neural network is its input-output mapping property (Haykin, 2003). The neural network learns from the examples by constructing an input-output mapping for the problem. Neural networks have a built in capability to adapt their synaptic weights to change in the surrounding environment. In particular, a neural

network trained to operate in a specific environment can easily be retrained to deal with minor changes in the operating environmental conditions. Another property of neural network is its evidential response. In the context of pattern classification, a neural network can be designed to provide information not only about which particular pattern to select, but also about the confidence in the decision made. This latter information may be used to reject ambiguous patterns and thereby improve the classification performance of the network (Haykin,2003).

Knowledge is represented by the very structure and activation state of a neural network. Every neuron in the network is potentially affected by the global activity of all other neurons in the network. Consequently, contextual information is dealt with naturally by a neural network. As indicated earlier, a neural network, implemented in hardware form, has the potential to be inherently fault tolerant, or capable of robust computation, in the sense that its performance degrades gracefully under adverse operating conditions. For example, if a neuron or its connecting links are damaged, recall of a stored pattern is impaired in quality. However, due to the distributed nature of information stored in the network, the damage has to be extensive before the overall response of the network is degraded seriously. Thus in principle, a neural network exhibits a graceful degradation in performance rather than catastrophic failure. The massively parallel nature of a neural network makes it potentially fast for the computation of certain tasks. This feature makes a neural network well suited for implementation using very-large-scale-integrated (VLSI) technology (Haykin, 2003). An important property of neural network is its uniformity of analysis and design. The same notation

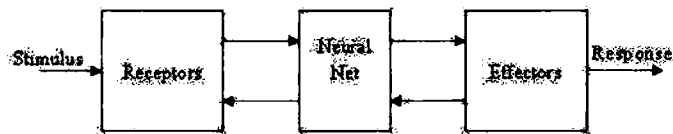
is used in all domains involving the application of neural networks. This feature manifests itself in different ways.

- Neuron in one form or another, represent an ingredient common to all neural networks
- This commonality makes it possible to share theories and learning algorithms in different applications of neural network.
- Modular networks can be built through a seamless integration of modules.

The design of a neural network is motivated by analogy with the brain, which is the living proof that fault to learnt parallel processing is not only physically possible but also sufficiently fast and powerful (Haykin, 2003).

### 1.2.1 Human Brain

The human nervous system may be viewed as a three-stage system as shown in Fig.1.1. Central to the system is the brain, represented by the neural net, which continually receives information, perceives it and make appropriate decisions. Two sets of arrows are shown in the Figure. Those pointing from the left to right indicate the forward transmission of information-bearing signals through the system.

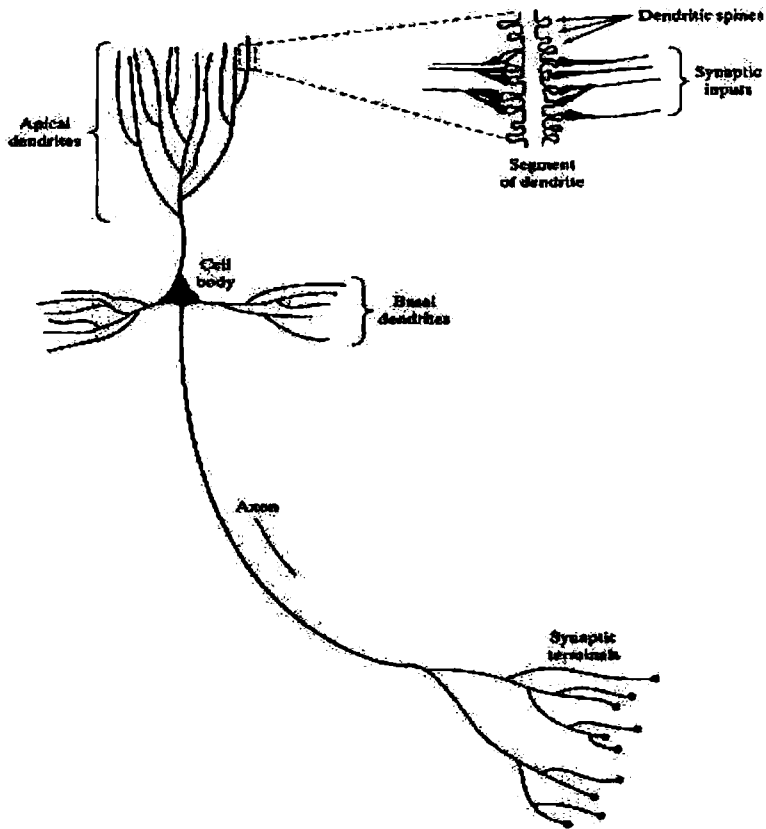


*Fig. 1.1 Block diagram representation of nervous system.*

The arrows pointing from right to left signify the presence of feed-back in the system. The receptors convert stimuli from the human body or the external environment into electrical impulse that convey information to the neural net. The effectors convert electrical impulse generated by the neural net into discernible responses as system outputs.

The human nervous system consists of billions of neurons of various types and lengths relevant to their location in the body (Schalkoff, 1997). The struggle to understand the brain has made easier because of the pioneering work of Ramon y Cajal, who introduced the idea of neurons as structural constituents of brain (Haykin, 2003). Typically, neurons are five to six orders of magnitude slower than silicon logic gates. However, the brain makes up for the relatively slow rate of operation of a neuron by having a truly staggering number of neurons with massive interconnections between them. It is estimated that there are approximately 10 billion neurons in the human cortex, and 60 trillion synapses or connections. The net result is that the brain is an enormously efficient structure. A neuron has three principal components: the dendrite, the cell body and the axon. The dendrites are tree-like receptive networks of nerve fibres that carry electrical signals into the cell body as in Fig.1.2. The cell body has a nucleus that contains information about heredity traits, and a plasma that holds the molecular equipment used for producing the material needed by the neuron (Jain et. al., 1996). The dendrites receive signals from other neurons and pass them over to the cell body. The total receiving area of the dendrites of a typical neuron is approximately  $0.25 \text{ mm}^2$  (Zupan and Gasteiger, 1993). The cell body effectively sums and thresholds these incoming signals. The axon is a single long fibre that carries the signal

from the cell body out to other neurons. The point of contact between an axon of one cell and a dendrite of another cell is called a synapse. It is the arrangement of neurons and the strengths of the individual synapses, determined by a complex chemical process that establishes the function of the neural network (Haykin, 2003).

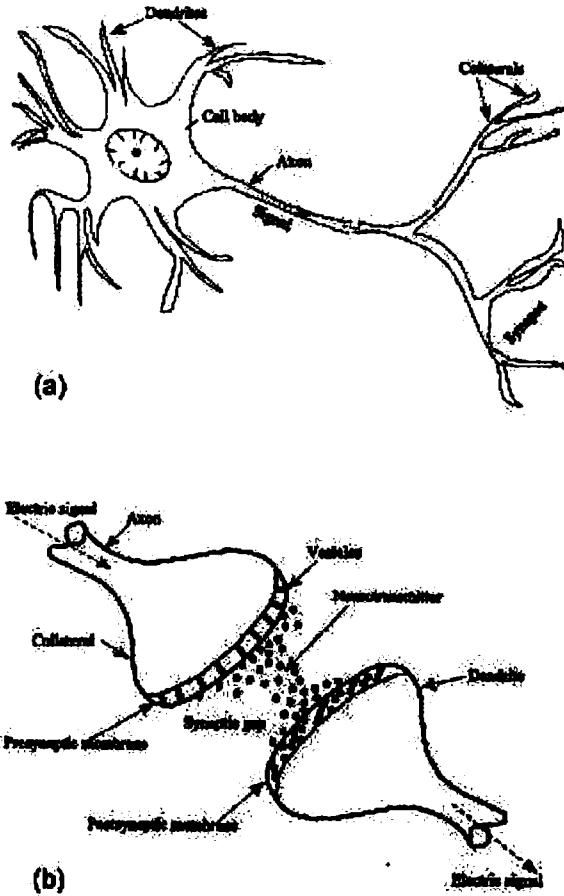


*Fig. 1.2 The Pyramidal Cell*

Synapses are elementary structural and functional units that mediate the interaction between neurons. The most common kind of synapse is a chemical synapse, which operates as follows: A presynaptic process liberates a transmitter substance that diffuses across the synaptic junction between neurons and then acts on a post synaptic process. Thus a synapse converts a presynaptic electrical signal into a chemical signal and then back into a post synaptic electrical signal. In traditional descriptions of neural organization, it is assumed that a synapse is a simple connection that can impose excitation or inhibition, but not both simultaneously on the receptive neuron. In an adult brain, plasticity may be accounted for by two mechanisms: the creation of new synaptic connections between neurons, and the modification of existing synapses. Axons, the transmission lines and dendrites which is the receptive zones, constitute two types of cell filaments that are distinguished on morphological grounds; an axon which has a smoother surface, fewer branches, and greater length, whereas a dendrite has an irregular surface and more branches. Neurons come in a wide variety of shapes and sizes in different parts of the brain. Fig. 1.2 illustrates the shape of a pyramidal cell, which is one of the most common types of cortical neurons. Like many other types of neurons, it receives most of the inputs through dendritic spines. The pyramidal cell can receive 10,000 or more synaptic contacts and it can project onto thousands of target cells.

The axon, which branches into collaterals, receives signals from the cell body and carries them away through the synapse (a microscopic gap) to the dendrites of neighboring neurons. A schematic illustration of the signal transfer between two neurons through the synapse is shown in

Fig.1.3b. An impulse, in the form of an electric signal, travels within the dendrites and through the cell body towards the pre-synaptic membrane of the synapse.



*Fig. 1.3 (a) Schematic of biological neuron. (b) Mechanism of signal transfer between two biological neuron*

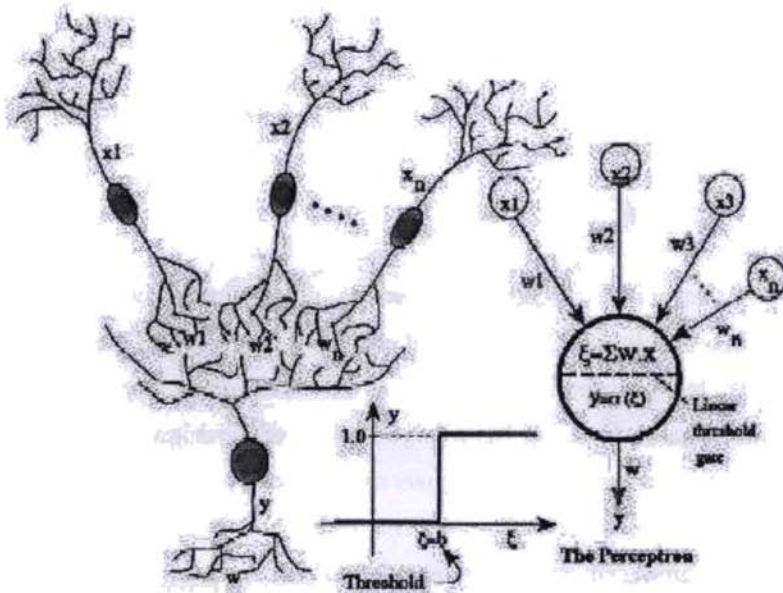


Upon arrival at the membrane, neurotransmitters (chemical like) are released from the vesicles in quantities proportional to the strength of the incoming signal. The neurotransmitters diffuse within the synaptic gap towards the post-synaptic membrane, and eventually into the dendrites of neighbouring neurons, thus forcing them (depending on the threshold of the receiving neuron) to generate a new electrical signal. The generated signal passes through the second neuron(s) in a manner identical to that just described.

The amount of signal that passes through a receiving neuron depends on the intensity of the signal emanating from each of the feeding neurons, their synaptic strengths, and the threshold of the receiving neuron. Because a neuron has a large number of dendrites /synapses, it can receive and transfer many signals simultaneously. These signals may either assist (excite) or inhibit the firing of the neuron depending on the type of neurotransmitters are released from the tip of the axons. This simplified mechanism of signal transfer constituted the fundamental step of early neurocomputing development (e.g., the binary threshold unit of McCulloch and Pitts, 1943) and the operation of the building unit of ANNs.

The crude analogy between artificial neuron and biological neuron is that the connections between nodes represent the axons and dendrites, the connection weights represent the synapses, and the threshold approximates the activity in the soma (Jain et. al., 1996). Fig.1.4 illustrates  $n$  biological neurons with various signals of intensity  $x$  and synaptic strength  $w$  feeding into a neuron with a threshold of  $b$ , and the equivalent artificial neurons system. Both the biological network and

ANN learn by incrementally adjusting the magnitudes of the weights or synaptic strengths (Zupan and Gasteiger, 1993).



*Fig. 1.4 Signal interaction from n neurons and analogy to signal summing in an artificial neuron comprising the single layer perceptron*

### 1.3. Models of a Neuron

In 1958, Rosenblatt introduced the mechanics of the single artificial neuron and introduced the 'Perceptron' to solve problems in the area of character recognition (Hecht-Nielsen, 1990). Basic findings from the biological neuron operation enabled early researchers (e.g., McCulloch and Pitts, 1943) to model the operation of simple artificial neurons. An artificial processing neuron receives inputs as stimuli from the

environment, combines them in a special way to form a 'net' input, passes that over through a linear threshold gate, and transmits the (output,  $y$ ) signal forward to another neuron or the environment, as shown in Fig. 1.4. Only when the net input exceeds (i.e., is stronger than) the neuron's threshold limit (also called bias,  $b$ ), will the neuron fire (i.e, becomes activated). Commonly, linear neuron dynamics are assumed for calculating net input (Haykin, 2003). The net input is computed as the inner (dot) product of the input signals ( $x$ ) impinging on the neuron and their strengths ( $w$ ) (Basheer and Hajmeer, 2000).

In the context of computation, a neuron is pictured as an information-processing unit that is fundamental to the operation of a neural network. The block diagram sketched in Fig.1.5 represents the model of a neuron, which forms the basis for designing artificial neural networks. There are three basic elements in the neuronal model:

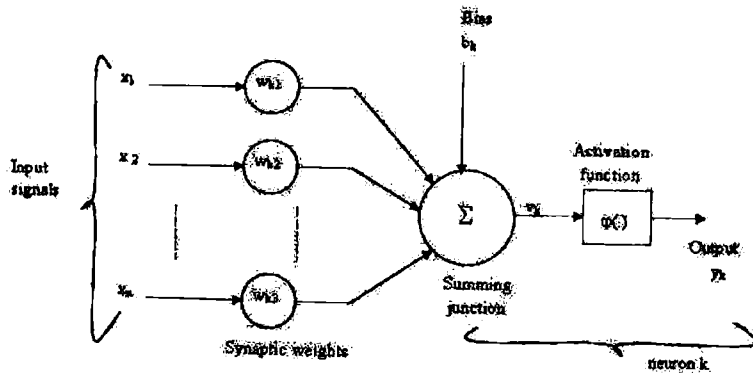


Fig. 1.5 Block diagram representing the Nonlinear model of a neuron

$$\text{Input signal } \vec{X} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ x_n \end{pmatrix} \quad \text{weight factor } \vec{W} = \begin{pmatrix} w_{k1} \\ w_{k2} \\ w_{k3} \\ \vdots \\ \vdots \\ w_{kn} \end{pmatrix}$$

Net output for the  $k^{th}$  neuron is:

$$v_k = X^T W = x_1 w_{k1} + x_2 w_{k2} + x_3 w_{k3} + \dots + x_n w_{kn} \quad (1.1)$$

- A set of synapses or connecting links, each of which is characterized by a weight or strength of its own. Specifically, a signal  $x_j$  at the input of synapse  $j$  connected to neuron  $k$  is multiplied by the synaptic weight  $w_{kj}$ .
- An adder for summing the input signals, weighted by the respective synapses of the neuron; the operations described here constitutes a linear combiner.
- An activation function for limiting the amplitude of the neuron output. The activation is also referred to as a squashing function or limiting function in that it squashes (limits) the permissible amplitude range of the output signal to some finite value. Typically, the normalized amplitude range of the output of a neuron is written as the closed unit interval  $[0,1]$  or alternatively  $[-1,1]$  representing unipolar and bipolar cases respectively.

The neuron model of Fig.1.5 also includes an externally applied bias, denoted by  $b_k$ . The bias  $b_k$  has the effect of increasing or lowering the net

input of the activation function. depending on whether it is positive or negative respectively.

In mathematical terms, we may describe a neuron  $k$  by writing the following pair of equations:

$$u_k = \sum_{j=1}^m w_{kj} x_j \quad (1.2)$$

and

$$y_k = \varphi (u_k + b_k) \quad (1.3)$$

where  $x_1, x_2, \dots, x_m$  are the input signals;  $w_{k1}, w_{k2}, \dots, w_{km}$  are the synaptic weights of neuron  $k$ ;  $u_k$  is the linear combiner output due to the input signals;  $b_k$  is the bias;  $\varphi (\cdot)$  is the activation function; and  $y_k$  is the output signal of the neuron. The use of the bias  $b_k$  has the effect of applying an affine transformation to the output  $u_k$  of the linear combiner in the model of Fig. 1.5 as shown by

$$v_k = u_k + b_k \quad (1.4)$$

In particular, depending on whether the bias  $b_k$  is positive or negative, the relationship between the induced local field or activation potential  $v_k$  of neuron  $k$  and the linear combiner output  $u_k$  is modified in the manner illustrated in Fig.1.6. The bias  $b_k$  is an external parameter of artificial neuron  $k$  and is an important parameter in describing the dynamics of the neuron.

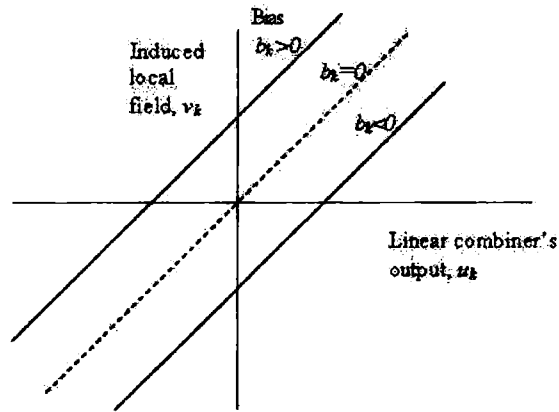


Fig. 1.6 Affine transformation produced by the presence of a bias

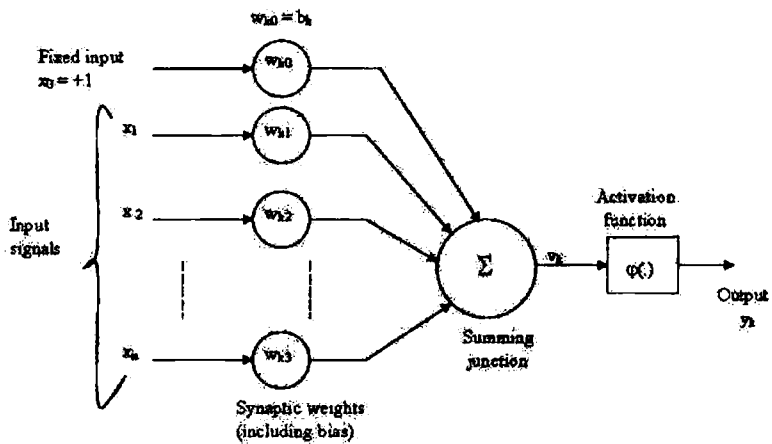


Fig. 1.7 Another Nonlinear model of a neuron including the effect of bias accounted as a input signal fixed at +1.

Combinations of Eqs. (1.2) and (1.4) as follows:

$$v_k = \sum_{j=0}^m w_{kj} x_j \quad (1.5)$$

and

$$y_k = \varphi(v_k) \quad (1.6)$$

In Eq.(1.4) a new synapse is added. Its input is

$$x_0 = +1 \quad (1.7)$$

and its weight is

$$w_{k0} = b_k \quad (1.8)$$

Therefore the model of the neuron  $k$  is reformulated as in Fig. 1.7. In this Figure, the effect of the bias is accounted as: adding a new input signal fixed at +1, and adding a new synaptic weight equal to the bias  $b_k$

## 1.4 Types of activation functions

The activation function may be a linear or a nonlinear function. The activation function, denoted by  $\varphi(v)$ , defines the output of a neuron in terms of the induced local field  $v$ . The activation function generates either unipolar or bipolar signals. In the following sections various types of function used for activating the neuron activities are described.

### 1.4.1 A step function

It is a unipolar function and is also referred to as a threshold function. This function is shown in Fig. 1.8(a) and is defined as:

$$\varphi(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{if } v < 0 \end{cases} \quad (1.9)$$

In engineering literature, this is a threshold function referred to as Heaviside function. Correspondingly, the output of neuron  $k$  employing such a threshold function is expressed as

$$y(k) = \begin{cases} 1 & \text{if } v_k \geq 0 \\ 0 & \text{if } v_k < 0 \end{cases} \quad (1.10)$$

where  $v_k$  is the induced local field of the neuron; so that

$$v_k = \sum_{j=1}^m w_{kj} x_j + b_k \quad (1.11)$$

Eqn.1.11 represents a neuron referred to in the literature as the McCulloch-Pitts model, in recognition of the pioneering work done by McCulloch and Pitts (1943). In this model, the output of the neuron takes on the value of 1 if the induced local field of that neuron is nonnegative, and 0 otherwise. This statement describes the all-or-none property of the McCulloch-Pitts model.

### 1.4.2 Piecewise-Linear Function

This is also a unipolar function. The piecewise-linear function described in Fig.1.8(b) is defined as:



$$\varphi(v) = \begin{cases} 1, & v \geq +\frac{1}{2} \\ v, & +\frac{1}{2} > v > -\frac{1}{2} \\ 0, & v \leq -\frac{1}{2} \end{cases} \quad (1.12)$$

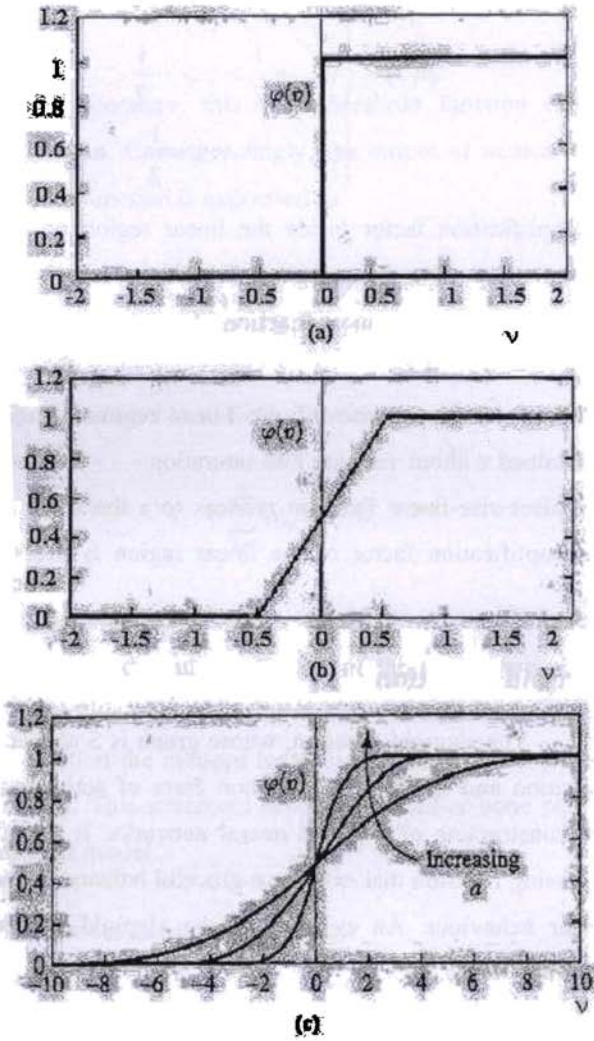
where the amplification factor inside the linear region of operation is assumed to be unity. The following two situations may be viewed as special form of the piecewise-linear function:

- A linear combiner arises if the linear region of operation is maintained without running into saturation
- The piecewise-linear function reduces to a threshold function if the amplification factor of the linear region is made infinitely large.

### 1.4.3 Sigmoid Function

The sigmoid function, whose graph is S shaped, is also a unipolar function and is the most common form of activation function used in the construction of artificial neural networks. It is defined as a strictly increasing function that exhibits a graceful balance between linear and nonlinear behaviour. An example of the sigmoid function is the logistic function, defined by

$$\varphi(v) = \frac{1}{1 + \exp(-av)} \quad (1.13)$$



**Fig. 1.8** Various types of activation functions (a) step function (b) piece-wise linear function (c) sigmoid function

where  $a$  is the slope parameter of the of the sigmoid function.

By varying the parameter  $a$ , sigmoid functions of different slopes are obtained, as illustrated in Fig.1.8(c). In fact, the slope at the origin equals  $a/4$ . In the limit, as the slope parameter approaches infinity, the sigmoid function becomes simply a threshold function. Whereas a threshold function assumes the value of 0 or 1, a sigmoid function assumes a continuous range of values from 0 to 1. Moreover the sigmoid function is differentiable, unlike in the case of other threshold functions. Differentiability is an important feature of neural network theory.

All the above mentioned activation functions are unipolar, which are varying between 0 and 1. It is sometimes desirable to have the activation function range from -1 to +1, in which case the activation function assumes an antisymmetric form with respect to the origin; that is, the activation function is an odd function of the induced local field. Specifically, the threshold function of Eq.(1.9) is now defined as

$$\phi(v) = \begin{cases} 1, & \text{if } v > 0 \\ 0, & \text{if } v = 0 \\ -1, & \text{if } v < 0 \end{cases} \quad (1.14)$$

which is commonly referred to as the signum function. For the corresponding form of the sigmoid function, the hyperbolic tangent function is used, which is defined by:

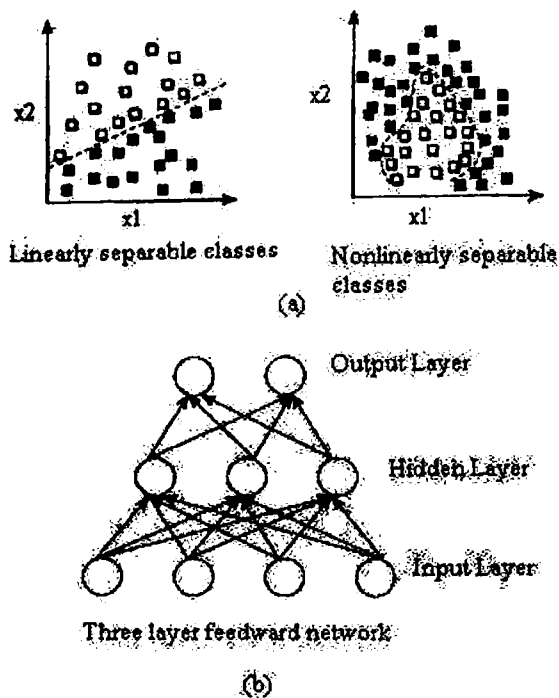
$$\phi(v) = \tanh(v) \quad (1.15)$$

## 1.5 Perceptrons

The perceptron (Fig.1.7) can be trained on a set of examples using a special learning rule (Hecht-Nielsen, 1990). The perceptron weights (including the threshold) are changed in proportion to the difference (error) between the target (correct) output,  $Y$ , and the perceptron solution,  $y$ , for each example. The error is a function of all the weights and forms an irregular multidimensional complex hyperplane with many peaks, saddle points, and minima. Using a specialized search technique, the learning process strives to obtain the set of weights that corresponds to the global minimum. Rosenblatt (1962) derived the perceptron rule that will yield an optimal weight vector in a finite number of iterations, regardless of the initial values of the weights.

This rule, however, can perform accurately with any linearly separable classes (Hecht-Nielsen, 1990), in which a linear hyperplane can place one class of objects on one side of the plane and the other class on the other side. Fig. 1.9 (a) shows linearly and nonlinearly separable two-object classification problems. In order to cope with nonlinearly separable problems, additional layer(s) of neurons placed between the input layer (containing input nodes) and the output neuron are needed leading to the multilayer perceptron (MLP) architecture (Hecht-Nielsen, 1990), as shown in Fig. 1.9 (b). Since these intermediate layers do not interact with the external environment, they are called hidden layers and their nodes called hidden nodes. The addition of intermediate layers revived the perceptron model by extending its ability to solve nonlinear classification problems. Using similar neuron dynamics, the hidden neurons process the

information received from the input nodes and pass them over to output layer.



**Fig. 1.9. (a) Linear vs. nonlinear separability. (b) Multilayer perceptron showing input, hidden, and output layers and nodes with feedforward links.**

The learning of MLP is not as direct as that of the simple perceptron. For example, the backpropagation network (Rumelhart et al., 1986) is one type of MLP trained by the delta learning rule (Zupan and Gasisteiger, 1993). However, the learning procedure is an extension of the simple perceptron algorithm so as to handle the weights connected to the hidden nodes (Hecht-Nielsen, 1990).

## 1.6 Learning Processes

Learning is a process by which the free parameters of a neural network are adapted through a process of simulation by the environment in which the network is embedded. The type of learning is determined by the manner in which the parameter changes take place. The above definition of learning process implies the following sequence of events:

- The neural network is stimulated by an environment.
- The neural network undergoes changes in its free parameters as a result of this simulation.
- The neural network responds in a new way to the environment because of the changes that have occurred in its internal structure.

A prescribed set of well-defined rules for the solution of a learning problem is called learning algorithm. As one would expect, there is no unique learning algorithm for the design of neural networks. Rather, there is kit of tools represented by a diverse variety of learning algorithms, each of which offers advantages of its own. Basically, learning algorithms differ from each other in the way in which the adjustment to a synaptic weight of a neuron is formulated. Another factor to be considered is the manner in which a neural network, made up of a set of interconnected neurons, relates to its environment.

Hebb's postulate of learning is the oldest and the most famous of all learning rules; it is named in honour of the neuropsychologist Hebb (1949). His postulate states that:

*When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic changes take place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.*

Hebb proposed this change as a basis of associative learning which would result in an enduring modification in the activity pattern of a spatially distributed assembly of nerve cells.

The Hebb's postulate can be expanded and rephrased as a two-part rule:

- If two neurons on either side of a synapse are activated simultaneously, then the strength of that synapse is selectively increased.
- If two neurons on either side of a synapse are activated asynchronously, then that synapse is selectively weakened or eliminated.

Such a synapse is called a Hebbian synapse. More precisely, a Hebbian synapse is a synapse that uses a time-dependent, highly local and strongly interactive mechanism to increase synaptic efficiency as a function of the correlation between the presynaptic and postsynaptic activities.

## **1.7 A Brief History**

In this section, in order to make the thesis self-contained an overview of the historical evolution of ANNs and neurocomputing is briefly presented. Anderson and Rosenfeld (1988) provide a detailed history

along with a collection of the many major classic papers that affected ANNs evolution. Nelson and Illingworth (1990) divide 100 years of history into six notable phases: (1) Conception, 1890–1949; (2) Gestation and Birth, 1950s; (3) Early Infancy, late 1950s and the 1960s; (4) Stunted Growth, 1961– 1981; (5) Late Infancy I, 1982–1985; and (6) Late Infancy II, 1986–present. The era of conception includes the first development in brain studies and the understanding of brain mathematics. It is believed that the year 1890 was the beginning of the neurocomputing age in which the first work on brain activity was published by William James (Nelson and Illingworth, 1990). Many (e.g., Hecht-Nielsen, 1990) believe that real neurocomputing started in 1943 after McCulloch and Pitts (1943) paper on the ability of simple neural networks to compute arithmetic and logical functions. This era ended with the book ‘The Organization of Behavior’ by Donald Hebb in which he presented his learning law for the biological neurons’ synapses (Hebb, 1949). The work of Hebb is believed to have paved the road for the advent of neurocomputing (Hecht- Nielsen, 1990).

The gestation and birth era began following the advances in hardware/software technology which made computer simulations possible and easier. In this era, the first neurocomputer (the Snark) was built and tested by Minsky at Princeton University in 1951, but it experienced many limitations (Hecht-Nielsen, 1990). This era ended by the development of the Dartmouth Artificial Intelligence (AI) research project which laid the foundations for extensive neurocomputing research (Nelson and Illingworth, 1990).

The era of early infancy began with John von Neuman’s work which was published a year after his death in a book entitled ‘The Computer and



the Brain' (von Neuman, 1958). In the same year, Frank Rosenblatt at Cornell University introduced the first successful neurocomputer (the Mark I perceptron), designed for character recognition which is considered nowadays the oldest ANN hardware (Nelson and Illingworth, 1990). Although the Rosenblatt perceptron was a linear system, it was efficient in solving many problems and led to what is known as the 1960s ANNs hype. In this era, Rosenblatt also published his book 'Principles of Neurodynamics' (Rosenblatt, 1962). The neurocomputing hype, however, did not last long due to a campaign led by Minsky and Pappert (1969) aimed at discrediting ANNs research to redirect funding back to AI. Minsky and Pappert published their book 'Perceptrons' in 1969 in which they over exaggerated the limitations of the Rosenblatt's perceptron as being incapable of solving nonlinear classification problems, although such a limitation was already known (Hecht-Nielsen, 1990; Wythoff, 1993). Unfortunately, this campaign achieved its planned goal, and by the early 1970s many ANN researchers switched their attention back to AI, whereas a few 'stubborn' others continued their research. Hecht-Nielsen (1990) refers to this era as the 'quiet years' and the 'quiet research'.

With the Rosenblatt perceptron and the other ANNs introduced by the 'quiet researchers', the field of neurocomputing gradually began to revive and the interest in neurocomputing renewed. Nelson and Illingworth (1990) list a few of the most important research studies that assisted the rebirth and revitalization of this field, notable of which is the introduction of the Hopfield networks (Hopfield, 1984), developed for retrieval of complete images from fragments. The year 1986 is regarded a cornerstone in the ANNs recent history as Rumelhart et al.(1986)

rediscovered the backpropagation learning algorithm after its initial development by Werbos (1974). The first physical sign of the revival of ANNs was the creation of the Annual IEEE International ANNs Conference in 1987, followed by the formation of the International Neural Network Society (INNS) and the publishing of the INNS Neural Network journal in 1988. It can be seen that the evolution of neurocomputing has witnessed many ups and downs, notable among which is the period of hibernation due to the perceptron's inability to handle nonlinear classification. Since 1986, many ANN societies have been formed, special journals published, and annual international conferences organized. At present, the field of neurocomputing is blossoming almost daily on both the theory and practical application fronts.

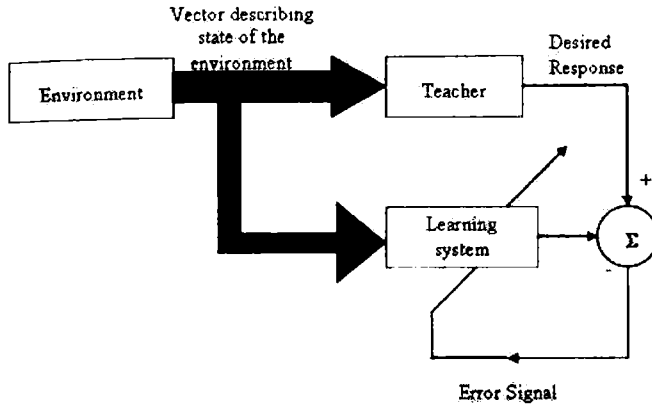
## **1.8 Learning Rules**

A learning rule is a procedure to modify the weight and biases of a network. This is also referred to as a training algorithm. The purpose of the learning rule is to train the network to perform some task. There are many types of neural network learning rules. They fall into three broad categories: supervised learning, unsupervised learning and reinforcement learning.

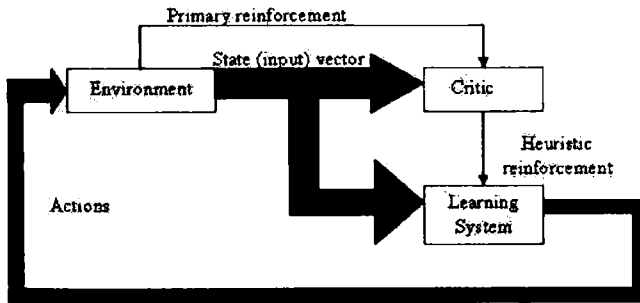
In supervised learning, the learning rule is provided with a set of examples (the training set) of proper network behaviour:

$$[x_1, t_1], [x_2, t_2], [x_3, t_3], \dots, [x_q, t_q], \quad (1.16)$$

where  $x_q$  is an input to the network and  $t_q$  is the corresponding correct (target) output.



(a)



(b)

**Fig.1.10 (a) Block diagram of learning with a teacher (b) Block Diagram of reinforcement learning**

As the inputs are applied to the network, the network outputs are compared to the targets (Hagan et. al., 2002). The learning rule is then used to adjust the weight and biases of the network in order to move the network outputs closer to the targets. This kind of learning is also known as learning with a teacher. Fig.1.10 (a) shows a block diagram that illustrates this form of learning. Suppose now that the teacher and the neural network are both exposed to a training vector drawn from the environment. By the virtue of built-in-knowledge, the teacher is able to provide the neural network with a desired response for that training vector. Indeed, the desired response represents the optimum action to be performed by the neural network. The network parameters are adjusted under the combined influence of the training vector and the error signal. The error signal is defined as the difference between the desired signal and the actual response of the network. This adjustment is carried out iteratively in a step-by-step fashion with the aim of eventually making the neural network emulate the teacher; the emulation is presumed to be optimum in some statistical sense. In this way knowledge of the environment available to the teacher is transferred to the neural network through training as fully as possible. When this condition is reached, the teacher is dispensed with and let the neural network deal with the environment completely by itself.

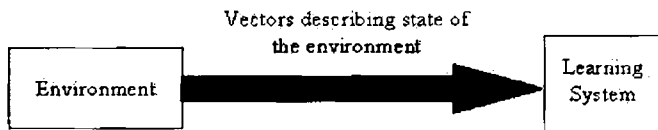
In supervised learning, the process takes place under the tutelage of a teacher. But, in the paradigm known as learning without a teacher, there is no teacher to oversee the learning process. That is to say, there are no labelled examples of the function to be learned by the network. Under this

paradigm, two subdivisions are identified: one is the reinforcement learning and the other unsupervised learning

In reinforcement learning, the learning of an input-output mapping is performed through continued interaction with the environment in order to minimize the scalar index of performance. Fig. 1.10 (b) shows the block diagram of one form of a reinforcement learning system built around a critic that converts a primary reinforcement signal received from the environment into a higher quality reinforcement signal called the heuristic reinforcement signal, both of which are scalar inputs. The system is designed to learn under delayed reinforcement, which means that the system observes a temporal sequence of state vectors also received from the environment, which eventually result in the generation of the heuristic reinforcement signal. The goal of learning is to minimize a cost-to-go function, defined as the expectation of the cumulative cost of actions taken over a sequence of steps instead of simply the immediate cost. It may turn out that certain actions taken earlier in that sequence of time steps are in fact the best determinants of overall system behaviour. The function of the learning machine, which constitutes the second component of the system, is to discover these actions and to feed them back to the environment.

In unsupervised learning or self-organized learning there is no external teacher or critic to oversee the learning process, as indicated in Fig. 1.11. Rather, provision is made for a task independent measure of the quality of representation that the network is required to learn, and the free parameters of the network are optimized with respect to that measure. Once the network has become tuned to the statistical regularities of the

input data, it develops the ability to form internal representations for encoding features of the input and thereby to create new classes automatically. To perform unsupervised learning, a competitive learning rule is used. For that a neural network consisting of two layers- an input layer and a competitive layer are employed. The input layer receives the available data. The competitive layer consists of neurons that compete with each other (in accordance with a learning rule) for the opportunity to respond to features contained in the input data. In its simplest form, the network operates in accordance with a winner-takes-all strategy.



*Fig. 1.11 Block diagram of unsupervised learning*

## 1.9 Learning Algorithms

In the formative years of the neural network (1943-1958), several researchers stand out for their pioneering contributions:

- McCulloch and Pitts (1943) for introducing the idea of neural network as computing machines.
- Hebb (1949) for postulating the first rule for self-organised learning.
- Rosenblatt (1958) for proposing the perceptron as the first model for supervised learning.

In the present research work only supervised learning is used. Therefore in the following sections only supervised learning algorithms are dealt with. The perceptron is the simplest form of a neural network used for the classification of patterns which are said to be linearly separable. Basically, it consists of a single neuron with adjustable synaptic weights and bias. Indeed, Rosenblatt proved that if the patterns used to train the perceptron are drawn from two linearly separable classes, then the perceptron algorithm converges and positions the decision surface in the form of a hyper plane between the two classes (Rosenblatt, 1962). The proof of convergence of the algorithm is known as the perceptron convergence theorem. The perceptron built around a single neuron is limited to performing pattern classification with only two classes. In the following sections the Perceptron algorithm and the backpropagation algorithms are given.

### 1.9.1 The Perceptron Algorithm

The Perceptron, an invention of Rosenblatt (1962), was one of the earliest neural network models. A perceptron models a neuron by taking a weighted sum of the inputs and sending the output 1 if the sum is greater than some adjustable threshold value (otherwise it sends 0). Fig. 1.12 shows the device.

The inputs ( $x_1, x_2, x_3, \dots, x_n$ ) and connection weights ( $w_1, w_2, w_3, \dots, w_n$ ) in the Figure are typically real values, both positive and negative. If the presence of some feature  $x_i$  tends to cause the perceptron to fire, the weight  $w_i$  will be positive; if the feature  $x_i$  inhibits the perceptron, the weight  $w_i$  will be negative. The perceptron itself consists

of the weights, the summation processor, and the adjustable threshold processor. Learning is a process of modifying the values of the weights and the thresholds (bias). It is convenient to implement the bias as just another weight  $w_0$ . This weight can be thought of as the propensity of the perceptron to fire irrespective of its inputs (Rich and Knight, 1994).

**Step 0**            *Set up the neural network model as shown in Fig.1.12*

*Initialize the weights and bias.*

*Set learning rate,  $\eta$  ( $0 < \eta < 1$ )*

*Set minimum error value for stopping.*

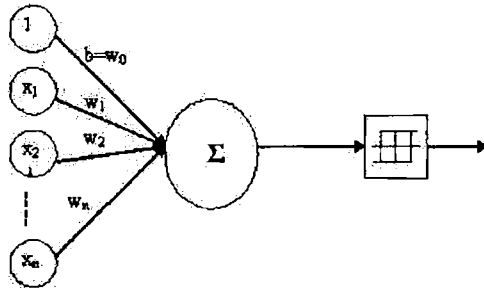
**Step 1**            *While the stopping condition is false, do steps 2 – 6*

**Step 2**            *For each training pair  $u:t$  do steps 3 – 5*

**Step 3**            *Set activations of input units:  $x_i = u_i, i = 1, 2, 3, \dots, n$*

**Step 4**            *Compute the response of output unit:*

$$s = b + \sum_i x_i w_i \quad (1.17)$$



**Fig. 1.12 A Perceptron neuron model**

**Step 5**            *Update weights and bias*



$$\begin{aligned}w_i(\text{new}) &= w_i(\text{old}) + \eta(t-s)x_i \\ b(\text{new}) &= b(\text{old}) + \eta(t-s)\end{aligned}\tag{1.18}$$
$$i = 1, 2, 3, \dots, n$$

**Step 6**      *Test for stopping condition:*

*If the largest weight change that occurred in step is smaller than a specified tolerance, then stop; else continue.*

## 1.9.2 The Backpropagation Algorithm

Fig. 1.13 shows a fully connected, layered, feedforward network. In this Figure, weights on connections between the input and hidden layers are denoted by  $w_i$ , while weights on connections between the hidden and output layers are denoted by  $w_h$ . This network has three layers, although it is possible and sometimes useful to have more. Each unit in one layer is connected in the forward direction to every unit in the next layer. Activations flow from the input layer through the hidden layer and then on to the output layer. The knowledge of the network is encoded in the weights on connection between units. The existence of hidden units allows the network to develop complex feature detectors, or internal representations.

The units in a backpropagation network require a slightly different activation function from the perceptron. A backpropagation unit will sum up its weighted inputs, but unlike the perceptron, it produces a real value between 0 and 1 as output based on a sigmoid function, which is continuous and differentiable, as required by the backpropagation

algorithm. Like a perceptron, a backpropagation network typically starts out with a random set of weights (Rich and Knight, 1994).

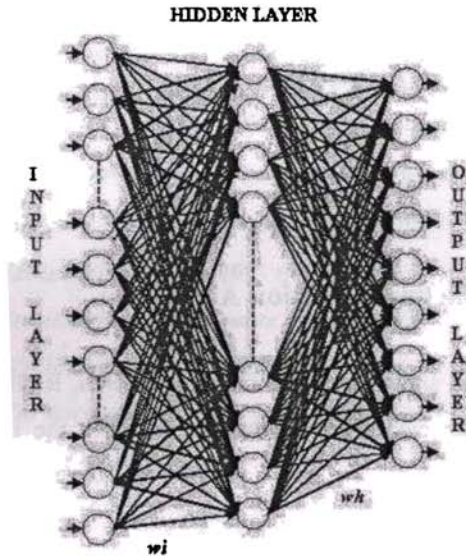


Fig. 1.13 A feed forward neural network  $w_i$  is the weight of the input layer to the hidden layer,  $w_h$  is the weight of the hidden layer to the output layer.

**Given:** A set of input-output ( $x:y$ ) vector pairs.

**Compute:** A set of weights for a three layer network that maps inputs onto corresponding outputs. ( $w_i$  is the weight of the input layer to the hidden layer,  $w_h$  is the weight of the hidden layer to the output layer)

**Step 1** Let  $A$  be the number of units in the input layer,  $B$  be the number of units in the hidden layer,  $C$  be the number of units in the output layer.

**Step 2**  $w_{i,j} = \text{random}(-0.1, 0.1)$  for all  $i = 0 \dots \dots \dots A, j = 1 \dots \dots \dots B$

$wh_{i,j} = \text{random}(-0.1, 0.1)$  for all  $i = 0 \dots \dots \dots B, j = 1 \dots \dots \dots C$

**Step 3** *Initialize the activation of the threshold units. The values of the threshold units should never change. Set the learning rate,  $\eta$*

**Step 4** *Choose an input-output pair. Assign activation levels to the input units.*

**Step 5** *Propagate the activations from the units in the input layer to the units in the hidden layer using the activation function*

$$h_j = \frac{1}{1 + e^{-\sum_{i=0}^A w_{i,j} x_i}} \quad \text{for all } j = 1 \dots \dots B \quad (1.19)$$

**Step 6** *Propagate the activations from the units in the hidden layer to the units in the output layer*

$$o_j = \frac{1}{1 + e^{-\sum_{i=0}^H w_{h_i,j} h_i}} \quad \text{for all } j = 1 \dots \dots C \quad (1.20)$$

**Step 7** *Compute the errors of the units in the output layer,  $\delta_{2j}$*

$$\delta_{2j} = o_j (1 - o_j) (y_j - o_j) \quad \text{for all } j = 1 \dots \dots C \quad (1.21)$$

**Step 8** *Compute the errors of the units in the hidden layer,  $\delta_{1j}$*

$$\delta_{1j} = h_j (1 - h_j) \sum_{i=1}^C \delta_{2i} w_{h_j,i} \quad \text{for all } j = 1 \dots \dots B \quad (1.22)$$

**Step 9** *Adjust the weights  $w_i$  and  $w_h$*

$$\Delta w h_{i,j} = \eta \delta_{2j} h_i$$

*for all  $i = 0 \dots B, j = 1 \dots C$*  (1.23)

$$\Delta w i_{i,j} = \eta \delta_{1j} x_i$$

*for all  $i = 0 \dots A, j = 1 \dots B$*  (1.24)

**Step 10** *Go to step 4 and repeat. When all the input-output pairs have been presented to the network, one epoch has been completed. Repeat steps 4 to 10 for as many epochs as desired.*

The algorithm generalizes straightforwardly to networks of more than three layers. For each extra hidden layer, insert a forward propagation step between steps 6 and 7, an error computation step between steps 8 and 9, and a weight adjustment step between steps 10 and 11. Error computation for hidden units should use the equation in step 8, but with  $i$  ranging over the units in the next layer, not necessarily the output layer (Rich and Knight, 1994).

## 1.10 Digital Image Processing:

### 1.10.1 Image Representation, Sampling, Quantization

An image may be defined as a two-dimensional function,  $f(x,y)$ , where  $x$  and  $y$  are spatial coordinates, and the amplitude of  $f$  at any pair of coordinates  $(x,y)$  is called the intensity or gray level of the image at that point. When  $x,y$  and the amplitude values of  $f$  are all finite, discrete quantities, it is called as a digital image. The field of digital image

processing refers to processing digital images by a digital computer. A digital image is composed of a finite number of elements, each of which has a particular location and value. These elements are referred to as picture elements, image elements, pels and pixels. In general, the fundamental steps in digital image processing consist of components like image acquisition, image enhancement, image restoration etc

Image acquisition is the first process. Note that acquisition could be as simple as being given an image that is already in digital form. Generally, the image acquisition stage involves preprocessing, such as scaling. The types of images in which we are interested are generated by the combination of an illumination source and the reflection or absorption of energy from that source by the elements of the scene being imaged.

When an image is generated from a physical process, its values are proportional to energy radiated by a physical source. As a consequence,  $f(x,y)$  must be nonzero and finite; that is,

$$0 < f(x,y) < \infty \quad (1.25)$$

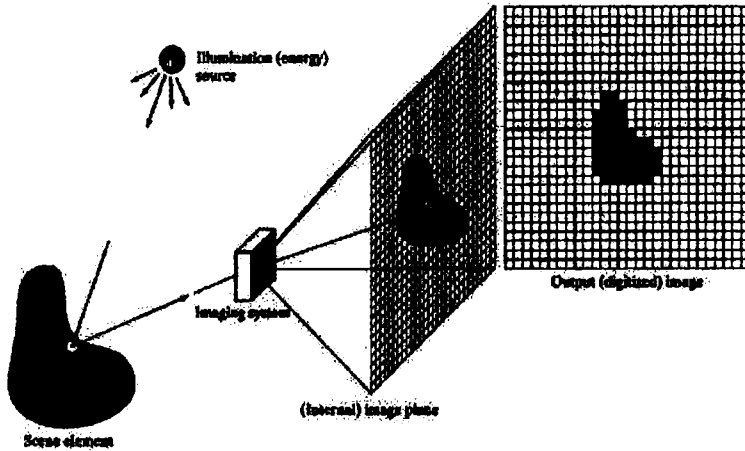
The function  $f(x,y)$  may be characterized by two components: (1) the amount of source illumination incident on the scene being viewed, and (2) the amount of illumination reflected by the objects in the scene. Appropriately, these are the illumination and reflectance components and are denoted by  $i(x,y)$  and  $r(x,y)$  respectively. The two functions combine as a product to form  $f(x,y)$ :

$$f(x,y) = i(x,y)r(x,y) \quad (1.26)$$

where

$$0 < i(x, y) < \infty \quad (1.27)$$

and



*Fig. 1.14 An example of the digital image acquisition process.*

$$0 < r(x, y) < 1 \quad (1.28)$$

Eq. (1.28) indicates that reflectance is bounded by 0 (total absorption) and 1 (total reflectance). The nature of  $i(x, y)$  is determined by the illumination source, and  $r(x, y)$  is determined by the characteristics of the imaged objects.

The intensity of a monochrome image at any coordinates  $(x_n, y_n)$  determines the gray level ( $I$ ) of the image at that point. That is,

$$I = f(x_0, y_0) \quad (1.29)$$

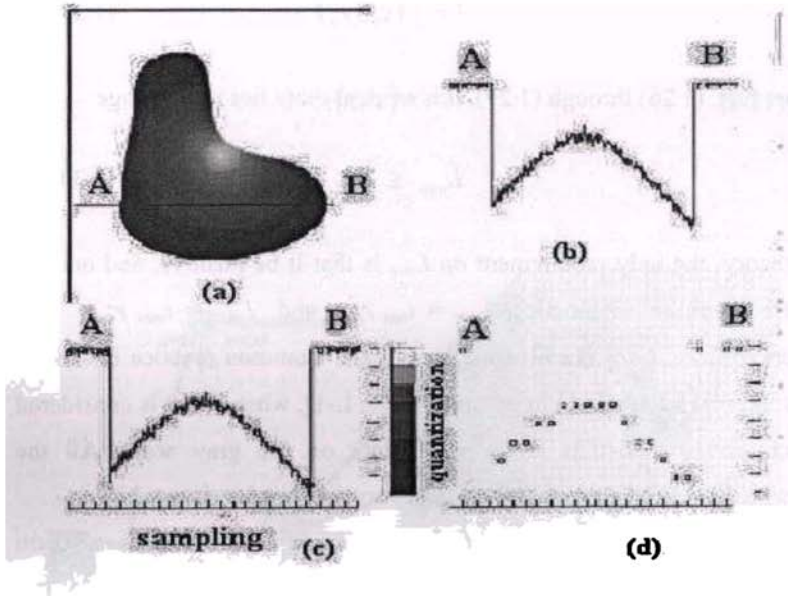
From Eqs. (1.26) through (1.29), it is evident that  $I$  lies in the range

$$L_{\min} \leq I \leq L_{\max} \quad (1.30)$$

In theory, the only requirement on  $L_{\min}$  is that it be positive, and on  $L_{\max}$  that it be finite. In practice,  $L_{\min} = i_{\min} r_{\min}$  and  $L_{\max} = i_{\max} r_{\max}$ . The interval  $[L_{\min}, L_{\max}]$  is called the gray scale. Common practice is to shift this interval numerically to the interval  $[0, L-1]$ , where  $I = 0$  is considered black and  $I = L-1$  is considered white on the gray scale. All the intermediate values are shades of gray varying from black to white.

The output of most sensors is a continuous voltage waveform whose amplitude and spatial behaviour are related to the physical phenomenon to be sensed. To create digital image, the continuous sensed data should be converted to digital form. This involves two processes: sampling and quantization. The basic idea behind sampling and quantization is illustrated in Fig.1.15. Fig. 1.15 (a) shows a continuous image,  $f(x,y)$ , that is to be converted to digital form. An image may be continuous with respect to the  $x$ - and  $y$ - coordinates, and also in amplitude. To convert to digital form, the function has to be sampled in both coordinates and in amplitude. Digitizing the coordinate values is called sampling. Digitizing the amplitude values is called quantization.

The one-dimensional function shown in Fig. 1.15 (b) is a plot of amplitude values (gray level) of the continuous image along the line segment AB in Fig. 1.15 (a). The random variations are due to image



**Fig.1.15** *Generating a digital image. (a) Continuous image (b) A scan line from A to B in the continuous image (c) sampling and quantization (d) digital scan line*

noise. To sample this function, equally spaced samples along line AB as shown in Fig. 1.15(c) are taken. The location of each sample is given by a vertical tick mark in the bottom part of the Figure. The samples are shown as small white squares superimposed on the function. The set of these discrete locations gives the sampled function. However, the values of the samples still span (vertically) a continuous range of gray-level values. In order to form a digital function, the gray level values also must be converted (quantized) into discrete quantities. The right side of Fig.1.15(c) shows the gray-level scale divided into eight discrete levels,



ranging from black to white. The vertical tick marks indicate the specific value assigned to each of the eight gray levels. The continuous gray levels quantized simply by assigning one of the eight discrete gray levels to each sample. The assignment is made depending on the vertical proximity of a sample to a vertical tick mark. The digital samples resulting from both sampling and quantization are shown in Fig.1.15 (d). Starting at the top of the image and carrying out this procedure line by line produces a two-dimensional digital image.

The result of sampling and quantization is a matrix of real numbers. Assume that an image  $f(x,y)$  is sampled so that the resulting digital image has  $M$  rows and  $N$  columns. The values of the coordinates  $(x,y)$  now become discrete quantities. The complete  $M \times N$  digital image can be written in a compact matrix form as:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & \dots & f(1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(M-1,0) & f(M-1,1) & \dots & f(M-1,N-1) \end{bmatrix} \quad (1.31)$$

The right side of the equation is by definition is a digital image. Each element of this matrix array is called an image element, picture element, pixel or pel.

The digitization process requires decision about the values of  $M$ ,  $N$ , and for the number,  $L$ , of discrete gray levels allowed for each pixel. There are no requirements on  $M$  and  $N$  other than that they have to be positive integers. However, due to processing, storage, sampling and hardware considerations, the number of gray levels typically is an integer power of 2:

$$L = 2^k \quad (1.32)$$

It is assumed that the discrete levels are equally spaced and that they are integers in the interval  $[0, L-1]$ . Sometimes the range of value spanned by the gray scale is called the dynamic range of the image, and the images whose gray levels span a significant portion of the gray scale are referred to as those having high dynamic range. When an appreciable number of pixels exhibit this property, the image will have high contrast. Conversely, an image with low dynamic range tends to have a dull washed out gray look.

The number,  $b$ , of bits required to store a digitized image is

$$b = MNk \quad (1.33)$$

When  $M = N$ , this equation becomes

$$b = N^2k \quad (1.34)$$

Sampling is the principal factor determining the spatial resolution of an image. Basically, spatial resolution is the smallest discernible detail in an image. Gray-level resolution refers to the smallest discernible change in gray level (Gonzalez and Woods, 2002).

## **1.11 Various tools for Digital Image Processing**

### **1.11.1 The Two-Dimensional DFT and its Inverse**

Image enhancement is among the simplest and most appealing areas of digital image processing. Basically, the idea behind enhancement

techniques is to bring out detail that is obscured, or simply to highlight features of interest in an image. The main objective of enhancement is to process an image so that the result is more suitable than the original image for a specific application. Image enhancement techniques fall into two broad categories: spatial domain methods and frequency domain methods. The term spatial domain refers to the image plane itself, and approaches in this category are based on direct manipulation of pixels in an image. Frequency domain processing techniques are based on modifying the Fourier transform of an image.

In the present research work, frequency domain spatial enhancement techniques are dealt with. Hence the focus is mostly on a discrete formulation of the Fourier transform. The discrete Fourier transform of a function (image)  $f(x,y)$  of size  $M \times N$  is given by the equation

$$F(u, v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi(ux/M + vy/N)} \quad (1.35)$$

This expression is computed for values of  $u = 0, 1, 2, \dots, M-1$ , and also for  $v = 0, 1, 2, \dots, N-1$ . The inverse Fourier transform is given by the expression:

$$f(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) e^{j2\pi(ux/M + vy/N)} \quad (1.36)$$

For  $x = 0, 1, 2, \dots, M-1$  and  $y = 0, 1, 2, \dots, N-1$ . Equations (1.35) and (1.36) comprise the two-dimensional, Discrete Fourier Transform pair.

The variables  $u$  and  $v$  are the transform or frequency variables, and  $x$  and  $y$  are the spatial or image variables. The location of  $1/MN$  constant in Eqn.1.34 is not important. Sometimes it is located in front of the transform. Other times it is found split into two equal terms of  $1/\sqrt{MN}$  multiplying the transform and its inverse.

The Fourier spectrum, phase angle, and power spectrum are defined as:

$$|F(u, v)| = [R^2(x, y) + I^2(x, y)]^{1/2} \quad (1.37)$$

$$\varphi(u, v) = \tan^{-1} \left[ \frac{I(u, v)}{R(u, v)} \right] \quad (1.38)$$

and

$$P(u, v) = |F(u, v)|^2 = R^2(u, v) + I^2(u, v) \quad (1.39)$$

where  $R(u, v)$  and  $I(u, v)$  are the real and imaginary parts of  $F(u, v)$ , respectively.

It is common practice to multiply the input image function by  $(-1)^{x+y}$  prior to computing the Fourier transform. Due to the properties of exponentials it can be proved that:

$$\mathfrak{F} \left[ f(x, y) (-1)^{x+y} \right] = F(u - M/2, v - N/2) \quad (1.40)$$

where  $\mathfrak{F}[\cdot]$  denotes the Fourier transform of the argument. This equation states that the origin of the Fourier transform of  $f(x, y)(-1)^{x+y}$  is located at  $u = M/2$  and  $v = N/2$ . In other words, multiplying  $f(x, y)$  by  $(-1)^{x+y}$  shifts the origin of  $F(u, v)$  to frequency coordinates  $(M/2, N/2)$ , which is the centre of the  $M \times N$  area occupied by the 2-D DFT. This area of the frequency domain is referred to as the frequency rectangle. It extends from  $u = 0$  to  $u = M-1$ , and from  $v = 0$  to  $v = N-1$ . In order to guarantee that these shifted coordinates are integers, usually  $M$  and  $N$  are taken to be even integers. When implementing the Fourier transform in a computer, the limit of summations are from  $u = 1$  to  $M$  and  $v = 1$  to  $N$ . The actual centre of the transform will then be at  $u = (M/2) + 1$  and  $v = (N/2) + 1$ . The value of the transform at  $(u, v) = (0, 0)$  is, from Eq.(1.35):

$$F(0, 0) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \quad (1.41)$$

which is the average of  $f(x, y)$ . In other words, if  $f(x, y)$  is an image, the value of the Fourier transform at the origin is equal to the average gray level of the image. Because both frequencies are zero at the origin,  $F(0, 0)$  sometimes is called the dc component of the spectrum. If  $f(x, y)$  is real, its Fourier transform is conjugate symmetric; that is,

$$F(u, v) = F^*(-u, -v) \quad (1.42)$$

where “\*” indicates the standard conjugate operation on a complex number. From this, it follows that

$$|F(u, v)| = |F(-u, -v)| \quad (1.43)$$

which says that the spectrum of the Fourier transform is symmetric (Gonzalez and Woods, 2002).

## 1.12 Image Interpolation techniques

### 1.12.1 Nearest Neighbour Interpolation

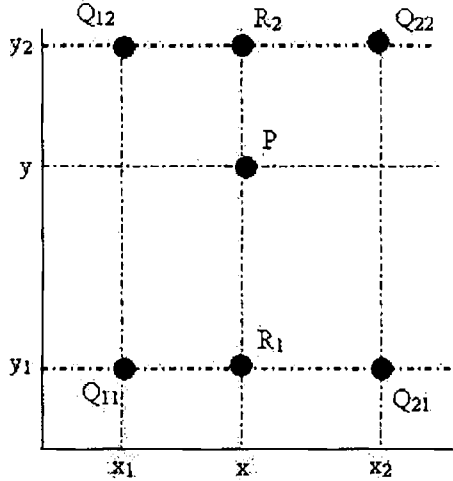
Nearest-neighbour interpolation (also known as proximal interpolation or point sampling in some contexts) is a simple method of multivariate\_interpolation in 1 or more dimensions. Interpolation is the problem of approximating the value for a non-given point in some space, when given some values of points around that point. The nearest neighbour algorithm simply selects the value of the nearest point, and does not consider the values of other neighbouring points at all, yielding a piecewise-constant interpolant. The algorithm is very simple to implement, and is commonly used in real-time 3D\_rendering to select colour values for a textured surface.

### 1.12.2 Bilinear Interpolation

In mathematics, bilinear interpolation is an extension of linear interpolation for interpolating functions of two variables on a regular grid. The key idea is to perform linear interpolation first in one direction, and then again in the other direction.

Suppose that we want to find the value of the unknown function  $f$  at the point  $P = (x, y)$ . It is assumed that we know the value of  $f$  at the four

points  $Q_{11} = (x_1, y_1)$ ,  $Q_{12} = (x_1, y_2)$ ,  $Q_{21} = (x_2, y_1)$ , and  $Q_{22} = (x_2, y_2)$  First do linear interpolation in the  $x$ -direction. This yields



*Fig.1.16 The four points ( $Q_{11}$ ,  $Q_{12}$ ,  $Q_{21}$ ,  $Q_{22}$ ) show the data point and point  $P$  is the point at which the data is to be interpolated*

$$f(R_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21})$$

$$\text{where } R_1 = (x, y_1), \quad (1.44)$$

$$f(R_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22})$$

$$\text{where } R_2 = (x, y_2). \quad (1.45)$$

We proceed by interpolating in the  $y$ -direction.

$$f(P) \approx \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2) \quad (1.46)$$

This gives us the desired estimate of  $f(x, y)$ .

$$\begin{aligned} f(x, y) \approx & \frac{f(Q_{11})}{(x_2 - x_1)(y_2 - y_1)} (x_2 - x)(y_2 - y) + \frac{f(Q_{21})}{(x_2 - x_1)(y_2 - y_1)} (x - x_1)(y_2 - y) \\ & + \frac{f(Q_{12})}{(x_2 - x_1)(y_2 - y_1)} (x_2 - x)(y - y_1) + \frac{f(Q_{22})}{(x_2 - x_1)(y_2 - y_1)} (x - x_1)(y - y_1) \end{aligned} \quad (1.47)$$

If we choose a coordinate system in which the four points where  $f$  is known are  $(0, 0)$ ,  $(0, 1)$ ,  $(1, 0)$ , and  $(1, 1)$ , then the interpolation formula simplifies to

$$f(x, y) \approx f(0, 0)(1 - x)(1 - y) + f(1, 0)x(1 - y) + f(0, 1)(1 - x)y + f(1, 1)xy \quad (1.48)$$

Or equivalently, in matrix operations:

$$f(x, y) \approx \begin{bmatrix} 1 - x & x \end{bmatrix} \begin{bmatrix} f(0, 0) & f(0, 1) \\ f(1, 0) & f(1, 1) \end{bmatrix} \begin{bmatrix} 1 - y \\ y \end{bmatrix} \quad (1.49)$$

Contrary to what the name suggests, the interpolant is not linear. Instead, it is of the form

$$(a_1x + a_2)(a_3y + a_4) \quad (1.50)$$



so that it is a product of two linear functions. Alternatively, the interpolant can be written as

$$b_1 + b_2x + b_3y + b_4xy \quad (1.51)$$

where

$$\begin{aligned} b_1 &= f(0, 0) \\ b_2 &= f(1, 0) - f(0, 0) \\ b_3 &= f(0, 1) - f(0, 0) \\ b_4 &= f(0, 0) - f(1, 0) - f(0, 1) + f(1, 1) \end{aligned} \quad (1.52)$$

In both cases, the number of constants (four) corresponds to the number of data points where  $f$  is given. The interpolant is linear along lines parallel to either the  $x$  or the  $y$  direction, equivalently if  $x$  or  $y$  is set constant. Along any other straight line, the interpolant is quadratic. The result of bilinear interpolation is independent of the order of interpolation. If we had first performed the linear interpolation in the  $y$ -direction and then in the  $x$ -direction, the resulting approximation would be the same.

In computer vision and image processing, bilinear interpolation is one of the basic resampling techniques. It is a texture mapping technique that produces a reasonably realistic image, also known as bilinear filtering or bilinear texture mapping. An algorithm is used to map a screen pixel location to a corresponding point on the texture map. A weighted average of the attributes (colour, alpha, etc.) of the four surrounding texels is

computed and applied to the screen pixel. This process is repeated for each pixel forming the object being textured.

When an image needs to be scaled-up, each pixel of the original image needs to be moved in certain direction based on scale constant. However, in scaling up an image, there are pixels (i.e. *Hole*) that are not assigned to appropriate pixel values. In this case, those *holes* should be assigned to appropriate image values so that the output image does not have non-value pixels.

Typically bilinear interpolation can be used where perfect image transformation, matching and imaging is impossible so that it can calculate and assign appropriate image values to pixels. Unlike other interpolation techniques such as nearest neighbour interpolation and bicubic interpolation (described below), bilinear interpolation uses the four nearest pixel values which are located in diagonal direction from that specific pixel in order to find the appropriate color intensity value of a desired pixel.

### **1.12.3 Bicubic Interpolation**

In mathematics, bicubic interpolation is an extension of cubic interpolation for interpolation of data points on a two dimensional regular grid. The interpolated surface is smoother than corresponding surfaces obtained by bilinear or nearest neighbour interpolation. Bicubic interpolation can be accomplished using Lagrange polynomials, cubic splines or cubic convolution algorithm.

In image processing, bicubic interpolation is often chosen over bilinear interpolation or nearest neighbor in image resampling, when

speed is not an issue. Images resampled with bicubic interpolation are smoother and have fewer interpolation artifacts

### 1.12.3a Bicubic spline interpolation

Suppose that the function values  $f$  and the derivatives  $f_x$ ,  $f_y$  and  $f_{xy}$  are known at the four corners  $(0,0)$ ,  $(1,0)$ ,  $(0,1)$ , and  $(1,1)$  of the unit square.

The interpolated surface can then be written

$$p(x, y) = \sum_{i=0}^3 \sum_{j=0}^3 a_{ij} x^i y^j \quad (1.53)$$

The interpolation problem consists of determining the 16 coefficients  $a_{ij}$ .

Matching  $p(x,y)$  with the function values yields four equations,

1.  $f(0,0) = p(0,0) = a_{00}$
2.  $f(1,0) = p(1,0) = a_{00} + a_{10} + a_{20} + a_{30}$
3.  $f(0,1) = p(0,1) = a_{00} + a_{01} + a_{02} + a_{03}$
4.  $f(1,1) = p(1,1) = \sum_{i=0}^3 \sum_{j=0}^3 a_{ij}$

(1.54)

Likewise, eight equations for the derivatives in the  $x$ -direction and the  $y$ -direction

1.  $f_x(0,0) = p_x(0,0) = a_{10}$
2.  $f_x(1,0) = p_x(1,0) = a_{10} + 2a_{20} + 3a_{30}$
3.  $f_x(0,1) = p_x(0,1) = a_{10} + a_{11} + a_{12} + a_{13}$
4.  $f_x(1,1) = p_x(1,1) = \sum_{i=1}^3 \sum_{j=0}^3 a_{ij} i$

(1.55)

5.  $f_x(0,0) = p_x(0,0) = a_{01}$
6.  $f_x(1,0) = p_x(1,0) = a_{01} + a_{11} + a_{21} + a_{31}$
7.  $f_x(0,1) = p_x(0,1) = a_{01} + 2a_{02} + 3a_{03}$
8.  $f_x(1,1) = p_x(1,1) = \sum_{i=0}^3 \sum_{j=1}^3 a_{ij} j$

And the following four equations represent the cross derivative

1.  $f_{xy}(0,0) = p_{xy}(0,0) = a_{11}$
2.  $f_{xy}(1,0) = p_{xy}(1,0) = a_{11} + 2a_{21} + 3a_{31}$
3.  $f_{xy}(0,1) = p_{xy}(0,1) = a_{11} + 2a_{12} + 3a_{13}$  (1.56)
4.  $f_{xy}(1,1) = p_{xy}(1,1) = \sum_{i=1}^3 \sum_{j=1}^3 a_{ij} ij$

The expressions above have used the following identities,

$$\begin{aligned}
 p_x(x, y) &= \sum_{i=1}^3 \sum_{j=0}^3 a_{ij} i x^{i-1} y^j \\
 p_y(x, y) &= \sum_{i=0}^3 \sum_{j=1}^3 a_{ij} x^i j y^{j-1} \\
 p_{xy}(x, y) &= \sum_{i=1}^3 \sum_{j=1}^3 a_{ij} i x^{i-1} j y^{j-1}
 \end{aligned} \tag{1.57}$$

This procedure yields a surface  $p(x,y)$  on the unit square  $[0, 1] \times [0,1]$  which is continuous and with continuous derivatives. Bicubic interpolation on an arbitrarily sized regular grid can then be accomplished by patching together such bicubic surfaces, ensuring that the derivatives match on the boundaries.

If the derivatives are unknown, they are typically approximated from the function values at points neighbouring the corners of the unit square, ie. using finite differences.

### 1.12.3b Bicubic convolution algorithm

Bicubic spline interpolation requires the solution of the linear system described above for each grid cell. An interpolator with similar properties can be obtained by applying convolution with the kernel in both dimensions:

$$W(x) = \begin{cases} 1 & \text{for } x = 0 \\ (a+2)|x|^3 - (a+3)|x|^2 + 1 & \text{for } 0 < |x| < 1 \\ a|x|^3 - 5a|x|^2 + 8a|x| - 4a & \text{for } 1 < |x| < 2 \\ 0 & \text{otherwise} \end{cases} \quad (1.58)$$

where  $a$  is usually set to  $-0.5$  or  $-0.75$ . Note that  $W(0) = 1$  and  $W(n) = 0$  for all nonzero integers  $n$ .

This approach was proposed by Keys who showed that  $a = -0.5$  (which corresponds to cubic Hermite spline) produces the best approximation of the original function. If we use the matrix notation for the common case  $a = -0.5$ , we can express the equation in a friendlier manner:

$$p(t) = \frac{1}{2} \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \begin{bmatrix} 0 & 2 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 2 & -5 & 4 & -1 \\ -1 & 3 & -3 & 1 \end{bmatrix} \begin{bmatrix} a_{-1} \\ a_0 \\ a_1 \\ a_2 \end{bmatrix} \quad (1.59)$$

for  $t$  between 0 and 1 for one dimension. For two dimensions first applied once in  $x$  and again in  $y$ :

$$\begin{aligned}
 b_{-1} &= p(t_x, a_{(-1,-1)}, a_{(0,-1)}, a_{(1,-1)}, a_{(2,-1)}) \\
 b_0 &= p(t_x, a_{(-1,0)}, a_{(0,0)}, a_{(1,0)}, a_{(2,0)}) \\
 b_1 &= p(t_x, a_{(-1,1)}, a_{(0,1)}, a_{(1,1)}, a_{(2,1)}) \\
 b_2 &= p(t_x, a_{(-1,2)}, a_{(0,2)}, a_{(1,2)}, a_{(2,2)})
 \end{aligned} \tag{1.60}$$

The bicubic algorithm is frequently used for scaling images and video for display. It preserves fine detail better than the common bilinear algorithm.

#### 1.12.4 Spline Interpolation

In the mathematical field of numerical analysis, spline interpolation is a form of interpolation where the interpolant is a special type of piecewise polynomial called a spline. Spline interpolation is preferred over polynomial interpolation because the interpolation error can be made small even when using low degree polynomials for the spline. Using polynomial interpolation, the polynomial of degree  $n$  which interpolates the data set is uniquely defined by the data points. The spline of degree  $n$  which interpolates the same data set is not uniquely defined, and we have to fill in  $n-1$  additional degrees of freedom to construct a unique spline interpolant.

Linear spline interpolation is the simplest form of spline interpolation and is equivalent to linear interpolation. The data points are graphically connected by straight lines. The resultant spline would be a

polygon if the end point is connected to the initial points. Algebraically, each  $S_i$  is a linear function constructed as:

$$S_i(x) = y_i + \frac{y_{i+1} - y_i}{x_{i+1} - x_i} (x - x_i) \quad (1.61)$$

The spline must be continuous at each data point, that is

$$S_{i-1}(x_i) = S_i(x_i), \quad i = 1, \dots, n-1 \quad (1.62)$$

This is the case as we can easily see

$$S_{i-1}(x_i) = y_{i-1} + \frac{y_i - y_{i-1}}{x_i - x_{i-1}} (x_i - x_{i-1}) = y_i \quad (1.63)$$

$$S_i(x_i) = y_i + \frac{y_{i+1} - y_i}{x_{i+1} - x_i} (x_{i+1} - x_i) = y_{i+1} \quad (1.64)$$

Commonly, magnification is accomplished through convolution of the image samples with a single kernel—typically the bilinear, bicubic (Netravali, 1995) or cubic B-spline kernel (Unser M et.al.,1991) . The mitigation of aliasing by this type of linear filtering is very limited. Magnification techniques based on a priori assumed knowledge are the subject of current research. Directional methods (Bayrakeri and Mersereau, 1995 and Jensen and Anastassiou, 1995) examine an image's local edge content and interpolate in the low frequency direction (along the edge) rather than in the high-frequency direction (across the edge).

Multiple kernel methods typically select between a few ad hoc interpolation kernels (Darwish and Bedair, 1996). Orthogonal transform methods focus on the use of the discrete cosine transform (DCT) (Martucci, 1995 and Shinbori and Takagi, 1994) and the wavelet transform (Chang et. al., 1995). Variational methods formulate the interpolation problem as the constrained minimization of a function (Karayiannis and Venetsanopoulos, 1991 and Schultz and Stevenson, 1994). It should be noted that these techniques make explicit assumptions regarding the character of the analog image.

With the rapid increase in available computing power, coupled with great strides in image feature analysis, model-based, often highly nonlinear interpolative techniques have become a viable alternative to classic linear methods and have received increasing attention recently. Several examples of model-based approaches to spatial image interpolation can be found in Jensen and Anastassiou (1995), Jensen and Anastassiou (1990), Martinez and Lim (1989), Wang and Mitra (1991). Each of these papers utilizes the concept of an edge in a different fashion to enhance interpolation results. Artificial neural network based interpolation of image processing is still in its infancy stage (Davila and Hunt, 2000). Hence it was thought worthwhile to pursue this technique for image processing applications.

## Summary

An introduction to neural networks and digital image processing is given in this chapter. Neural network can be viewed as a computational tool for solving complex real world problems. The advantage of using



neural network is that most of the computational complexities are encountered in the training phase itself. When implemented for a real world problem, the output obtained for the given inputs is only a mapping between the input and the output. The various tools for image processing is also introduced. The various available interpolation techniques are also discussed.

## References

- [1] Anderson, J.A., Rosenfeld, E., 1988. Neurocomputing: Foundations of Research. MIT Press, Cambridge, MA.
- [2] Basheer I A and M Hajmeer, 2000 Artificial neural networks: fundamentals, computing, design and application, Journal of microbiological methods 43 pp. 3-31
- [3] Bayrakeri S D and R. M. Mersereau, 1995. A new method for directional image interpolation, Proc. Int. Conf. Acoust., Speech, Signal Processing, vol. 4, 2383–2386.
- [4] Darwish A M and M. S. Bedair, 1996. An adaptive resampling algorithm for image zooming Proc. SPIE, vol. 2666, 131–144.
- [5] Davila C A and B R Hunt, 2000. Super-Resolution of Binary Images with a Nonlinear Interpolative Neural Network, Applied Optics, vol.39, No.14, . 2291-2299.
- [6] Davila C A and B R Hunt, 2000. Training of a Neural Network for Image Super-Resolution Based on a Nonlinear Interpolative Vector Quantiser , Applied Optics, vol.39, No.20, . 3473-3485.
- [7] Gonzalez R C and Richard E Woods, 2002. Digital Image Processing, Second Edition, Pearson Edn.

- [8] Hagan, Martin T Howard B Demuth and Mark Beale, 2002 Neural Network Design, first ed., Boston, Thomson Learning
- [9] Haykin, S., 2003. Neural Networks: A Comprehensive Foundation. Second Edition, Pearson Education
- [10] Hebb, D.O., 1949. The Organization of Behavior. Wiley, New York.
- [11] Hecht-Nielsen, R., 1990. Neurocomputing. Addison-Wesley, Reading, MA.
- [12] Hopfield, J.J., 1984. Neurons with graded response have collective computational properties like those of two-state neurons. Proc. Natl. Acad. Sci. 81, 3088–3092.
- [13] Jain, A.K., Mao, J., Mohiuddin, K.M., 1996. Artificial neural networks: a tutorial. Comput. IEEE March, 31–44.
- [14] Jensen K and Dimitris Anastassiou, 1995. Subpixel Edge Localization and the Interpolation of Still Images, IEEE Trans. on Image Processing, vol.4, . 285-295.
- [15] Jensen K and D. Anastassiou, 1990 .Spatial resolution enhancement of images using non-linear interpolation, in Proc. IEEE Int. Conf: Acoust. Speech. and Signal Processing (Albuquerque, NM).
- [16] Karayiannis N B and A. N. Venetsanopoulos, 1991. Image interpolation based on variational principles, Signal Processing, vol. 25, 259–288.
- [17] Martinez D M and J. S. Lim, 1989. Spatial interpolation of interlaced television pictures in Proc. IEEE Int. Conf. Acoust Speech. Signal Processing Scotland, 1886-1889.

- [18] Martucci S A, 1995. Image resizing in the discrete cosine transform domain, Proc. Int. Conf. Image Processing, vol. 2, 244–247.
- [19] Netravali A N and B. G. Haskell, 1995. Digital Pictures: Representation, Compression and Standards, 2nd ed. New York: Plenum.
- [20] McCulloch, W.S., Pitts, W., 1943. A logical calculus of the ideas immanent in nervous activity. Bull. Math. Biophys. 5, 115– 133.
- [21] Minsky, M., Pappert, S., 1969. Perceptrons. MIT Press, Cambridge, MA.
- [22] Nelson, M., Illingworth, W.T., 1990. A Practical Guide To Neural Nets. Addison-Wesley, Reading, MA.
- [23] Rich, Elaine., Knight, Kevin., 1994. Artificial Intelligence, Second Edition, Tata McGraw Hill Edition
- [24] Rosenblatt, R., 1962. Principles of Neurodynamics. Spartan Books, New York.
- [25] Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1986. Learning internal representation by error propagation. In: Rumelhart, D.E., McClelland, J.L. (Eds.). Parallel Distributed Processing: Exploration in the Microstructure of Cognition, Vol. 1. MIT Press, Cambridge, MA, Chapter 8.
- [26] Schultz R R and R. L. Stevenson, 1994. A Bayesian approach to image expansion for improved definition, IEEE Trans. Image Processing, vol.3, 233–242.

- [27] Shinbori E and M. Takagi, 1994. High-quality image magnification applying the Gerchberg-Papoulis iterative algorithm with DCT, Syst. Comput. Japan, vol. 25, no. 6, 80–90.
- [28] Unser M, A. Aldroubi, and M. Eden, 1991. Fast B-spline transforms for continuous image representation and interpolation, IEEE Trans. Pattern Anal. Machine Intell., vol. 13, 277–285.
- [29] von Neuman, J., 1958. The Computer and the Brain. MIT Press, Cambridge, MA.
- [30] Wang. Y and S. K. Mitra, 1991. Motion/pattern adaptive interpolation of interlaced video sequences, Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing (Toronto, Canada). 2829-2832.
- [31] Werbos, P.J., 1974. Beyond regression: new tools for prediction and analysis in the behavioral sciences. PhD Thesis, Harvard University.
- [32] Wythoff, B.J., 1993. Backpropagation neural networks: a tutorial. Chemometr. Intell. Lab. Syst. 18, 115–155.
- [33] Zupan, J., Gasteiger, J., 1993. Neural Networks For Chemists: An Introduction. VCH, New York.

## **CHAPTER 2**

# **DEVELOPMENT OF A SUCCESSFUL ARTIFICIAL NEURAL NETWORK**

### **2.1 Introduction**

Even though Artificial Neural Network, ANN, based models are empirical in nature, they can provide practically accurate solutions for precisely or imprecisely formulated problems and for phenomena that are only understood through experimental data and field observations. ANNs have been utilized in a variety of applications ranging from modeling, classification, pattern recognition, and multivariate data analysis. An attempt is made here to provide a preliminary understanding of the modeling methodologies, design considerations, applications of ANN to real world problems. Such understanding of ANN is essential for making efficient use of their features.

### **2.2 Backpropagation networks**

These networks are the most widely used type of networks and are considered the workhorse of ANNs. A backpropagation (BP) network is a multilayer Perceptron, MLP, consisting of (i) an input layer with nodes representing input variables to the problem, (ii) an output layer with nodes representing the dependent variables

(i.e., what is being modeled), and (iii) one or more hidden layers containing nodes to help capture the nonlinearity in the data. Using supervised learning, these networks can learn the mapping from one data space to another using examples. The term backpropagation refers to the way the error computed at the output side is propagated backward from the output layer, to the hidden layer, and finally to the input layer. In BPANNs, the data are fed forward into the network without feedback (i.e., all links are unidirectional and there are no same layer neuron-to-neuron connections). The neurons in BPANNs can be fully or partially interconnected. These networks are so versatile and can be used for data modeling, classification, forecasting, control, data and image compression, and pattern recognition (Hausson, 1995).

To extend the understanding of ANNs from the level of identifying what these systems are and to know how to design them, it is imperative to become familiar with ANN computation and design. For this objective, the BPANNs are discussed in more detail, considering their popularity, and their flexibility and adaptability in modeling a wide spectrum of problems in many application areas.

The feedforward error-backpropagation learning algorithm is the most famous procedure for training ANNs. BP is based on searching an error surface (error as a function of ANN weights) using gradient descent for point(s) with minimum error. Each

iteration in BP constitutes two sweeps: forward activation to produce a solution, and a backward propagation of the computed error to modify the weights. In an initialized ANN (i.e., an ANN with assumed initial weights), the forward sweep involves presenting the network with one training example. This starts at the input layer where each input node transmits the value received forward to each hidden node in the hidden layer. The collective effect on each of the hidden nodes is summed up by performing the dot product of all values of input nodes and their corresponding interconnection weights. Once the net effect at one hidden node is determined, the activation at that node is calculated using a transfer function (e.g., sigmoidal function) to yield an output between 0 and +1 or -1 and +1. The amount of activation obtained represents the new signal that is to be transferred forward to the subsequent layer (e.g., either hidden or output layer). The same procedure of calculating the net effect is repeated for each hidden node and for all hidden layers. The net effect(s) calculated at the output node(s) is consequently transformed into activation(s) using a transfer function. The activation(s) just calculated at the output node(s) represents the ANN solution of the fed example, which may deviate considerably from the target solution due to the arbitrary selection of interconnection weights. In the backward sweep, the difference (i.e., error) between the ANN and target outputs is used to adjust the interconnection weights, starting from the output layer, through all

hidden layers, to the input layer. The forward and backward sweeps are performed repeatedly until the ANN solution agrees with the target value within a prespecified tolerance. The BP learning algorithm provides the needed weight adjustments in the backward sweep (Basheer and Hajmeer, 2000).

### 2.3 BP Algorithm

Because of its importance and simplicity, the BP algorithm will be presented here in its final form. Detailed derivation of the algorithm is found elsewhere (eg. Haykin, 2003; Hagan et. al., 2002; Zupan and Gasteiger, 1993). In order to be able to run the algorithm, it is essential to define the interlayer as the gap between two successive layers that encloses the connection weights and contains only the neurons of the upper layer, as shown in Fig. 2.1 (assuming that all layers are positioned above the input layer). Consider an MLP network with L interlayers. For interlayer  $l \in \{1, 2, \dots, L\}$  there are N nodes and  $N_l \times N_{l-1}$  connection links with weights  $\mathbf{W} \in \mathbb{R}^{N_l \times N_{l-1}}$ , where  $N_l$  and  $N_{l-1}$  are the number of nodes (including thresholds) in interlayers  $l$  and  $l-1$ , respectively (Fig.2.1). A connection weight is denoted by  $W_{ji}^l$  if it resides in interlayer  $l$  and connects node  $j$  of interlayer  $l$  with node  $i$  of lower (preceding) internode layer  $l-1$  (node  $i$  is the source node and node  $j$  is the destination node).



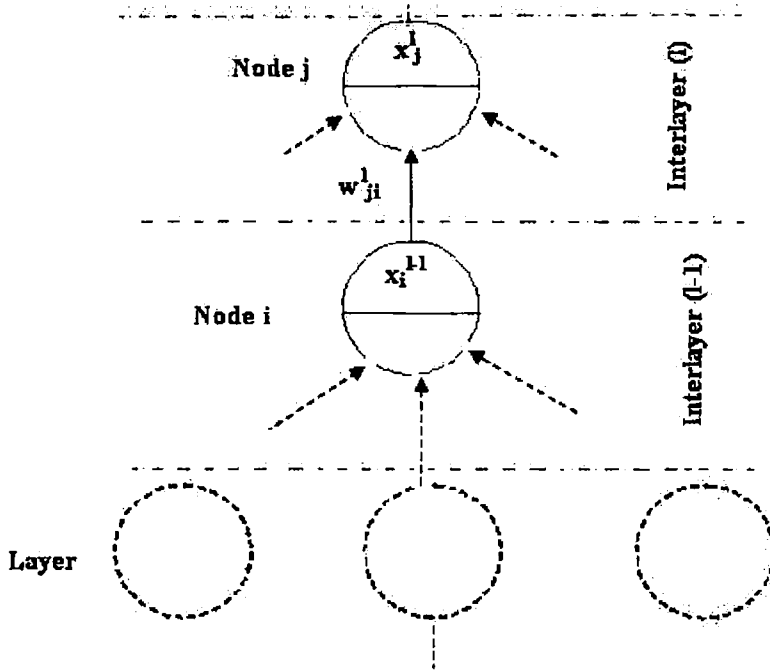


Fig.2.1 Notations and index labeling used in backpropagation ANNS

In any interlayer  $l$ , a typical neuron  $j$  integrates the signals,  $x_j$ , impinging onto it, and produces a net effect,  $\xi_j$ , according to linear neuron dynamics:

$$\xi_j^l = \sum_{i=1}^{N_{l-1}} w_{ji}^l x_i^{l-1} \quad (2.1)$$

The corresponding activation,  $x_j$ , of the neuron is determined using a transfer function,  $\sigma$ , that converts the total signal into a real number from a bounded interval:

$$x'_j = \sigma(\xi'_j) = \sigma\left(\sum_{i=1}^{N_{l-1}} w'_{ji} x_i^{l-1}\right) \quad (2.2)$$

One popular function used in BP is the basic continuous sigmoid:

$$\sigma(\xi) = \frac{1}{1 + e^{-\xi}} \quad (2.3)$$

where  $-\infty < \xi < \infty$  and  $0.0 \leq \sigma \leq 1.0$ . Eqs. (2.1)–(2.3) are used for all nodes to calculate the activation. For the input nodes the activation is simply the raw input. In any interlayer, an arbitrary weight  $w'_{ji}$  at iteration ( $t$ ) will be updated from its previous state ( $t-1$ ) value according to

$$w'_{ji}(t) = w'_{ji}(t-1) + \Delta w'_{ji}(t) \quad (2.4)$$

where  $\Delta w'_{ji}$  is the (+/-) incremental change in the weight. The weight change is determined via the modified delta rule which can be written as

$$\Delta w'_{ji} = \eta \delta'_j x_i^{l-1} + \mu \Delta w'_{ji}^{(previous)} \quad (2.5)$$

where  $\eta$  is the learning rate controlling the update step size,  $\mu$  is the momentum coefficient, and  $x_i^{l-1}$  is the input from the  $l-1$ th interlayer. The first part of the right-hand side of Eq. (2.5) is the original delta rule. The added momentum term helps to direct the search on the error hyperspace to the global minimum by allowing a portion of the previous updating (magnitude and direction) to be added to the current updating step. Note that Eq. (2.5) can also be

applied to any neuron threshold (bias) which can be assumed as a link, with weight equal to the threshold value, for an imaginary neuron whose activation is fixed at 1.0. The weight change can also be determined using a gradient descent written in generalized form for an interlayer  $l$ :

$$\Delta w_{ji}^l = \kappa \left( \frac{\partial \varepsilon^l}{\partial w_{ji}^l} \right) \quad (2.6)$$

Therefore, in order to determine the incremental changes for the  $l$ th interlayer, the main task is to quantify the error gradient  $(\partial \varepsilon^l / \partial w_{ji}^l)$ . Using Eqs. (2.5) and (2.6), the required weight change can be derived with different expressions depending on whether the considered neuron is in the output layer or in a hidden layer. If the neuron is in the output layer, then  $l=L$  in Eq. (2.5), with  $\delta_j^L$  calculated from

$$\delta_j^L = (x_j^L - y_j) x_j^L (1 - x_j^L) \quad (2.7)$$

If the neuron is in a hidden layer, the weight change is also calculated using Eq. (2.5) with  $\delta_j^l$  determined from

$$\delta_j^l = x_j^l (1 - x_j^l) \left( \sum_{k=1}^r \delta_k^{l+1} w_{kj}^{l+1} \right) \quad (2.8)$$

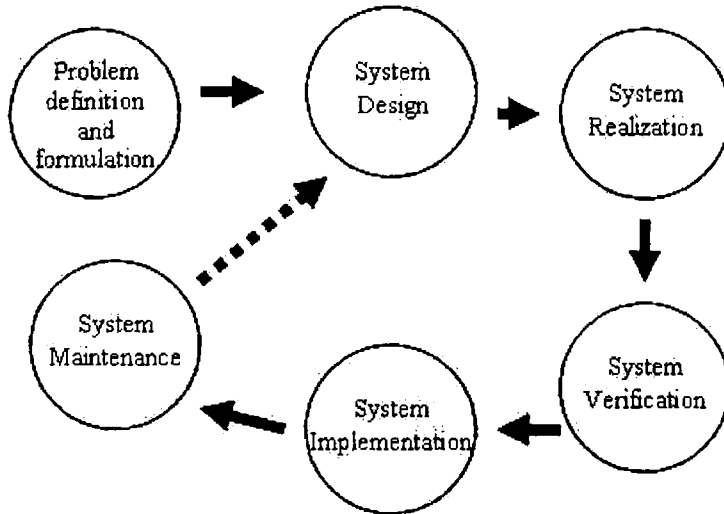
where  $\delta_k^{l+1}$  is calculated for a given non-output layer ( $l$ ) beginning with a layer one level up ( $l+1$ ) and moving down layer by layer. That is, for the last (uppermost) hidden layer in a network,  $\delta_j^l$  is

determined via  $\delta_k^{l+1}$  of the output layer calculated using Eq. (2.7). The above delta equations (Eqs. (2.7) and (2.8)) are based on the sigmoid transfer function given in Eq. (2.3). For a different function, the terms  $x_j^l(1-x_j^l)$  and  $x_j^l(1-x_j^l)$  in Eqs. (2.7) and (2.8), respectively, should be replaced with the relevant first derivative of the used function. This technique of distributing backward the errors starting from the output layer down through the hidden layer gives the method the name backpropagation of error with the modified delta rule (Rumelhart et. al., 1986). The standard BP have been modified in several ways to achieve a better search and accelerate and stabilize the training process (Hagan et. al., 2002; Looney, 1996; Masters, 1994).

## 2.4 ANN Development Project

The development of a successful ANN project constitutes a cycle of six phases, as illustrated in Fig. 2.2. The problem definition and formulation (phase 1) relies heavily on an adequate understanding of the problem, particularly the 'cause-effect' relationships. The benefits of ANNs over other techniques (if available) should be evaluated before final selection of the modeling technique. System design (phase 2) is the first step in the actual ANN design in which the type of ANN is determined along with the learning rule that fit the problem. This phase also involves data collection, data

preprocessing to fit the type of ANN used, statistical analysis of data, and partitioning the data into three distinct subsets (training, test, and validation subsets).



*Fig. 2.2 Various phases of ANN development project*

System realization (phase 3) involves training of the network utilizing the training and test subsets, and simultaneously assessing the network performance by analyzing the prediction error. Optimal selection of the various parameters (e.g., network size, learning rate, number of training cycles, acceptable error, etc.) can affect the design and performance of the final network. Splitting the problem into smaller sub-problems, if possible, and designing an ensemble of networks could enhance the overall system accuracy. This takes

us back to phase 2. In system verification (phase 4), although network development includes ANN testing against the test data while training is in progress, it is good practice (if data permit) that the 'best' network be examined for its generalization capability using the validation subset. Verification is intended to confirm the capability of the ANN-based model to respond accurately to examples never used in network development. This phase also includes comparing the performance of the ANN-based model to those of other approaches (if available) such as statistical regression and expert systems. System implementation (phase 5) includes embedding the obtained network in an appropriate working system such as hardware controller or computer program. Final testing of the integrated system should also be carried out before its release to the end user. System maintenance (phase 6) involves updating the developed system as changes in the environment or the system variables occur (e.g., new data), which involves a new development cycle.

## **2.5 General issues in ANN development**

A number of issues should be addressed before initiation of any network training. Some of the following issues are only relevant to BP ANNs while others are applicable to the design of all ANN types.

### **2.5.1 Database size and partitioning**

Models developed from data generally depend on database size. ANNs, like other empirical models, may be obtained from databases of any size, however generalization of these models to data from outside the model development domain will be adversely affected. Since ANNs are required to generalize for unseen cases, they must be used as interpolators. Data to be used for training should be sufficiently large to cover the possible known variation in the problem domain.

The development of an ANN requires partitioning of the parent database into three subsets: training, test, and validation. The training subset should include all the data belonging to the problem domain and is used in the training phase to update the weights of the network. The test subset is used during the learning process to check the network response for untrained data. The data used in the test subset should be distinct from those used in the training; however they should lie within the training data boundaries. Based on the performance of the ANN on the test subset, the architecture may be changed and/or more training cycles be applied. The third portion of the data is the validation subset which should include examples different from those in the other two subsets. This subset is used after selecting the best network to further examine the network or confirm its accuracy before being implemented in the neural system and/or delivered to the end user.

Currently, there are no mathematical rules for the determination of the required sizes of the various data subsets. Only some rules of thumb derived from experience and analogy between ANNs and statistical regression exist. Baum and Haussler (1989) propose the minimum size of the training subset to be equal to the number of weights in the network times the inverse of the minimum target error. Dowla and Rogers (1995) and Haykin (1994) suggest an example-to-weight ratio (EWR)  $> 10$ , while Masters (1994) suggests  $EWR > 4$ . For database partitioning, a large test subset may highlight the generalization capability better; however, the remaining smaller training subset may not be adequate to train the network satisfactorily. Looney (1996) recommends 65% of the parent database to be used for training, 25% for testing, and 10% for validation, whereas Swingler (1996) proposes 20% for testing and Nelson and Illingworth (1990) suggest 20–30%.

### **2.5.2 Data preprocessing, balancing, and enrichment**

Several preprocessing techniques are usually applied before the data can be used for training to accelerate convergence. Among these are noise removal, reducing input dimensionality, and data transformation (Dowla and Rodgers, 1995; Swingler, 1996), treatment of non-normally distributed data, data inspection, and deletion of outliers (Masters, 1994; Stein, 1993). Balancing of data is especially important in classification problems. It is desired that the



training data be distributed nearly evenly between the various classes to prevent the network from being biased to the over-represented classes. To balance a database, some of the over-represented classes may be removed or extra examples pertaining to the under-represented class added. Another way is by duplicating the under-represented input/ output examples and adding random noise to their input data (while keeping the output class unchanged). Swingler (1996) suggests the use of information theory to measure the degree of balance of the training database.

Small database size poses another problem in ANN development because of the inability to partition the database into fairly-sized subsets for training, test, and validation. To expand the size of the database, the trivial way is to get new data (if possible) or introduce random noise in the available examples to generate new ones. Noise addition normally enhances the ANN robustness against measurement error (e.g., noise =  $\pm$  Instrument sensitivity). If data enrichment is not possible, the leave-one-out method (or leave-k-out method) may be used for developing a network (Hecht – Nielsen, 1990; Rizzo and Dougherty, 1994). With M exemplars available, a network is trained on M-1 (or M-k) exemplars, and tested on the one (or k) unused exemplar(s). The procedure is repeated M times, each with a new set of randomly initialized weights. The solutions of the M networks are then averaged to obtain a representative solution to the problem. Other techniques to

train and validate networks with limited data include grouped cross-validation, grouped jackknife, and bootstrap (Twomey and Smith, 1997).

### 2.5.3 Data normalization

Normalization (scaling) of data within a uniform range (e.g., 0–1) is essential (i) to prevent larger numbers from overriding smaller ones, and (ii) to prevent premature saturation of hidden nodes, which impedes the learning process. This is especially true when actual input data take large values. There is no unique standard procedure for normalizing inputs and outputs. One way is to scale input and output variables ( $z_i$ ) in interval  $[\lambda_1, \lambda_2]$  corresponding to the range of the transfer function:

$$x_i = \lambda_1 + (\lambda_2 - \lambda_1) \left( \frac{z_i - z_i^{\min}}{z_i^{\max} - z_i^{\min}} \right) \quad (2.9)$$

where  $x_i$  is the normalized value of  $z_i$ , and  $z_i^{\max}$  and  $z_i^{\min}$  are the maximum and minimum values of  $z$  in the database. It is recommended that the data be normalized between slightly offset values such as 0.1 and 0.9 rather than between 0 and 1 to avoid saturation of the sigmoid function leading to slow or no learning (Haussoun, 1995; Masters, 1994). Other more computationally involved techniques are given by Masters (1994), Swingler (1996), and Dowla and Rogers (1995). Masters (1994) indicates that more complicated techniques may not produce any better solution than

that obtained using linear normalization (Eq. (2.9)). For parameters with an exceptionally large range, it may be beneficial to take the logarithm of data prior to normalization [if data contain zeros,  $\log(z_i + 1)$  may be used].

#### **2.5.4 Input /output representation**

Proper data representation also plays a role in the design of a successful ANN (Masters, 1994). The data inputs and outputs can be continuous, discrete, or a mixture of both. For example, in a classification problem where each of the input variable belongs to one of several classes and the output is also a class, all the inputs and outputs may be represented by binary numbers such as 0 and 1 (or 0.1 and 0.9 to prevent saturation). If two inputs (A and B) are to be assigned to four levels of activation (e.g., low, medium, high, and very high), then each input may be represented by two binary numbers such as 00, 01, 10, and 11 to indicate the four levels. Another representation may assign four binary numbers to each input such as 0001, 0010, 0100, and 1000 where the location of 1 determines the type of activation of the input variable. Similar treatment applies to the output variables. This representation increases the dimensionality of the input vector (the two-digit representation converts the input vector into four inputs and the four-digit representation into eight inputs). Binary inputs and outputs are very useful in extracting rules from a trained network.

For this purpose, a continuous variable may be replaced by binary numbers by partitioning its range into a number of intervals, each assigned to a unique class. Specialized algorithms for discretizing variables based on their distribution also exist (Kerber, 1992).

### **2.5.5 Network weight initialization**

Initialization of a network involves assigning initial values for the weights (and thresholds) of all connections links. Some researchers (eg., Li et. al., 1993; Schmidt et. al., 1993) indicate that weights initialization can have an effect on network convergence. Hassoun (1995) explained that if the initial weight vector is stationed in a flat of the error surface the convergence may become extremely slow. Other studies (eg., Fahlman, 1988) have shown that initialization has an insignificant effect on both the convergence and final network architecture. Typically, weights and thresholds are initialized uniformly in a relatively small range with zero-mean random numbers. However, an extremely small range can lead to very small error gradients which may slow down the initial learning process. The choice of small numbers is very essential to reduce the likelihood of premature neurons saturation. ASCE (2000) recommends that weights and thresholds be assigned initial small random values between -0.30 and +0.30. Weight initialization can also be performed on a neuron-by-neuron basis by assigning values uniformly sampled from the range  $(-r/N_j, +r/N_j)$ , where  $r$  is a real

number depending on the neuron activation function, and  $N_j$  is the number of connections feeding into neuron  $j$ . Wessels and Barnard (1992) use zero-mean and unit standard deviation for links feeding neurons with weights sampled from  $[-3M^{1/2}, +3M^{1/2}]$ , where  $M$  is the number of weights in a given interlayer. Nguyen and Widrow (1990) initialize the weight vector so that each input exemplar is likely to force a hidden unit to learn efficiently.

### 2.5.6 BP learning rate ( $\eta$ )

A high learning rate,  $\eta$ , will accelerate training (because of the large step) by changing the weight vector,  $W$ , significantly from one cycle to another. However, this may cause the search to oscillate on the error surface and never converge, thus increasing the risk of overshooting a near-optimal  $W$ . In contrast, a small learning rate drives the search steadily in the direction of the global minimum, though slowly. A constant learning rate may be utilized throughout the training process. Wythoff (1993) suggests  $\eta = 0.1-1.0$ , Zupan and Gasteiger (1993) recommend  $\eta=0.3-0.6$ , and Fu (1995) recommends  $\eta=0.0-1.0$ . The adaptive learning rate  $[\eta(t)]$ , which varies along the course of training, could also be used and can be effective in achieving an optimal weight vector for some problems. Generally, larger steps are needed when the search is far away from a minimum, and smaller steps as the search approaches a

minimum. Since the distance from a minimum cannot be predicted, various heuristics have been proposed.

### **2.5.7 BP momentum coefficient ( $\mu$ )**

A momentum term is commonly used in weight updating to help the search escape local minima and reduce the likelihood of search instability. As implied in Eq.(2.5),  $\mu$  accelerates the weight updates when there is a need to reduce  $\eta$  to avoid oscillation. A high  $\mu$  will reduce the risk of the network being stuck in local minima, but it increases the risk of overshooting the solution as does a high learning rate. A  $\mu > 1.0$  yields excessive contributions of the weight increments of the previous step and may cause instability. Conversely, an extremely small  $\mu$  leads to slow training. Both a constant and adaptable momentum can be utilized. Wythoff (1993) suggests  $\mu = 0.4-0.9$ , Hassoun (1995) and Fu (1995) suggest  $\mu = 0.0-1.0$ , Henseler (1995) and Hertz (1991) suggest  $\mu \approx 1.0$ , and Zupan and Gasteiger (1993) suggest that  $\eta + \mu \approx 1$ . Swingler (1996) uses  $\mu = 0.9$  and  $\eta = 0.25$  in solving all problems unless a good solution could not be obtained. Depending on the problem being solved, it seems that the success of training varies with the selected  $\mu$ , and a trial-and-error procedure is normally preferred. Adaptive momentum involves varying  $\mu$  with the training cycle [i.e.,  $\mu(t)$ ] in which the changes in  $\mu$  are made in relation to error gradient

information. Other methods relate  $\mu$  to the adaptive learning rate  $\eta$  such that  $\mu$  is decreased when learning speeds up. Finally, the addition of momentum should be considered with caution because of the need of doubling computer space for storing weights of current and previous iterations (see Eq. (2.5)).

### **2.5.8 Transfer function, $\sigma$**

The transfer (activation) function,  $\sigma$ , is necessary to transform the weighted sum of all signals impinging onto a neuron so as to determine its firing intensity. Some functions are designed to indicate only whether a neuron can fire (step functions) regardless of the magnitude of the net excitation ( $\xi$ ) by comparing  $\xi$  to the neuron threshold. Most applications utilizing BPANNs employ a sigmoid function, which possesses the distinctive properties of continuity and differentiability on  $(-\infty, \infty)$  interval, essential requirements in BP learning. Moody and Yarvin (1992) reported various success rates with different transfer functions in relation to data nonlinearity and noisiness. Han (1996) use a variant logistic function with three adjustable parameters, and each neuron is assigned a different set of values for these parameters. The advantage of choosing a particular transfer function over another is not yet theoretically understood.

### 2.5.9 Convergence criteria

Three different criteria may be used to stop training: (i) training error ( $\rho \leq \varepsilon$ ), (ii) gradient of error ( $\nabla\rho \leq \delta$ ), and (iii) cross-validation, where  $\rho$  is the arbitrary error function, while  $\varepsilon$  and  $\delta$  are small real numbers. The third criterion is more reliable; however it is computationally more demanding and often requires abundant data. Convergence is usually based on the error function,  $\rho$ , exhibiting deviation of the predictions from the corresponding target output values such as the sum of squares of deviations. Training proceeds until  $\rho$  reduces to a desired minimum. The function  $\rho$  may also be expressed as the relative error of the absolute values of the deviations averaged over the subset. Another criterion is the coefficient-of-determination,  $R^2$ , representing the agreement between the predicted and target outputs. Other more involved methods for monitoring network training and generalization are based on information theory (Swingler, 1996).

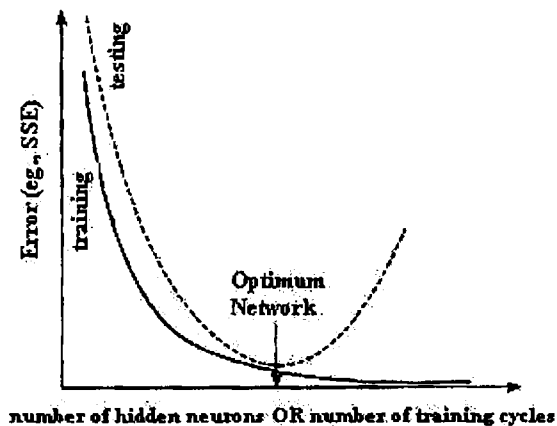
The most commonly used stopping criteria in neural network training is the sum-of-squared-errors (SSE) calculated for the training or test subsets as

$$SSE = \frac{1}{N} \sum_{p=1}^N \sum_{i=1}^M (t_{pi} - O_{pi})^2 \quad (2.10)$$

where  $O_{pi}$  and  $t_{pi}$  are, respectively, the actual and target solution of the  $i$ th output node on the  $p$ th example,  $N$  is the number of training examples, and  $M$  is the number of output nodes. Some SSE criteria



incorporate a measure of complexity of the network architecture. Generally, the error on training data decreases indefinitely with increasing number of hidden nodes or training cycles, as shown in Fig.2.3. The initial large drop in error is due to learning, but the subsequent slow reduction in error may be attributed to (i) network memorization resulting from the excessively large number of training cycles used, and/or (ii) overfitting due to the use of a large number of hidden nodes. During ANN training, the error on test subsets is monitored which generally shows an initial reduction and a subsequent increase due to memorization and overtraining of the trained ANN. The final (optimal) neural network architecture is obtained at the onset of the increase in test data error.



*Fig.2.3 Criteria of termination of training and selection of optimum network architecture*

Other error metrics may be used and may perform equally well in terms of optimizing network structure. For classification problems (discrete-valued output), the convergence criterion should be based on the hit (or miss) rate representing the percentage of examples classified correctly (or incorrectly), or confusion matrices (Lakshmanan, 1997), rather than the absolute deviation of the network classification from the target classification.

### **2.5.10 Number of training cycles**

The number of training cycles required for proper generalization may be determined by trial and error. For a given ANN architecture, the error in both training and test data is monitored for each training cycle. Training for so long can result in a network that can only serve as a look-up table, a phenomenon called overtraining or memorization (Zupan and Gasteiger, 1993; Wythoff, 1993). Theoretically, excessive training can result in near-zero error on predicting the training data (called recall), however generalization on test data may degrade significantly (Fig. 2.3). Initially, the test subset error continues to decrease with the number of training cycles. As the network loses its ability to generalize on the test data, the error starts to build up after each epoch. Although the error on the test data may not follow a smooth path, the onset of a major increase in the error is considered to represent the optimal number of cycles for that ANN architecture.

### 2.5.11 Training modes

Training examples are presented to the network in either one or a combination of two modes: (i) example-by-example training (EET), and (ii) batch training (BT) (Zupan and Gasteiger, 1993; Wythoff, 1993). In EET mode, the weights are updated immediately after the presentation of each training example. Here, the first example is presented to the network, and the BP learning algorithm consisting of feedforward and backward sweeps is applied for either a specified number of iterations or until the error drops to the desired level. Once the first example is learnt, the second example is presented and the procedure is repeated. The advantages of EET include the smaller storage requirements for the weights as opposed to BT, and the better stochastic search, which prevents entrapment in local minima. The disadvantage of EET is associated with the fact that learning may become stuck in a first very bad example, which may force the search in the wrong direction. Conversely, BT requires that weight updating be performed after all training examples have been presented to the network. That is, the first learning cycle will include the presentation of all the training examples, the error is averaged over all the training examples (e.g., Eq. (2.10)), and then backpropagated according to the BP learning law. Once done, the second cycle includes another presentation of all examples, and so on. The advantages of the BT mode include a better estimate of the error gradient vector and a

more representative measurement of the required weight change. However, this training mode requires a large storage of weights, and is more likely to be trapped in a local minimum. For a better search, the order of presentation of the training examples may be randomized between successive training cycles. The effectiveness of the two training modes can be problem specific (Hertz et. al., 1991; Haykin, 2003; Swingler, 1996).

### **2.5.12 Hidden layer size**

In most function approximation problems, one hidden layer is sufficient to approximate continuous functions (Basheer et. al., 2000; Hecht-Nielsen, 1990). Generally, two hidden layers may be necessary for learning functions with discontinuities. The determination of the appropriate number of hidden layers and number of hidden nodes (NHN) in each layer is one of the most critical tasks in ANN design. Unlike the input and output layers, one starts with no prior knowledge as to the number and size of hidden layers. As shown in Fig. 2.4, a network with too few hidden nodes would be incapable of differentiating between complex patterns leading to only a linear estimate of the actual trend. In contrast, if the network has too many hidden nodes it will follow the noise in the data due to overparameterization leading to poor generalization for untrained data (Fig. 2.4). With increasing number of hidden nodes, training becomes excessively time-consuming.

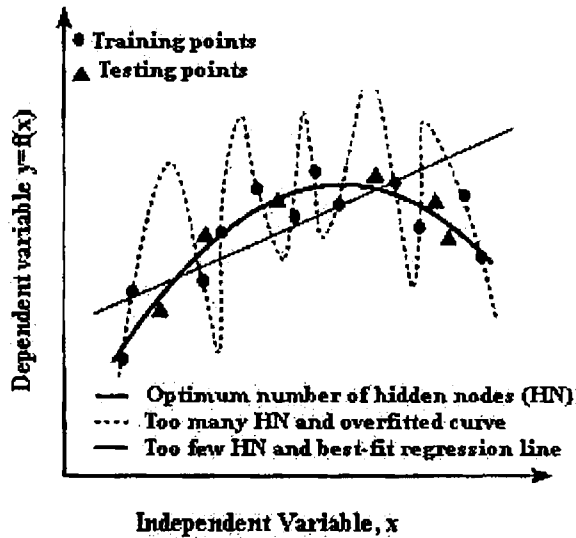


Fig.2.4 Effect of hidden layer size on network generalization

The optimal number of HN essential for network generalization may be a function of input / output vector sizes, size of training and test subsets, and, more importantly, the problem of nonlinearity. Several rules of thumb are available in the literature which relate hidden layer size to the number of nodes in input ( $N_{INP}$ ) and output ( $N_{OUT}$ ) layers. Jadid and Fairbairn (1996) called for an upper bound on NHN equal to  $N_{TRN} / [R + (N_{INP} + N_{OUT})]$ , where  $N_{TRN}$  is the number of training patterns and  $R = 5 - 10$ . Lachtermacher and Fuller (1995) suggest that NHN for a one-output ANN with no biases be determined from  $0.11N_{TRN} \leq \text{NHN}(N_{INP} + 1) \leq 0.30N_{TRN}$ .

Masters (1994) suggests that the ANN architecture should resemble a pyramid with  $NHN \approx (N_{INP} \cdot N_{OUT})$ . Hecht-Nielsen (1990) used the Kolmogorov theorem to prove that  $NHN \leq N_{INP} + 1$ . Upadhaya and Eryureka (1992) related  $NHN$  to  $N_{TRN}$  (via the total number of weights,  $N_w$ ) according to  $N_w = N_{TRN} \log_2(N_{TRN})$ , and Widrow and Lehr (1990) according to  $(N_w / N_{OUT}) \leq N_{TRN} \leq (N_w / N_{OUT}) \log_2(N_w / N_{OUT})$ .

Facing exotic problems such as those with high nonlinearity and hysteresis normally forces us to try networks with hidden layers that may not conform to any of these rules of thumb. The most popular approach to find the optimal number of hidden nodes is by trial and error with one of the above rules as starting point. Another way is to begin with a small number of hidden nodes and build on as needed to meet the model accuracy demand. Again, the cross-validation technique for determining the proper size of the hidden layer involves monitoring the error on both the training and test subsets in a way similar to that used to stop training (Fig. 2.3). Among other popular but more sophisticated techniques of optimizing network size are the growing and pruning methods.

### **2.5.13 Parameter optimization**

As can be seen, BP training requires a good selection of values of several parameters, commonly through trial and error. Six parameters should not be set too high (large) or too low (small), and

thus should be optimized or carefully selected. Table 2.1 lists these parameters and their effect on both learning convergence and overall network performance.

Design Parameter	Too high or too large	Too low or too small
Number of hidden nodes, $N_{HN}$	Overfitting ANN (No generalization)	Underfitting (ANN unable to obtain the rules embedded in the data)
Learning rate, $\eta$	Unstable ANN (weights) that oscillates about the optimal solution	Slow training
Momentum coefficient, $\mu$	Reduces risk of local minima. Speeds up training. Increased risk of overshooting the solution (instability)	Suppresses effect of momentum leading to increased risk of potential entrapment in local minima. Slows training
Number of training cycles	Good recalling ANN (ie., ANN memorization of data) and bad generalization of untrained data	Produces ANN that is incapable of representing the data
Size of training subset, $N_{TRN}$	ANN with good recalling and generalization	ANN unable to fully explain the problem. ANN with limited or bad generalization
Size of test subset, $N_{TST}$	Ability to confirm ANN generalization capability	Inadequate confirmation of ANN generalization capability

*Table 2.1 Effect of extreme values of design parameters on training convergence and network generalization*

## Summary

The remarkable information processing capabilities of ANNs and their ability to learn from examples make them efficient problem-solving paradigms. A review of the basic issues pertaining

to ANN-based computing and ANN design is discussed. A generalized methodology for developing ANN projects from the early stages of data acquisition to the latest stages of utilizing the model to derive useful information was also proposed and discussed. The increased utilization of ANNs is linked to several features they possess, namely (i) the ability to recognize and learn the underlying relations between input and output without explicit physical consideration, regardless of the problem's dimensionality and system nonlinearity, and (ii) the high tolerance to data containing noise and measurement errors due to distributed processing within the network. ANNs also have limitations that should not be overlooked. These include (i) ANNs' success depends on both the quality and quantity of the data, (ii) a lack of clear rules or fixed guidelines for optimal ANN architecture design, (iii) a lack of physical concepts and relations, and (iv) the inability to explain in a comprehensible form the process through which a given decision (answer) was made by the ANN. ANNs are not a panacea to all real-world problems; for that, other traditional (non-neural) techniques are powerful in their own ways. Hybridizing ANNs with conventional approaches such as expert systems can yield stronger computational paradigms for solving complex and computationally expensive problems. Recently, ANNs have attracted the attention of the microbiology community, particularly in the area of pyrolysis mass spectrometry and microbial growth in food systems.



## References

- [1] ASCE, 2000. Artificial neural networks in hydrology. I. Preliminary concepts. *J. Hydro. Eng. ASCE* 5, 115–123.
- [2] Basheer I.A., M. Hajmeer - Artificial neural networks: fundamentals, computing, design, and application, *Journal of Microbiological Methods* 43 (2000) 3-31
- [3] Baum, E., Haussler, D., 1989. What size net gives valid generalization? *Neural Computation* 1, 151–160.
- [4] Dowla, F.U., Rogers, L.L., 1995. *Solving Problems in Environmental Engineering and Geosciences With Artificial Neural Networks*. MIT Press, Cambridge, MA.
- [5] Fahlman, S.E., 1988. An empirical study of learning speed in backpropagation. Technical Report, CMU-CS-88-162, Carnegie-Mellon University.
- [6] Fu, L., 1995. *Neural Networks in Computer Intelligence*. McGraw-Hill, New York.
- [7] Han, J., Moraga, C., Sinne, S., 1996. Optimization of feedforward neural networks. *Eng. Appl. Artif. Intell.* 9 (2), 109–119.
- [8] Hassoun, M.H., 1995. *Fundamentals of Artificial Neural Networks*. MIT Press, Cambridge, MA.
- [9] Haykin, S., 2003. *Neural Networks: A Comprehensive Foundation*. Second Edition, Pearson Education

- [10] Hagan, Martin T., Howard B Demuth, Mark Beale, 2002–  
Neural Network Design, Thomson Learning, New Delhi.
- [11] Hecht-Nielsen, R., 1990. Neurocomputing. Addison-  
Wesley, Reading, MA.
- [12] Hertz, J., Krogh, A., Palmer, R.G., 1991. Introduction to the  
Theory of Neural Computation. Addison-Wesley, Reading,  
MA.
- [13] Jadid, M.N., Fairbairn, D.R., 1996. Predicting moment-  
curvature parameters from experimental data. Eng. Appl.  
Artif. Intell. 9 (3), 309–319.
- [14] Kerber, R., 1992. ChiMerge: discretization of numeric  
attributes. In: AAAI-92, Proceedings of the 9th National  
Conference on AI. MIT Press, Cambridge, MA, pp. 123–  
128
- [15] Lachtermacher, G., Fuller, J.D., 1995. Backpropagation in  
timeseries forecasting. J. Forecasting 14, 381–393.
- [16] Lakshmanan, V., 1997. Detecting rare signatures. In: Dagli,  
C.H. et al. (Eds.). Artificial Neural Networks in  
Engineering, ANNIE, Vol. 7. ASME Press, NY, pp. 521–  
526.
- [17] Li, G., Alnuweiri, H., Wu, W., 1993. Acceleration of  
backpropagation through initial weight pre-training with  
Delta rule. In: Proceedings of an International Joint

- Conference on Neural Networks, San Francisco, CA, pp. 580–585.
- [18] Looney, C.G., 1996. Advances in feedforward neural networks: demystifying knowledge acquiring black boxes. *IEEE Trans. Knowledge Data Eng.* 8 (2), 211–226.
- [19] Masters, T., 1994. *Practical Neural Network Recipes in C++*. Academic Press, Boston, MA.
- [20] Moody, J., Yarvin, N., 1992. Networks with learned unit response functions. In: Moody, J. et al. (Eds.). *Advances in Neural Information Processing Systems*, Vol. 4. Morgan Kaufmann, San Mateo, CA, pp. 1048–1055.
- [21] Nelson, M., Illingworth, W.T., 1990. *A Practical Guide To Neural Nets*. Addison-Wesley, Reading, MA.
- [22] Nguyen, D., Widrow, B., 1990. Improving the learning speed of two-layer neural networks by choosing initial values of the adaptive weights. In: *Proceedings of an IEEE International Joint Conference on Neural Networks*, San Diego, CA, pp.21–26.
- [23] Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1986. Learning internal representation by error propagation. In: Rumelhart, D.E., McClelland, J.L. (Eds.). *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*, Vol. 1. MIT Press, Cambridge, MA, Chapter 8.

- [24] Rizzo, D.M., Dougherty, D.E., 1994. Characterization of aquifer properties using artificial neural networks: neural kriging *Water Res.* 30 (2), 483–497.
- [25] Schmidt, W., Raudys, S., Kraaijveld, M., Skurikhina, M., Duin, R., 1993. Initialization, backpropagation and generalization of feed-forward classifiers. In: *Proceeding of the IEEE International Conference on Neural Networks*, pp. 598–604.
- [26] Stein, R., 1993. Selecting data for neural networks. *AI Expert* 2, 42–47.
- [27] Swingler, K., 1996. *Applying Neural Networks: A Practical Guide*. Academic Press, New York.
- [28] Twomey, J., Smith, A., 1997. Validation and verification. In: Kartam, N., Flood, I. (Eds.), *Artificial Neural Networks for Civil Engineers: Fundamentals and Applications*. ASCE, pp.44–64.
- [29] Upadhaya, B., Eryureka, E., 1992. Application of neural network for sensory validation and plant monitoring. *Neural Technol.* 97, 170–176.
- [30] Widrow, B., Lehr, M.A., 1990. 30 years of adaptive neural networks: perceptron, Madaline, and backpropagation. *Proc.IEEE* 78 (9), 1415–1442.
- [31] Wythoff, B.J., 1993. Backpropagation neural networks: a tutorial. *Chemometr. Intell. Lab. Syst.* 18, 115–155.

- [32] Zupan, J., Gasteiger, J., 1993. Neural Networks For Chemists: An Introduction. VCH, New York.

## CHAPTER 3

# IDENTIFICATION OF SPECTRAL LINES OF ELEMENTS WITH ARTIFICIAL NEURAL NETWORKS

### 3.1 Introduction

Spectra of various compounds and elements are taken for spectroscopic studies. In spectroscopic studies, the spectrum of the sample, taken using a spectrometer, is plotted as a graph and the various photo-peaks are identified. The spectrum of a sample contains the characteristic spectral lines of all the elements present. Thus, it is a linear superposition of the spectral lines of the elements present, but scaled. Even the weak spectral line of a particular element is obtained if the concentration of that element in the sample is high. Also, the strongest line of an element becomes unobservable if the concentration of it is very low. Under such conditions only persistent lines are obtained. A spectrum can be thought of as a linear superposition of all the weak, strong and persistent lines of all elements present in the sample. Once the spectrum is recorded it becomes a tedious task to identify the various peaks present. Spectrum contains spurious peaks as well as real peaks. Spurious peaks are due to the noise. Wavelengths corresponding to the peaks are identified and they are compared with the data in the data hand-book (Sansonetti and Martin, 2005), which is readily available. This is both time consuming and often requires manual intervention. It may lead to

errors. So they must be avoided as far as possible. Artificial neural networks (ANNs) are capable of rejecting noisy data (Haykin, 20003 and Hagan et.al. 2002). The ANN approach employs pattern recognition on the entire spectrum. This recognition is performed by a single vector-matrix multiplication that results in rapid analysis of the elements and can be used in automated systems. This helps to identify the elements present in the sample and also to test the purity of the elements. In this research work, the possibility of using ANNs to tackle such problems has been explored.

ANNs have demonstrated their benefits in analysis of various spectral regions. Resin identification was done from near infrared spectroscopic data with neural networks (Alam et. al., 1994). Keller and Kouzes (1995) have shown that Gamma spectral analysis can be successfully done with ANNs. Also the same team has done an identification of the nuclear spectrum for waste water handling (1995). Olmos with his colleagues (1994) has analysed the drift problems in gamma ray spectra with ANNs. Olmos and his team (1991) has also suggested an automation analysis of radiation spectrum using ANNs. Neural network techniques has been applied to gamma spectroscopy by Olmos et.al., 1995. Lerner and Lu (1993) have also done spectroscopic analysis with neural networks. All these research works points to the effectiveness of using ANNs for the spectral identification. Wythoff and his colleagues (1990) done spectral peak identification and recognition with multilayered neural networks. But here all have suggested that the ANNs trained can be used for automation of specific types of spectrometers. Keller and Kouzes with their colleagues (1995, 1993)

always used data generated by Monte Carlo simulations and automated this type of spectrometer. The spectra used in these investigations showed various levels of quality degradation due to calibration, salt build up etc. The task for the ANN was to learn the spectra with quality coefficient by using the knowledge of a human expert. The input to the ANN is provided as the number of channels of the spectrometer without giving any specifications to the wavelength of the obtained spectrum. An attempt is done here to take into consideration the characteristic spectral lines of elements with their wavelength and intensity in the whole visible range. The spectral lines in the visible range of Cadmium, Calcium, Iron, Lithium, Mercury, Potassium and Strontium are chosen for the project. Also the performance of the system for intensity variations and different noise levels is evaluated. This technique can be used with any type of spectrometer and a method to automate a practical system is also discussed.

## 3.2 Modelling Issues

The development of a successful ANN project constitutes a cycle of six phases (Basheer and Hajmer, 2000). The first phase is the problem definition and formulation. In the present case, the problem is to identify the spectral lines in the visible range of seven elements namely Cadmium, Calcium, Iron, Lithium, Mercury, Potassium and Strontium. Second is the system design phase. Usually supervised learning is suitable. Analyzing the spectrum of elements taken, from data hand-book, it is evident that the spectral lines occur in discrete values at different wavelengths as in Fig.3.1. These consist of strong, persistent and weak lines. Spectra of



various samples are taken for spectroscopic studies. They consists of various photo-peaks which are characteristic spectral lines of the elements which constitute the sample. Also it is not necessary that all characteristic lines of each constituent element be found in the spectrum. But the probability of occurrence of the persistent lines of the elements is very high. Thus the spectrum taken is a linear superposition of the spectra of the constituent elements in the sample. Indeed, the photo-peaks do not have the same relative intensity as specified in the data handbook. They are scaled.

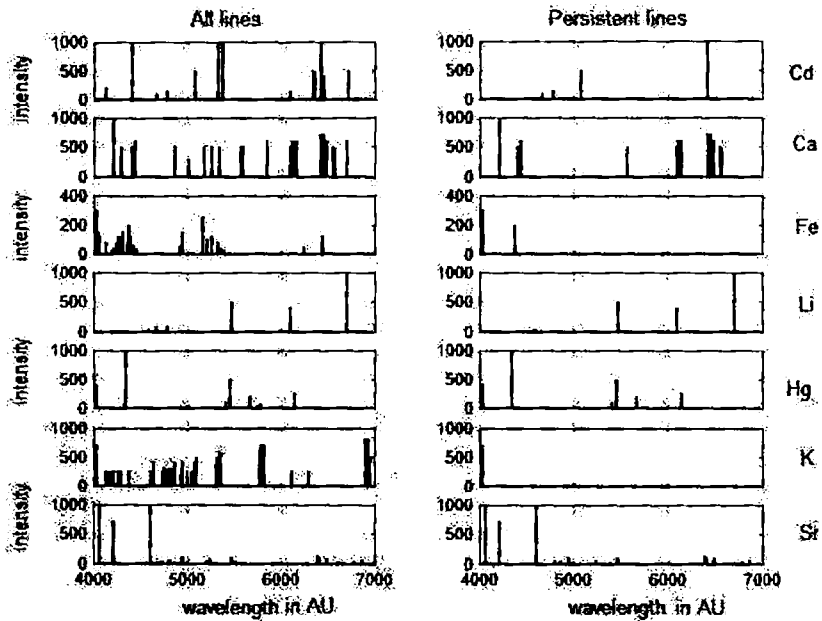


Fig.3.1 Spectral lines of elements with (a) all lines and (b) persistent lines

So, if  $e_i$  is the spectrum of element  $i$  in the sample, then the intensity of the characteristic line of the sample  $S$  can be given as:

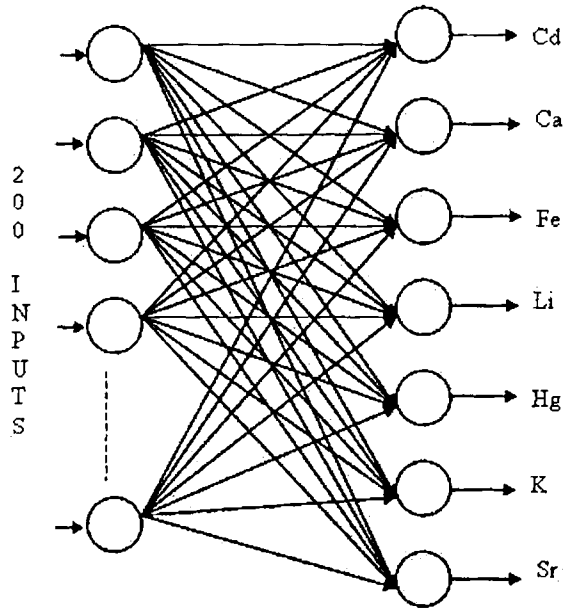
$$S = \sum_i \alpha_i e_i \quad (3.1)$$

where  $\alpha_i$  is the scaling factor of the relative intensity of the spectral lines of element  $i$ .

The output has a linear response with the input. Therefore, the classification system should have a linear response with respect to the input. An ANN designed to have a linear response employs linear activation functions. A feed forward ANN that implements linear activation functions can be reduced to a network with a single input layer and single output layer. The ANN used in the present application has a single input and single output layer as illustrated in Fig.3.2. Two ANN paradigms were studied for implementing the linear response: the linear perceptron and optimal linear associative memory. Linear perceptron is one of the oldest ANN paradigms. It originally sparked interest in the pattern recognition community in the late 1950s and early 1960s (Rosenblatt, 1958). However, it was unable to solve pattern recognition problems that were not linearly separable. Here, for spectral identification, the neural network can be trained using linear perceptron models or using optimal linear associative memory (OLAM) algorithms. A linear perceptron does not converge to accurate results and OLAM is most suited for such applications as shown by Keller and Kouzes (1995).

The optimal linear associative memory (OLAM) approach is based on a simple matrix associative memory model (Kohonen, 1972, 1989). It was developed in the early 1970s as a content addressable memory and is useful in situations where the input consists of linear combinations of known patterns. It is an improvement over the original

matrix memory approach in that it projects an input pattern onto a set of orthogonal vectors where each orthogonal vector represents a unique pattern. With linear activation functions, the training is a straight forward matrix orthogonalization process where each pattern from the training set is made to project onto a separate, unique orthogonal axis in the output space (Keller and Kouzcs, 1995) .



*Fig.3.2 An ANN to identify the elements.*

### OLAM Weight Specification

Step 1. Form matrices of spectra. Arrange spectra as columns in an  $n \times p$  dimensional matrix  $\underline{X}$ , where  $n$  is the number of inputs and  $p$  is the number of elements and target as columns in an  $p \times p$  dimensional matrix  $\underline{T}$ .

Step 2. Generate inverse of the spectral matrix  $\underline{X}$ . Since  $\underline{X}$  is generally not a square matrix, a pseudo-inverse technique is used to generate  $\underline{X}^\dagger$ .

( $\dagger$  indicates pseudo-inverse)

Step 3. Form the synaptic weight matrix.

$$\underline{W} = \underline{TX}^\dagger$$

The third phase of an ANN development project is system realization. The spectral lines data from the handbook for each element is as shown in Fig.3.1. In the system realization phase, the number of input neurons and the number of output neurons are to be determined. The problem specification determines both (Hagan et. al., 2002). Usually the number of output neurons is taken as the number of problem outputs. Since there are seven elements to be identified the number of output neurons is taken as seven. Next phase is to determine the number of input neurons. In this particular problem the number of input neurons is determined by training and testing. Two sets of data are given for testing: One, a set of noisy data and other the persistent lines. The probability of occurrence of persistent lines (ultimate lines) is the highest. Therefore, it should be identified in any worse condition, even though it is few in number.

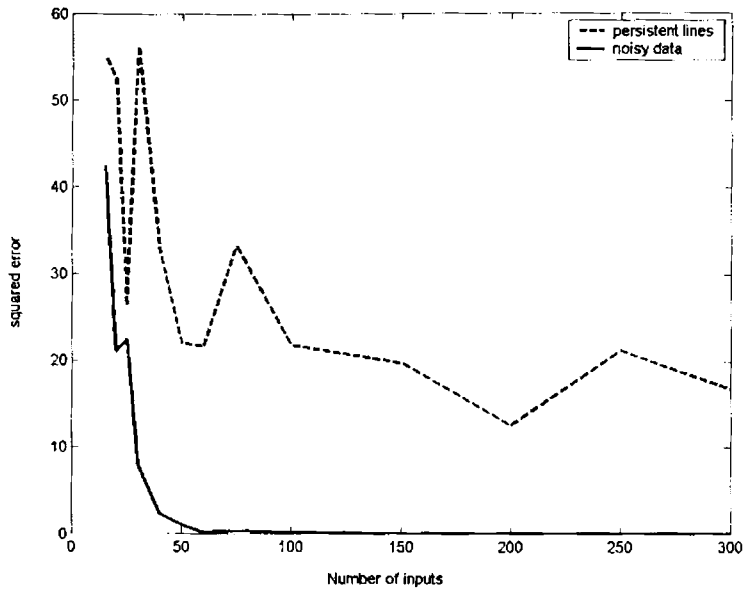
The data is scanned with a resolution of  $1\text{\AA}^0$ . This is to ensure discretion with spectral lines which are very close to each other. In the nanometre scale they are treated as the same line. There are about 3000 wavelength points with their intensities. As seen in Fig.3.1, most of these points have intensity values of zeroes. To be more precise, consider the

element Cadmium with its characteristic spectral lines in the range 400 - 500nm, given by Table 3.1.

Relative Intensity	Wavelength in $\text{\AA}$
200	4134.768
1000	4415.63
100	4678.149
150	4799.912

*Table 3.1: The characteristic spectral lines for Cadmium in the range 400-500nm.*

When scanned for a resolution of  $1\text{\AA}^0$ , up to the wavelength  $4134\text{\AA}^0$ , the intensity value is zero and at  $4135\text{\AA}^0$  it is 200, then up to  $4415\text{\AA}^0$  it is again zero and again at  $4416\text{\AA}^0$  it is 1000 and so on. When most of the data contains very low values or zeroes the learning algorithms will not converge to accurate results. So a reduction in data is required. The most common method in data reduction is to find the area under the curve. Area is taken by considering a polygon. It is to be determined the optimum number of wavelength points that is required to make the polygon so as to get a better result from the trained ANN. The data is divided into equal parts and the area is taken for each segment. As an example, consider that the data is segmented into 150 equal parts each of 20 wavelength points and their intensities. Area is taken by considering polygon with these 20 points. Therefore the data is now reduced from 3000 data points to 150 data points. The data is then normalized, so that there are now 150 input nodes with normalized data. This kind of data reduction is done for each element and is arranged in a matrix form. The



*Fig.3.3. Error plot to determine the number of input nodes*

matrix,  $X$  (as in the OLAM algorithm), is now having 150 rows and 7 columns, each column specifying an element. Since 7 elements are to be identified, the target matrix  $T$  is a  $7 \times 7$  matrix. By taking the pseudo inverse of  $X$ , it became a  $7 \times 150$  matrix. The weight matrix,  $7 \times 150$ , is calculated as per the OLAM algorithm given. Testing of the result is also done with the persistent line data and the noisy data. For the persistent line and noisy data of each element, the data is again scanned with  $1A^0$  resolution and the data is segmented into 150 equal parts and the area is taken. The data is normalized. The output is verified for these inputs and the error is calculated. This is done for segments of varied lengths. For noisy data, the error goes on decreasing as the number input nodes

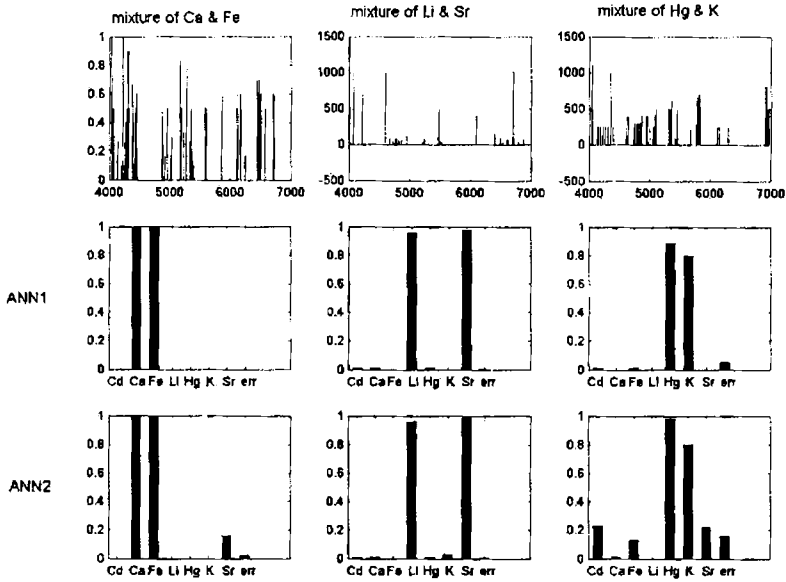
increases. But when the number of input segments is 200, the system has minimum error for the identification of persistent lines, shown in Fig.3.3. Therefore, the number of inputs to the system is 200. Thus the network is ready for training.

The goal of the training is to learn an association between the spectra and the labels representing the spectra. The training process for the OLAM is a non-iterative process and it converges very fast. The weight matrix is obtained using pseudo-inverse rule. Two types of ANNs are trained, one with all the spectral lines (ANN1) and the later with the persistent lines (ANN2). Only the visible range (400-700nm) of the spectrum is considered. The persistent lines are very few in number. For elements like potassium there are only two persistent lines in the required range, as shown in Fig.3.1, whereas, there are about 44 spectral lines for potassium in the visible range. The ANNs are tested with known samples and unknown samples.

### 3.3 The Results

Now the system is to be tested. Spectra of mixtures were generated by combining spectra of different elements. Random noise is also added to the mixture. The data is scanned with a resolution  $1\text{Å}^0$  and is segmented into 200 equal parts and area is taken. The data thus got is normalized and fed to the system. The output got for each ANN for 3 different mixtures is as shown in Fig.3.4. The first sample is a mixture of calcium and iron in pure form without any noise. But the other two samples, one a mixture of lithium and strontium and other a combination

of mercury and potassium, are noisy data. ANN1 gives a consistent performance than ANN2 even in noisy environment. This is because the number of observable spectral lines in the visible range is very high compared to the persistent lines. For the third sample which is a mixture



*Fig.3.4 Output of the ANN for different samples*

of Hg and K, ANN1 gives more accurate result than ANN2. For K, there are only two persistent lines in the visible range and these lines are very close to each other also. To identify K with ANN2 is a very tedious task and most of the time it leads to errors. ANN1 on the other hand gives a very consistent result. In this context, the need of enough spectral lines in the required range for training is emphasized.

More results are shown in Table 3.2 also. The identification of Fe also gave some errors. From Fig.3.1, the highest relative intensity of Fe is



A mixture of Fe & Hg								
	Cd	Ca	Fe	Li	Hg	K	Sr	error
ANN1	0	0	1	0	1	0	0	0
ANN2	0	0.01	1	0	1	0	0.03	0.001
A mixture of Hg & Sr								
ANN1	0	0	0.01	0	0.99	0	0.99	0.0003
ANN2	0	0	0	0	1	0.02	0.99	0.0005
A mixture of Cd & Sr								
ANN1	1	0	0.01	0	0.01	0	1	0.0002
ANN2	0.99	0.02	0	0	0.01	0.03	1	0.0015
A mixture of Ca & Li								
ANN1	0	1	0	1	0.01	0	0.01	0.0002
ANN2	0.01	0.99	0.01	1	0.01	0	0.13	0.0173
A mixture of Li & K								
ANN1	0.01	0	0	1	0	0.8	0.02	0.0405
ANN2	0.23	0.01	0.11	0.99	0	0.8	0.23	0.1581
A mixture of Ca (peak reduced to 80%) & Hg (peak reduced to 70%)								
ANN1	0	0.8	0	0	0.69	0	0.01	0.0002
ANN2	0.01	0.79	0	0	0.73	0	0.11	0.0132

**Table3.2: Output obtained for different samples. Each column represents different elements. RMS error is listed in the right-hand column**

only 400 when compared to other elements having highest value of 1000. In the training phase, since the data is normalized, Fe requires no enhancement. But when the spectrum of Fe is combined with that of others, the intensity of the spectral lines of Fe becomes very low. So the spectrum of Fe is enhanced before combining.

ANN1 correctly identifies most of the elements fed to it. But ANN2 had hard times in differentiating potassium with strontium. In certain times, ANN1 shows presence of mercury, which is not present. Hg has only 15 spectral lines in the required range and most of them have very low intensity. Certain spectral lines of Hg coincide with the spectral lines of elements like K. However, the errors with ANN1 were always smaller than the ANN2.

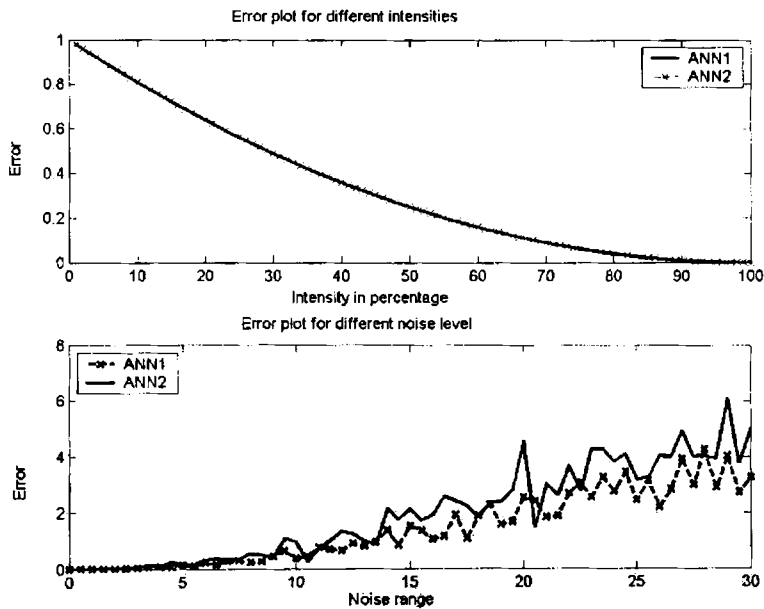


Fig.3.5 Error plots for (a) different intensities (b) noise level

The performance of the networks is to be checked. Testing is done with varying relative intensities and noise levels. First the intensities of the lines are reduced. Here, no noise is added and all the spectral lines in both data sets are considered but with reduced intensity. With the intensity as the original, the outputs of the ANNs were 1. When the intensity is reduced, the output also correspondingly reduces. As shown in table 3.2, when the intensity of the spectral lines of Ca is reduced to 80%, ANNs output is only 0.8. The error plot for the output got for different intensity levels are shown in Fig. 3.5(a). It is evident that the performances of both the networks are the same when the intensity is reduced. When the intensity is reduced below 70% of the original relative intensity value, then the network gives errors. Here, it is worthwhile to note that all spectral lines in the visible range are considered.

The networks are now tested with noise. The output for different noise levels are shown in Fig. 3.5 (b). Random noise is added to the data at different noise levels. The graph shows the average error for 1000 such data. Here the performance of ANN1 is better than ANN2. This can be seen in Fig. 5 also. When the noise levels are very low, the networks output is not affected. But as the noise levels are increased, the output of the network shows errors. As shown in the Fig. 3.5(b), noise levels cannot be increased beyond a factor of 7 for both ANNs. Noise levels in practical cases will not be very high. From this it is clear that random noise with normal distribution will not affect the performance of the network. Only if the noise amplitude is increased to 7 times its original value, some error occurs, which is not a practical case. ANN1 is preferred over ANN2.

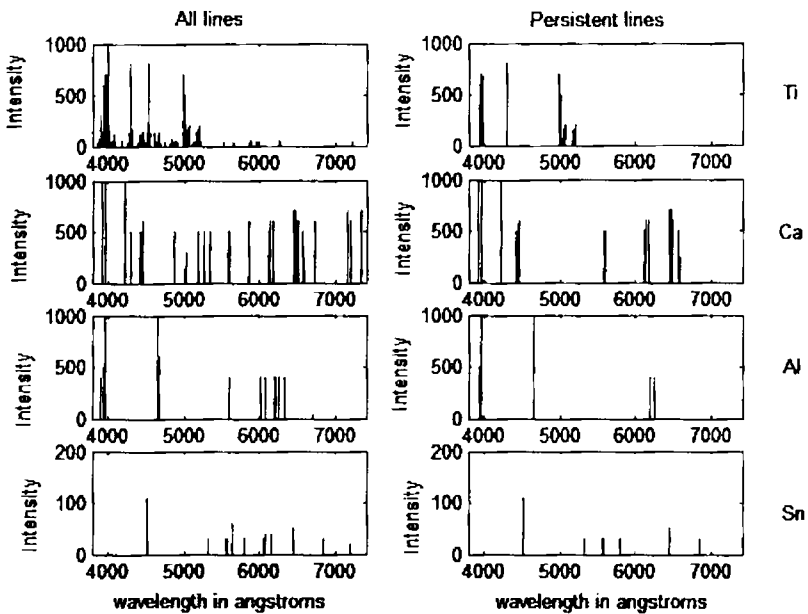
ANN1 is trained with all spectral lines but ANN2 with the persistent lines only which may lead to errors.

The initial results of our research have demonstrated the pattern recognition capabilities of the neural networks. It also emphasized the need for a large number of spectral lines in the desired range for the accurate classification of elements. ANN1, which is trained with more number of spectral lines than ANN2, gives a better performance. This is because ANNs can easily generalize when data is large. The classification is attributed to the orthogonalization process used by the OLAM during training. Since this training is a non iterative process, the OLAM offers a substantially shorter training time. One of the disadvantages of the OLAM, is that all the spectral lines of each element, weak, strong and persistent within the visible range, are used for training. Good results are obtained when all the lines are considered. But in a practical case, it is not possible to obtain the whole spectral lines. Further work is directed in this direction, to train a network with the characteristic lines of the elements and to observe the performance of the network for practical cases.

### **3.4 An Automated System**

The success of the identification of spectral peaks with the neural network led the approach for the automation of a practical system (Saritha and VPN Nampoore, 2009, 2002). It now turns out that a system is set up so that when a spectrum is fed to it, it will identify all the elements present in the sample by recognizing the elements learnt by it. Preliminary results are good enough to consider this method for automating spectral identification. Spectrum recorded with a CCD camera coupled to a

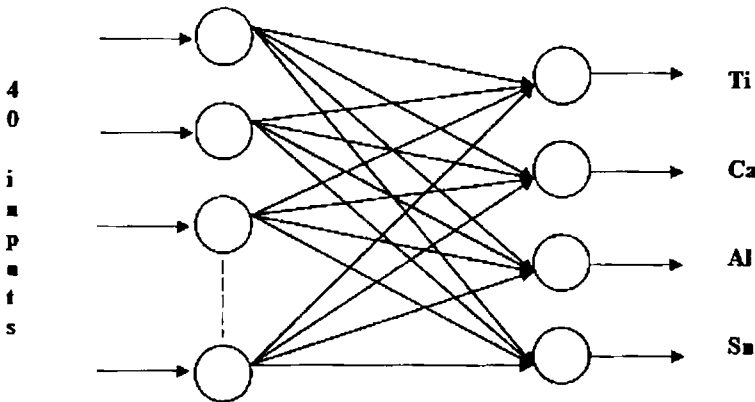
spectrograph having a grating blazed at 750nm with 1200 grooves/mm and using the fundamental emission of Nd: YAG laser having 10ns pulse width was employed for the investigations. In the present investigation, the characteristic spectral lines of elements with their wavelength and intensity in the whole visible range are taken into consideration.



*Fig.3.6 Spectral Lines of Elements*

The spectral lines of Titanium, Calcium, Aluminium and Tin in the visible region are chosen for the studies. Two simple ANNs are trained. One of them is trained with all the characteristic spectral lines of elements namely Titanium, Calcium, Aluminium and Tin and the latter using the persistent lines of these elements. With the help of these two networks, it became possible to identify the elements present in a practical

spectrum. Usually supervised learning is suitable. Analyzing the spectrum of elements taken, from data hand-book (Sansonetti and Martin, 2005), it is evident that the spectral lines occur in discrete values at different wavelengths as in Fig.3.6. These consist of strong, persistent and weak lines.



*Fig.3.7 The ANN Model*

The output has a linear response with the input. Therefore, the classification system should have a linear response with respect to the input. An ANN designed to have a linear response employs linear activation functions. A feed forward ANN that implements linear activation functions can be reduced to a network with a single input layer and single output layer. The ANN used in the present application has a single input and single output layer as illustrated in Fig.3.7. This can be trained using linear perceptron models or using optimal linear associative memory (OLAM) algorithms. A linear perceptron does not converge to accurate results and OLAM is most suited for such applications as shown by Keller and Kouzes(1995)

### 3.5 The Approach

Now we will determine the number of input nodes and the number of output nodes for the ANN. Since there are four elements to be identified, the number of output nodes is 4. The number of input nodes is determined by actual training and testing. For training, the data from the handbook is taken. For testing, the persistent line data from the data handbook and the data taken by the spectrometer are used. Spectrum taken with a CCD camera coupled to a spectrograph having a grating blazed at 750nm with 1200 grooves/mm and using the fundamental emission of Nd: YAG laser having 10ns pulse width was taken for the studies.

Relative Intensity	Wavelength in $\text{\AA}$
400	3900.675
500	3944.006
1000	3961.520

*Table 3.3 The characteristic spectral lines of Aluminium in the range 380-420nm*

The wavelength range 380-740nm is split into 9 spectra each of 40nm span, since a 40nm grating is used. The example of such a spectrum extending from 380-420nm is as shown in Fig.3.8. The table3.3 shows the characteristic spectral lines for Al in this range. There are only 3 lines in this range out of which 2 are persistent lines. This data is scanned with a resolution of  $1\text{\AA}$ . This is to ensure discretion with spectral lines which are very close to each other. In the nanometre scale they are treated as the

same line. From the table it can be seen that, when scanned with a resolution of  $1\text{\AA}$ , up to  $3900\text{\AA}$  the relative intensity is zero and at  $3901\text{\AA}$ , the relative intensity.

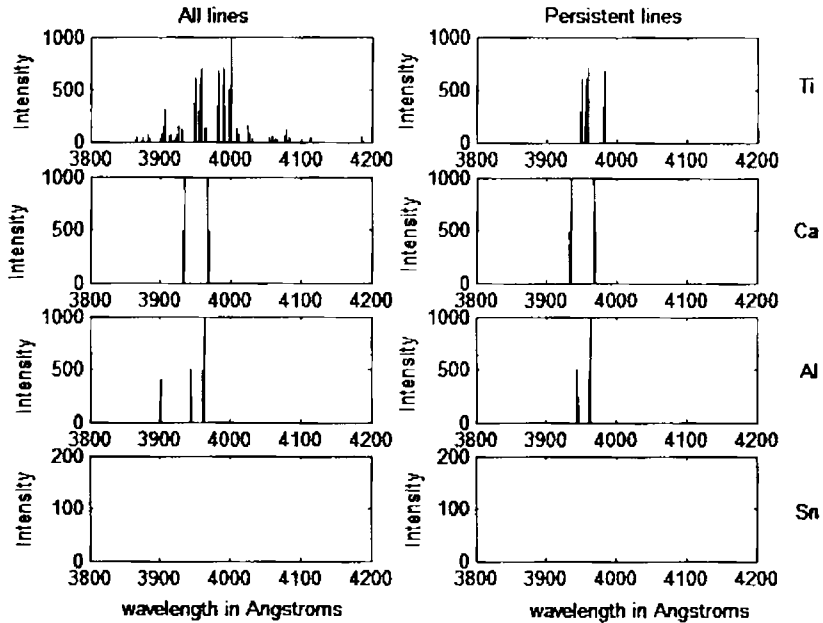
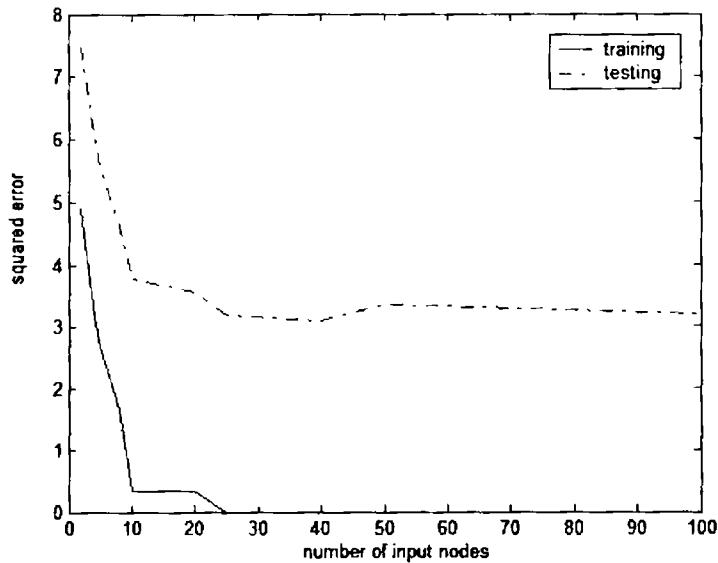


Fig.3.8 Example of the split spectrum of 40nm

is 400. Then up to  $3943\text{\AA}$  it is zero and at  $3944\text{\AA}$  it is 500. In the range 380-420nm, there are about 400 wavelength points. Of these 400 wavelength points, most of them have relative intensity zero. ANN algorithms do not converge to accurate results when most of the data are zeros or very low values. So a reduction in data is required. The easiest way of data reduction is to find the area under the curve. The 400 data points are divided into 80 equal parts of 5 data each. A polygon is considered with each of these 5 data points and the area of the polygon is



taken and the data is normalized, so that the number of input nodes for the ANN is 80. This process is done to all the 4 elements and to all the 9 spectra of each element.



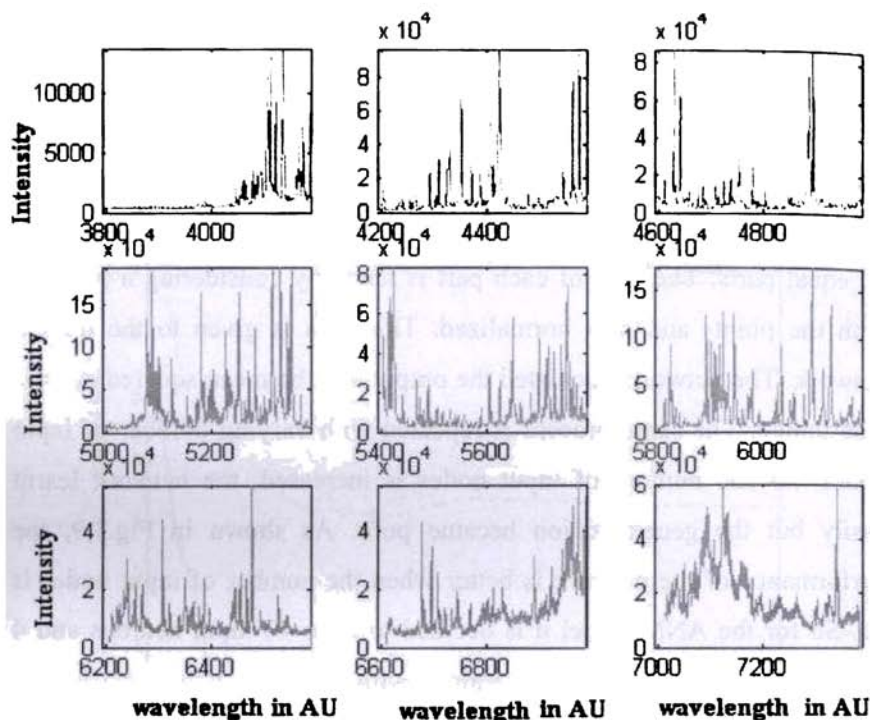
*Fig.3.9. Error plot to determine the number of input nodes*

In certain ranges, there is no characteristic spectral line for a particular element. For instance, as shown in Fig.3.8, there is no characteristic spectral line for Sn in the range 380-420nm. In such cases, that element is discarded in that particular range. This is because the relative intensity is zero for all the wavelengths in that range. From the OLAM weight specification given, it is required to calculate the pseudoinverse of the input matrix  $X$ . If one of the columns of the input matrix  $X$  becomes zero, then it is singular and no inverse exists. This is true with the persistent line data also as shown in Fig.3.8.

The ANN now has 80 input nodes for each element, X is a 80x4 matrix, and 4 output nodes, T is a 4x4 matrix. The weight matrix W, a 4x80 matrix, is determined as per the OLAM weight specification. The testing data, the persistent line data and the actual data from the spectrometer are also scanned at a resolution of  $1\text{Å}^0$  and is segmented into 80 equal parts. The area of each part is taken by considering a polygon with the points and it is normalized. This data is given to the trained network. The network calculated the output and the mean squared error is determined. The same process is repeated with varying number of input nodes. As the number of input nodes is increased, the network learnt easily but the generalization became poor. As shown in Fig.3.9, the performance of the network is better when the number of input nodes is 40. So for the ANN model it is decided to have 40 input neurons and 4 output neurons. The network is trained with OLAM weight specification and the weight matrix is determined

### **3.6 The Output**

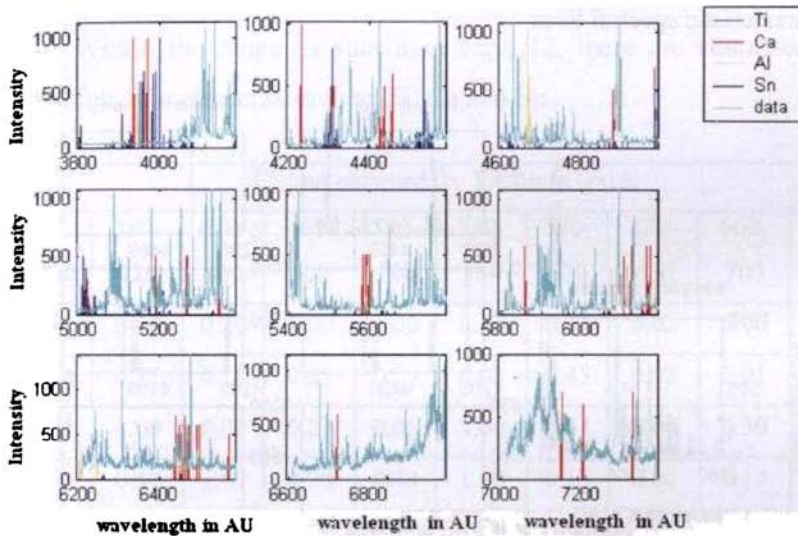
The artificial neural network with 40 input neurons and 4 output neurons is trained and is now ready to automate the spectra taken with the spectrometer. Here, it is to be noted that the ANN is trained with the actual data taken from the data handbook (Sansone and Martin, 2005). No spectrum from any practically obtained spectrometer is given during the training phase. It is used only for testing. The spectra of pure Titanium, Titanium oxide and Aluminum oxide are taken.



*Fig. 3.10 Sample spectrum of Titanium oxide taken with a CCD camera coupled to a spectrograph having a grating blazed at 750nm with 1200grooves/mm and using the fundamental emission of Nd: YAG*

Spectrum for the studies is recorded with a CCD camera coupled to a spectrograph having a grating blazed at 750nm with 1200 grooves/mm and using the fundamental emission of Nd: YAG laser having 10ns pulse width. Fig. 3.10 shows a sample spectrum of titanium oxide. Since a 40nm grating is used, each spectrum is having a span of 400  $\text{\AA}$ . The spectrum shows various photo-peaks. These consist of original peaks and spurious peaks. Here the neural network is trained to identify only 4 elements, viz., titanium, aluminium, calcium and tin. It is

the purpose of the neural network to recognize these elements from the spectrum shown in Fig.3.10. Here the pattern recognition capability of the neural network is made use of.



**Fig.3.11 Sample spectrum of Titanium oxide taken with a CCD camera with all the characteristic spectral lines of elements trained with the ANN.**

The neural network can recognize patterns even from a noisy background. Once the network is trained efficiently, it is robust and reliable at any worse conditions of the input, unless the input is highly distorted. The trained ANN is now tested with a practical data given in Fig.3.10.

The sample spectrum of titanium oxide with the occurrence of all the characteristic spectral lines of elements trained with the ANN is as shown in Fig. 3.11. Some photo-peaks of the spectrograph spectrum is coinciding with the characteristic spectral lines of certain elements and there are photo-peaks which are spurious also. The spectrum of Titanium

oxide in the range 4200 - 4600  $\text{\AA}$  is as shown in Fig.3.12 with the characteristic spectral lines of the elements and the persistent lines in particular in the same range is also given. In the specified range Al has no characteristic spectral lines.

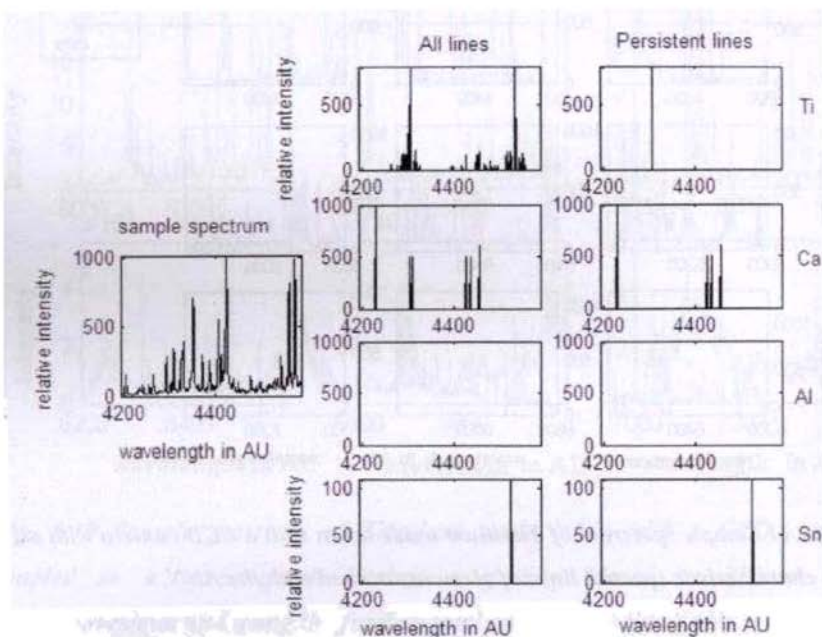


Fig. 3.12 Sample spectrum of Titanium oxide with the characteristic spectral line of elements and persistent lines in the range 4200 – 4600  $\text{\AA}$

With the coincidence of certain photo-peaks obtained in a 40nm span with the characteristic photo-peaks of certain elements, one cannot conclude that a specific element is present in the sample. The confirmation is obtained from the spectrum taken from other wavelength spans and also from the occurrence of the persistent lines in the obtained spectrum. So the spectra for a certain range is required for the

conclusions. Here, the whole visible range, with 9 spectra each of 40nm span, is considered. Each of the 9 spectra is scanned with a resolution of 1Å<sup>0</sup>. The scanned data is divided into 40 equal parts and the normalized area is taken. This data is fed to the trained neural network.

Within the range as shown in Fig.3.12, there are characteristic spectral lines for elements such as Ti, Ca and Sn.

Output obtained for Titanium Oxide									
nm	380-420	420-460	460-500	500-540	540-580	580-620	620-660	660-700	700-740
Ti	0.12	0.29	1.00	1.00	0.13	0.31	0.02	0.00	0.00
Ca	0.77	0.18	0.85	0.29	0.00	0.45	0.02	0.01	0.04
Al	1.00	0.00	0.27	0.00	1.00	0.34	0.60	0.39	0.00
Sn	0.40	1.00	0.00	0.43	1.00	0.15	0.00	0.19	0.35
Output obtained for Pure Titanium									
Ti	0.06	0.21	0.51	0.91	0.07	0.26	0.39	0.00	0.00
Ca	0.19	0.12	0.50	0.45	0.00	0.20	0.43	0.04	0.85
Al	0.00	0.00	0.07	0.00	1.00	1.00	1.00	0.09	0.00
Sn	0.36	0.00	0.00	0.72	1.00	0.12	0.00	0.09	0.25

**Table 3.4** Output obtained from the ANN for the 9 spectra of Titanium oxide and pure Titanium

The spurious peaks obtained in the sample are misclassified as the elements which are not present in the sample. But the result was not encouraging as shown in Table 3.4. In order to overcome this problem another neural network is designed with the persistent lines only. Persistent lines of all the elements in the desired range is taken and

processed as discussed and the normalized area is given as the input to the ANN. The weight matrix is determined using the OLAM weight specification.

Sample spectrum taken with the CCD camera and considering the persistent lines only is as shown in Fig.3.13. Literally speaking, one can conclude the presence of a particular element in a sample by testing for its characteristic spectral lines and also making sure the presence of its persistent line in the taken spectrum. With the knowledge of these two things only recognition of the elements can be satisfactorily done. Hence, two artificial neural networks are made to solve the problem. The two artificial neural networks, ANN1 which is trained with all the spectral lines and ANN2 trained with the persistent lines only, are used to determine the elements present in the given sample. Processed data from each of 40 inputs for the 9 spectra is given to ANN1.

The probability of occurrence of an element is determined from the obtained output. For this probability, a threshold is kept. If the probability is above this threshold, the second ANN, ANN2, is used to check the presence of the persistent lines of the element. If the persistent line is also present, then it could be inferred that the element is present in the sample. The block diagram is shown in Fig.3.14. This technique gave a good result for the entire sample spectrum fed to it. The 9 spectra ranging from 380-740 nm taken using the spectrometer are processed as each of 40 inputs and fed to ANN1. The output is tabulated as in table 3.4. From the table, the probability of occurrences of each element is taken.

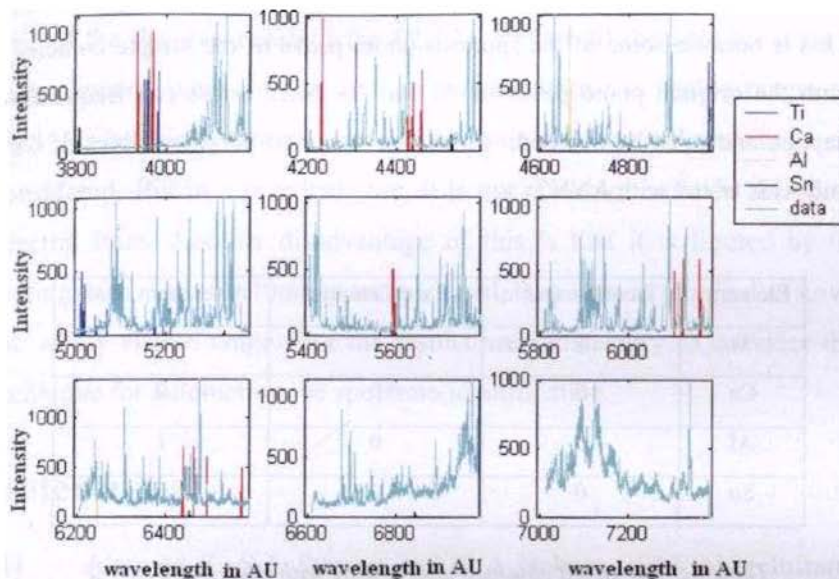


Fig.3.13 Sample spectrum of Titanium oxide taken with a CCD camera with only the persistent spectral lines of elements trained with the ANN

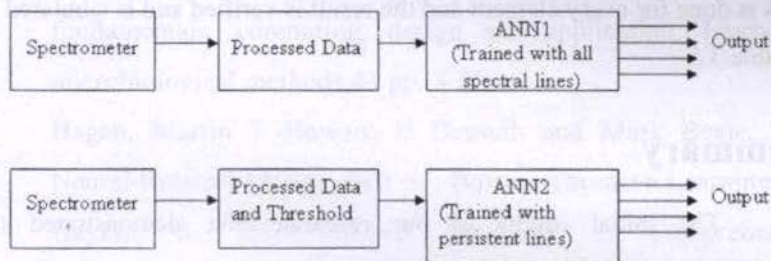


Fig.3.14. Block diagram showing the spectral line identification

It can be seen that in certain ranges, the probability of occurrence of some elements is high. For instance, it is seen that in the range 380-420



for Titanium oxide, the neural network shows the presence of Ca and Al. This is because some of the spurious photo peaks of the sample coincide with the original photo peaks of Al and Ca. Such errors can frequently happen and must be avoided. So the existence of persistent lines of Ca and Al is tested with ANN2.

Elements	Titanium oxide	Pure Titanium	Aluminum oxide
Ti	1	1	0
Ca	0	0	0
Al	0	0	1
Sn	0	0	0

*Table 3.5 The result obtained after testing with ANN1 and ANN2*

When tested with ANN2, it gave a result of 0 ruling out the possibility of occurrence of Ca and Al, showing that a spurious peak is misclassified. This is done for every element and the result is verified and is tabulated as in table 3.5.

## Summary

The initial results of our research have demonstrated the capability of artificial neural networks to identify elements even from a noisy spectrum. With the help of two ANNs, it became possible to identify the elements present in the sample from the obtained spectrograph. The classification is attributed to the orthogonalization process used by the OLAM during training. Since this training is a non

iterative process, the OLAM offers a substantially shorter training time. One of the disadvantages of the OLAM, is that all the spectral lines of each element, weak, strong and persistent within the visible range, are used for training. Good results are obtained when all the lines are considered. But in a practical case, it is not possible to obtain the whole spectral lines. Another disadvantage of this is that it is limited by the grating used. Since a 40nm grating is used, 9 spectra is required to cover the whole visible range. But the results are satisfactory to consider this technique for automating the spectrum identification.

## References

- [1] Alam M K, S L Stanton and G A Hebner, 1994. Near Infrared Spectroscopy and Neural Networks for Resin Identification, Spectroscopy, pp 30-40
- [2] Basheer I A and M Hajmeer, 2000 Artificial neural networks: fundamentals, computing, design and application, Journal of microbiological methods 43 pp. 3-31
- [3] Hagan, Martin T Howard B Demuth and Mark Beale, 2002 Neural Network Design, first ed., Boston, Thomson Learning
- [4] Haykin, S., 2003. Neural Networks: A Comprehensive Foundation. Second Edition, Pearson Education
- [5] Keller, Paul E., and Richard T Kouzes, 1995 Gamma spectral analysis via neural networks , IEEE trans. on Nuclear Science pp. 341- 345
- [6] Keller, Paul E., Lars J Kangas, Gary L Troyer, Sherif Hashem, Richard T Kouzes, 1995 Nuclear spectral analysis via artificial

- neural networks for waste handling, IEEE trans. on Nuclear Science vol. 42. pp. 709- 715
- [7] Keller, P E, R T Kouzes and L J Kangas, 1993 Applications of Neural Networks to Real- Time Data Processing at the Environmental and Molecular Sciences Laboratory, In conference record of the Eighth conference on real-time computer applications in Nuclear, Particle and Plasma Physics, Vancouver, BC, Canada, pp. 438-440
- [8] Kohonen T, 1972 Correlation Matrix memories, IEEE Transactions on Computers, vol. C-21, pp 353
- [9] Kohonen, T Self Organization and Associative Memory, 1989, third ed., New York: Springer-Verlag.
- [10] Lerner J M and T Lu, 1993. Practical Neural Networks Aid Spectroscopic Analysis, Photonic Spectra, pp. 93-98
- [11] Olmos P, J C Diaz, J M Perez, P Aguayo, P Gomez and V Rodellar, 1994 Drift Problems in the Automatic Analysis of Gamma Ray Spectra Using Associative Memory Algorithms, IEEE trans. on Nuclear Science, vol. 41, pp. 637-641
- [12] Olmos P, J C Diaz, J M Perez, P Gomez, V Rodellar, P Aguayo, A Bru, G Garcia-Belmonte, and J L de Pablos , 1991 A New Approach to Automatic Radiation Spectrum Analysis , IEEE trans. on Nuclear Science vol. 38. pp. 971- 975
- [13] Olmos P, J C Diaz, J M Perez, G Garcia-Belmonte, P Gomez, and V Rodellar, , 1992. Application of Neural Network Techniques in Gamma Spectroscopy, Nuclear Instruments and Methods in Physics Research, vol. A312, pp 167-173

- [14] Rosenblatt F, 1958. Two theorems of statistical separability in the Perceptron, in Mechanisation of Thought Process, Proceedings of symposium No.10, National Physical Laboratory, London, vol I pp. 421-456
- [15] Sansonetti J.E and W. C. Martin, 2005 Handbook of Basic Atomic Spectroscopic Data, J. Phys. Chem. Ref. Data, Vol. 34, No. 4, pp. 1559 -2259
- [16] Saritha M and V.P.N. Nampoori, 2009. Identification of spectral lines of elements using artificial neural networks Microchemical Journal 91 pp. 170–175
- [17] Saritha M and V. P. N. Nampoori, 2002. Peak Identification in Optical Spectrum using Artificial Neural Networks. Proc. of DAE BRNS National Laser Symposium, pp. 578-580.
- [18] Wythoff B J, S P Levine, and S A Tomellini, 1990. Spectral Peak Verification and Recognition using a Multilayered Neural Network, Analytical Chemistry, pp 2702-2709



## CHAPTER 4

# LEARNING BASED SUPER-RESOLUTION OF BINARY IMAGES WITH DISCRETE COSINE TRANSFORMS

### 4.1 Introduction

Super-resolution is the process of obtaining an image at a resolution higher than that afforded by the physical sensor. Super-resolution has been used in obtaining high quality image prints and has found applications in areas such as surveillance and automatic target recognition. This chapter aims the issue of image magnification (in optical images the issue is referred as interpolation, zooming, enlargement etc) from a finite set of collected data sampled at Nyquist rate.

Currently, there are many algorithms capable of achieving super-resolution. The best known super-resolution algorithms is first published by Gerchberg (1974) and later by Papoulis (1975). For the reconstruction of the image  $\hat{f}(x, y)$ , these algorithms rely on the prior knowledge of the original object which is the major disadvantage of these algorithms. In most of the cases a priori knowledge of the object is not known.

A different superresolution algorithm, known as Poisson-maximum a posteriori (Poisson-MAP or PMAP), was developed by Hunt (1974). It is also an iterative algorithm that constructs the Bayes MAP

estimate of the object, assuming Poisson statistics. Multiframe versions of this algorithm have also been developed (Lucy, 1974).

The formulation of PMAP is similar to another iterative Bayesian algorithm, developed independently by Richardson (1972) and Lucy (1974). The Richardson-Lucy algorithm which is known as the expectation maximization algorithm is in the class of maximum likelihood estimators. As was with the PMAP algorithm, a Poisson model for the image is assumed. Another superresolving algorithm is the maximum entropy method which is a popular technique amongst the astronomical community (Frieden and Burke, 1972).

All the above mentioned algorithms are iterative methods. Several noniterative algorithms have also been developed such as those by Byrne et. al. (1983) and Darling et. al. (1983) which perform regularized approximation in a weighted Hilbert space by incorporating a priori information. More further efforts include method based on singular value decomposition that was developed by Walsh and Nielsen-Delaney (1994) and a variant of a nonlinear interpolative vector quantizer by Sheppard et. al.(1998).

Candocia and Principe (1999) proposed a method using linear associative memory (LAM). They implemented a LAM via vector quantization algorithm (VQ) to find the best mapping between the low-resolution image and high resolution image, thereby capturing the information across the scales with the assumption that the information embodied in the code book vectors and LAM describes a mapping between a low-resolution neighborhoods and its high resolution counter

part. However, they ensure no warranty for the analysis that mathematically supports their assumption.

Another nonlinear interpolation scheme is the sub-pixel edge localization developed by Kris Jensen and Dimitris Anastassiou (1995). Also T Q Pham and his fellowmen (2006) mention an example-based super-resolution in the discrete cosine transform (DCT) domain. All the aforementioned algorithms are iterative algorithms, in which a single blurred image is operated on repetitively until an acceptable estimate is obtained on the basis of some criterion. These methods are suited for off line processing, but are ill-suited for real-time operation requiring a high through-put rate. Thus the search for noniterative super-resolution algorithms is of real practical importance.

There are only a few works available relating super-resolution of images with neural networks. Concerning image restoration most often considered architecture is the Hopfield network which is again an iterative method like the super-resolution algorithms. In contrast to the feedforward architecture of the multilayer perceptron (MLP), this network is a single layer network with complete interconnections. The output of each node feed backs to every other node in the network, even possibly to itself. The Hopfield net is in the class of dynamic networks, in the sense that the node equations are described by differential or difference equations (Chen et. al., 1995, Hopfield, 1982, 1984).

Perhaps the earliest discussion on the use of a Hopfield network for image restoration was by Abbiss et. al. (1988, 1991), who discussed the plausibility of this method without presenting any results. The first documented results on actual images were presented by Zhou et. al.



(1988), who used a Hopfield network to restore a gray scale image degraded by a known shift-invariant blur function and signal independent white noise. Their basic approach was the same as that outlined by Abbiss et. al. (1988, 1991), namely, to define a regularized error function, which was mapped term by term to the Hopfield energy function. The energy-reduction properties of the network were then used to minimize the image estimation error.

This research inspired many others to study the Hopfield net under different image-restoration situations and to improve on the original Hopfield design. Zhang et. al., (1991) proposed using multistate neurons to avoid the exploding number of neurons needed when gray levels are represented as a simple sum of binary neurons. They demonstrated performance against an image degraded by linear motion blur plus additive noise. Paik and Katsaggelos (1992) also considered motion blur with and without additive Gaussian noise. An improvement on Paik and Katsaggelos (1992) method was proposed by Sun et.al. (1995) who presented simulation results showing better artifact suppression and higher signal to noise ratio than those obtained with Paik and Katsaggelos.

The robustness of Hopfield net as an image-restoration tool has been demonstrated in many works. Bilgen and Hung (1994) considered a random shift-variant blur and Gaussian Noise in restoring one-dimensional signals. Perry and Guan (1995) also applied shift-variant distortions to 2D images. All these researchers used a common strategy that of mapping the error function to be minimized into the Hopfield network's energy function to exploit its energy-reduction ability. Figueirido and Leitão (1994) followed a different approach. They

proposed neural implementations of iterative restoration schemes that were shown to converge. The so called Gauss-Seidel algorithm and a modified Jacobi algorithm were examples of this approach, both implemented in Hopfield type networks of graded elements.

An architecture that lends itself more naturally to real-time operation is the multilayer feed forward neural network. Even though this network requires a relatively long training processes, once it is trained, it requires only a single, forward pass over the blurred image to produce a restored version. Thus it processes images faster than recursive architecture, smaller in size and less complex to implement.

Multilayer feed forward neural network had considered by Sivakumar and Desai (1993) for image restoration. In this algorithm they modified the transfer function used by neural network. A multilevel sigmoidal is defined and is used with a three layer perceptron. Restoration is achieved by exploiting the generalization capabilities of the multilayer perceptron network. They considered a shift-invariant blur with and without zero-mean white Gaussian noise on both binary and gray level images. Nathalie Plaziac (1999) showed that the neural filter outperforms the linear and median filters. The results show that the proposed neural network looks very promising for image interpolation, showing superior performance under noisy conditions. Moreover, the proposed nonlinear filter allows interpolating several pixels at a time, saving time and memory storage in the process. For all these reasons neural networks should be considered when intending some image interpolation. Davila and Hunt (2000) considered multilevel feed forward networks for image interpolation of both binary and gray level images. Neural networks have

been used for solving the super-resolution image reconstruction problem by Salari and Zhang, 2003 and Tsagaris et. al. 2004.

In this chapter we discuss the performance of a multilayer feed forward network in the super resolution binary images using discrete cosine transforms (DCT). The network considered here is a multilayer perceptron trained with Backpropagation algorithm. Although binary images are discussed here, this can be extended to gray scale images also and is considered in later chapters.

## 4.2 Discrete Cosine Transforms

A discrete cosine transform (DCT) expresses a sequence of finitely many data points in terms of a sum of cosine functions oscillating at different frequencies. DCTs are important to numerous applications in science and engineering, from lossy compression of audio and images (where small high-frequency components can be discarded), to spectral methods for the numerical solution of partial differential equations. The use of cosine rather than sine functions is critical in these applications: for compression, it turns out that cosine functions are much more efficient (as explained below, fewer are needed to approximate a typical signal), whereas for differential equations the cosines express a particular choice of boundary conditions.

In particular, a DCT is a Fourier-related transform similar to the discrete Fourier transform (DFT), but using only real numbers. DCTs are equivalent to DFTs of roughly twice the length, operating on real data with even symmetry (since the Fourier transform of a real and even function is real and even), where in some variants the input and/or output

data are shifted by half a sample. There are eight standard DCT variants, of which four are common.

The most common variant of discrete cosine transform is the type-II DCT, which is often referred to as "the DCT"; its inverse, the type-III DCT, is correspondingly often called "the inverse DCT" or "the IDCT". Two related transforms are the discrete sine transforms (DST), which is equivalent to a DFT of real and odd functions, and the modified discrete cosine transforms (MDCT), which is based on a DCT of overlapping data

Like any Fourier-related transform, discrete cosine transforms (DCTs) express a function or a signal in terms of a sum of sinusoids with different frequencies and amplitudes. Like the discrete Fourier transforms (DFT), a DCT operates on a function at a finite number of discrete data points. The obvious distinction between a DCT and a DFT is that the former uses only cosine functions, while the latter uses both cosines and sines (in the form of complex exponentials). However, this visible difference is merely a consequence of a deeper distinction: a DCT implies different boundary conditions than the DFT or other related transforms.

The Fourier-related transforms that operate on a function over a finite domain, such as the DFT or DCT or a Fourier series, can be thought of as implicitly defining an extension of that function outside the domain. That is, once a function  $f(x)$  is written as a sum of sinusoids, one can evaluate that sum at any  $x$ , even for  $x$  where the original  $f(x)$  was not specified. The DFT, like the Fourier series, implies a periodic extension of the original function. A DCT, like a cosine transform, implies an even extension of the original function.

Formally, the discrete cosine transform is a linear, invertible function  $F: \mathbf{R}^N \rightarrow \mathbf{R}^N$  (where  $\mathbf{R}$  denotes the set of real numbers), or equivalently an invertible  $N \times N$  square matrix. There are several variants of the DCT with slightly modified definitions. The  $N$  real numbers  $x_0, \dots, x_{N-1}$  are transformed into the  $N$  real numbers  $X_0, \dots, X_{N-1}$  according to one of the formulas:

### DCT-I

We write

$$X_k = \frac{1}{2} \left( x_0 + (-1)^k x_{N-1} \right) + \sum_{n=1}^{N-2} x_n \cos \left[ \frac{\pi}{N-1} nk \right] \quad (4.1)$$

$$k = 0, 1, \dots, N-1$$

Some authors further multiply the  $x_0$  and  $x_{N-1}$  terms by  $\sqrt{2}$ , and correspondingly multiply the  $X_0$  and  $X_{N-1}$  terms by  $1/\sqrt{2}$ . This makes the DCT-I matrix orthogonal, if one further multiplies by an overall scale factor of  $\sqrt{2/(N-1)}$ , but breaks the direct correspondence with a real-even DFT. The DCT-I is exactly equivalent (up to an overall scale factor of 2), to a DFT of  $2N - 2$  real numbers with even symmetry. However, the DCT-I is not defined for  $N$  less than 2. (All other DCT types are defined for any positive  $N$ ). Thus, the DCT-I corresponds to the boundary conditions:  $x_n$  is even around  $n=0$  and even around  $n=N-1$ ; similarly for  $X_k$ .

### DCT-II

We have

$$X_k = \sum_{n=0}^{N-1} x_n \cos \left[ \frac{\pi}{N} \left( n + \frac{1}{2} \right) k \right] \quad (4.2)$$

$$k = 0, \dots, N-1$$

The DCT-II is probably the most commonly used form, and is often simply referred to as "the DCT". This transform is exactly equivalent (up to an overall scale factor of 2) to a DFT of  $4N$  real inputs of even symmetry where the even-indexed elements are zero. That is, it is half of the DFT of the  $4N$  inputs  $y_n$ , where  $y_{2n} = 0$ ,  $y_{2n+1} = x_n$  for  $0 \leq n < N$ , and  $y_{4N-n} = y_n$  for  $0 < n < 2N$ . Some authors further multiply the  $X_0$  term by  $1/\sqrt{2}$ . This makes the DCT-II matrix orthogonal, if one further multiplies by an overall scale factor of  $\sqrt{2/N}$ , but breaks the direct correspondence with a real-even DFT of half-shifted input. The DCT-II implies the boundary conditions:  $x_n$  is even around  $n=-1/2$  and even around  $n=N-1/2$ ;  $X_k$  is even around  $k=0$  and odd around  $k=N$ .

### DCT-III

In this case,

$$X_k = \frac{1}{2} x_0 + \sum_{n=1}^{N-1} x_n \cos \left[ \frac{\pi}{N} n \left( k + \frac{1}{2} \right) \right] \quad (4.3)$$

$$k = 0, 1, \dots, N-1$$

Because eqn.(4.3) is the inverse of DCT-II this form is sometimes simply referred to as "the inverse DCT" ("IDCT"). Some authors further multiply the  $x_0$  term by  $\sqrt{2}$ , so that the DCT-II and DCT-III are transposes

of one another. This makes the DCT-III matrix orthogonal, if one further multiplies by an overall scale factor of  $\sqrt{2/N}$ , but breaks the direct correspondence with a real-even DFT of half-shifted output. The DCT-III implies the boundary conditions:  $x_n$  is even around  $n=0$  and odd around  $n=N$ ;  $X_k$  is even around  $k=-1/2$  and even around  $k=N-1/2$ .

### DCT-IV

We write,

$$X_k = \sum_{n=0}^{N-1} x_n \cos \left[ \frac{\pi}{N} \left( n + \frac{1}{2} \right) \left( k + \frac{1}{2} \right) \right] \quad (4.4)$$

$$k = 0, 1, \dots, N-1$$

The DCT-IV matrix becomes orthogonal if one further multiplies by an overall scale factor of  $\sqrt{2/N}$ . A variant of the DCT-IV, where data from different transforms are overlapped, is called the modified discrete cosine transform (MDCT). The DCT-IV implies the boundary conditions:  $x_n$  is even around  $n=-1/2$  and odd around  $n=N-1/2$ ; similarly for  $X_k$ .

### DCT V-VIII

DCT types I-IV are equivalent to real-even DFTs of even order (regardless of whether  $N$  is even or odd), since the corresponding DFT is of length  $2(N-1)$  (for DCT-I) or  $4N$  (for DCT-II/III) or  $8N$  (for DCT-VIII). In principle, there are actually four additional types of discrete cosine transform, corresponding to real-even DFTs of logically odd order,

which have factors of  $N \pm 1/2$  in the denominators of the cosine arguments.

Equivalently, DCTs of types I-IV imply boundaries that are even/odd around either a data point for both boundaries or halfway between two data points for both boundaries. DCTs of types V-VIII imply boundaries that even/odd around a data point for one boundary and halfway between two data points for the other boundary. However, these variants seem to be rarely used in practice. One reason is that FFT algorithms for odd-length DFTs are generally more complicated than FFT algorithms for even-length DFTs (e.g. the simplest radix-2 algorithms are only for even lengths), and this increased intricacy carries over to the DCTs as described below. (The trivial real-even array, a length-one DFT (odd length) of a single number  $a$ , corresponds to a DCT-V of length  $N=1$ .)

### **Inverse transforms**

Using the normalization conventions above, the inverse of DCT-I is DCT-I multiplied by  $2/(N-1)$ . The inverse of DCT-IV is DCT-IV multiplied by  $2/N$ . The inverse of DCT-II is DCT-III multiplied by  $2/N$  and vice versa. Like for the DFT, the normalization factor in front of these transform definitions is merely a convention and differs between treatments. For example, some authors multiply the transforms by  $\sqrt{2/N}$  so that the inverse does not require any additional multiplicative factor. Combined with appropriate factors of  $\sqrt{2}$ , this can be used to make the transform matrix orthogonal.



## Multidimensional DCTs

Multidimensional variants of the various DCT types follow from the one-dimensional definitions: they are simply a separable product (equivalently, a composition) of DCTs along each dimension. For example, a two-dimensional DCT-II of an image or a matrix is the one-dimensional DCT-II, from above, performed along the rows and then along the columns (or vice versa). That is, the 2D DCT-II is given by the formula (omitting normalization and other scale factors, as above):

$$X_{k_1, k_2} = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} x_{n_1, n_2} \cos \left[ \frac{\pi}{N_1} \left( n_1 + \frac{1}{2} \right) k_1 \right] \cos \left[ \frac{\pi}{N_2} \left( n_2 + \frac{1}{2} \right) k_2 \right] \quad (4.5)$$

Technically, computing a two- (or multi-) dimensional DCT by sequences of one-dimensional DCTs along each dimension is known as a row-column algorithm (after the two-dimensional case). As with multidimensional FFT algorithms, however, there exist other methods to compute the same thing while performing the computations in a different order (i.e. interleaving/combining the algorithms for the different dimensions). The inverse of a multi-dimensional DCT is just a separable product of the inverse(s) of the corresponding one-dimensional DCT(s), e.g. the one-dimensional inverses applied along one dimension at a time in a row-column algorithm.

## 4.3 Super Resolution

In chapter 1, the sampling and quantization of images are dealt with. Closely related to image sampling and quantization is the zooming and shrinking of an image. This is because zooming may be viewed as over sampling, while shrinking may be viewed as under sampling. The key difference between these two operations and sampling and quantizing an original continuous image is that zooming and shrinking are applied to digital image. Enlarging an image is also known as super-resolution because when enlarged, it is just extrapolating the band above the cut off frequency. Zooming requires two steps: the creation of new pixel locations, and the assignment of gray levels to those new locations. Image shrinking is done by row-column deletion (Gonzalez and Woods, 2002).

Super-resolution (SR) is techniques that in some way enhance the resolution of an imaging system. There are different views as to what is considered an SR-technique: some consider only techniques that break the diffraction-limit of systems, while others also consider techniques that merely break the limit of the digital imaging sensor as SR. There are both single-frame and multiple-frame variants of SR, where multiple-frame are the most useful. Algorithms can also be divided by their domain: frequency or space domain. By fusing together several low-resolution (LR) images one enhanced-resolution image is formed

In the most common SR algorithms, the information that was gained in the SR-image was embedded in the LR images in the form of aliasing. This requires that the capturing sensor in the system is weak enough so that aliasing is actually happening. A diffraction-

limited system contains no aliasing, nor does a system where the total system Modulation Transfer Function is filtering out high-frequency content.

There are also SR techniques that extrapolate the image in the frequency domain, by assuming that the object on the image is an analytic function, and that we can exactly know the function values in some interval. This method is severely limited by the noise that is ever-present in digital imaging systems, but it can work for radar, astronomy or microscopy.

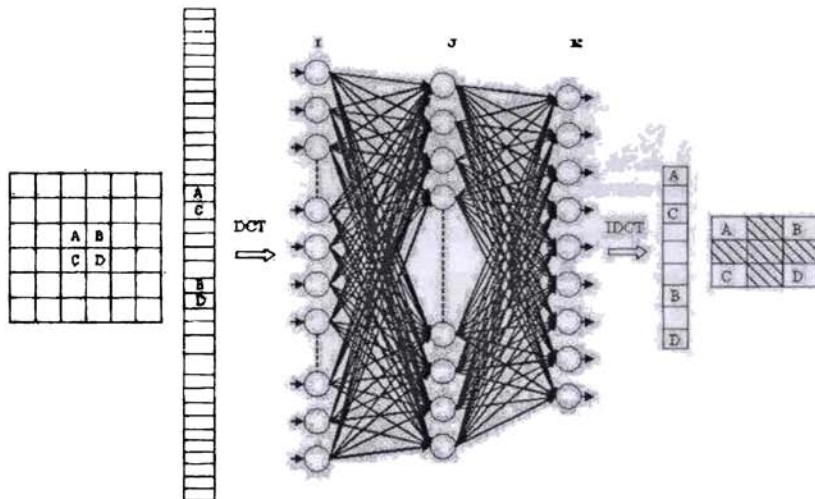
#### 4.4 Network Design and Training

The present problem is the design of a neural network trained with backpropagation algorithm for the reconstruction of the binary image above a cut off frequency  $\rho_c$ . The binary images of numbers 0 to 9 are chosen for the task. A 32x32 image of each number has been made and stored. The neural network used for training has the I-J-K format, where I is the number of neurons in the input layer, J, that in the hidden layer and K, the number of neurons in the output layer.

Design of a neural network consists of the determination of suitable I, J and K for a given problem so as to get a better performance for the network. Usually I and K are determined by the problem itself. The image is blurred using a low pass filter (lpf) with filter function given as:

$$lpf = \frac{1}{16} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (4.6)$$

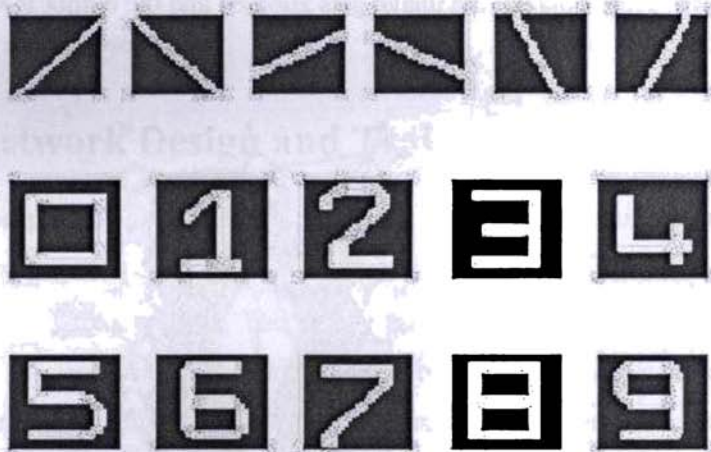
The image chosen for training is an image of size 32x32. This is blurred by the low pass filter given in Eq.4.6. The image is now down sampled to a size of 16x16. It is then split into overlapping sections of 6x6 (36) pixels each as shown in Fig.4.1. This is then lexicographically arranged to form a matrix of size 36x1. The discrete cosine transform (DCT) of this matrix is taken and is then fed to input of the neural network. Thus the number of input nodes for the neural network is now 36. As shown in Fig.4.1, the in between lines of the centre pixels A, B, C, D are interpolated. When these lines are interpolated, the image gets zoomed and the output is as shown.



**Fig.4.1 Neural network trained with backpropagation algorithm with the input and the interpolated output (shaded pixels are interpolated)**

The shaded portion in the output is the interpolated pixels; the output is thus 9. So the number of the output neurons is 9. But the DCT of the enlarged image is got. The inverse discrete cosine transform (IDCT) is

taken and is arranged to get a portion of the image of size 3x3. Since backpropagation is used the target must be given for training. The original 32x32 image is now split into overlapping sections of size 3x3, and lexicographically arranged to a matrix of size 9x1, which is given as the target for the network. The number of hidden layer neurons is found by actual training and testing. The peak signal to noise ratio (PSNR) is used as an indicator for the image comparison (Nathalie Plaziac, 1999).



*Fig.4.2 The training and testing set for the neural network. The first two rows of data are used for training and the last row for testing*

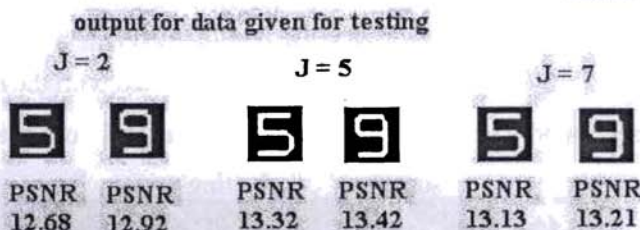
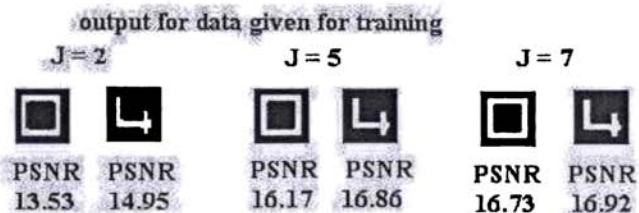
This ratio is described by:

$$PSNR = 10 \log \left( \frac{M^2}{\frac{1}{mn} \sum_{i=1}^m \sum_{j=1}^n e^2(i, j)} \right) \quad (4.7)$$

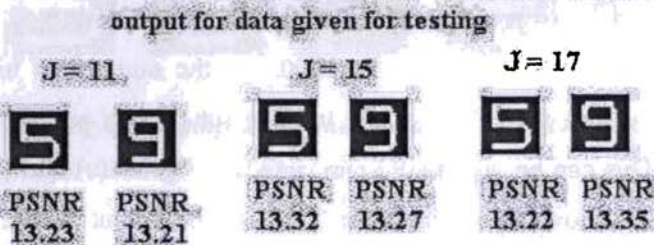
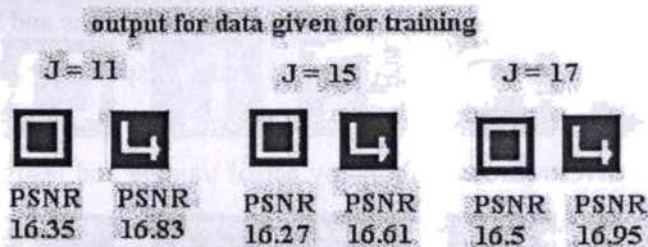
where  $m$  is the number of rows of the image,  $n$  is the number of columns of the image,  $M$  is the maximum value that a pixel can have, and  $e(i,j)$  is the difference between the two images ( the original image and the interpolated image) at pixel located at position  $(i,j)$ .

The data set for training and testing of the neural network is as shown in Fig.4.2. It is decided to use the numbers from 0-4 for training and the numbers 5-9 for testing. When trained and tested with the data, the neural network found it difficult to recognize number 7. In order to tackle this problem some slanting lines of  $30^\circ$ ,  $45^\circ$ ,  $60^\circ$ ,  $120^\circ$ ,  $135^\circ$  and  $150^\circ$  are also included for training. As indicated in Fig.4.2, the first two rows of data are used for training and the last row of data for testing. The number of input neurons and the number of output neurons are now fixed to 36 and 9 respectively. There is no thumb rule to determine the number of hidden layer neurons. The number of hidden layer neurons is found by actual training and testing. For that, the number of hidden layer neurons is varied from 2 to 17.

The number of iterations to be performed with each number of hidden layer neuron is fixed to 1000. As the number of hidden layer neurons varies, the PSNR of the training set of data and the testing set varies. This can be very well appreciated in Fig.4.3 (a) and (b). Fig. 4.3 (a) and (b) shows the variation in PSNR of the output for the number of hidden layer neurons  $J = 2$ ,  $J = 5$ ,  $J = 7$ ,  $J = 11$ ,  $J = 15$ ,  $J = 17$ . When a neural network is set up for a specific task, initially the weight factors from the input layer to the hidden layer ( $w_i$ ) and that from the hidden layer to the output layer ( $w_h$ ) are randomly selected. This weight factor initialization has an important role to play in the neural network



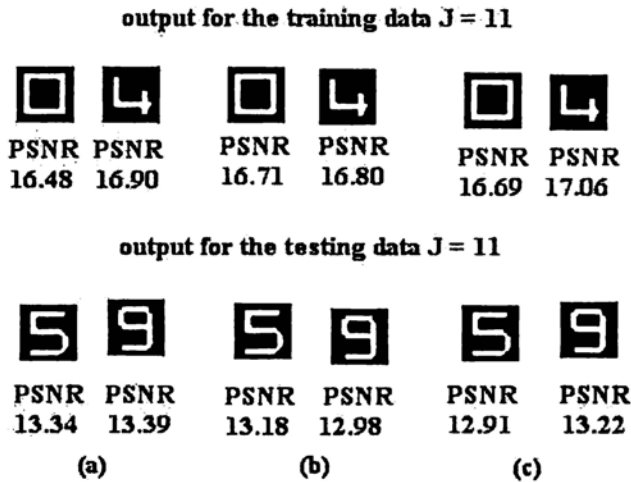
(a)



(b)

Fig. 4.3 (a) and (b) Variation in the PSNR of the training and testing set of data as the number of neurons in the hidden layer varies for  $J = 2$ ,  $J = 5$ ,  $J = 7$ ,  $J = 11$ ,  $J = 15$  and  $J = 17$

convergence (Hagan et. al., 2002, Haykin, 2003). Even if the number of input layer neurons, the hidden layer neurons and the output layer neurons are kept constant, the output of the neural network will vary with every fresh training of the network. This due to the change in the initialization of the weight factors with every fresh training of the network. Fig.4.4 clearly illustrates the variation of PSNR for the neural network with the



*Fig. 4.4 Variation of PSNR for different initialization of weight factor ( (a), (b), (c) ) for the number of hidden layer neurons, J = 11.*

number of neurons in the hidden layer,  $J = 11$  for different weight initialization. With each number of hidden layer neuron, iterations are done for 1000 times for 25 random weight initializations. For each weight initialization, the PSNR, as per Eq. 4.27, of each set of data (training and testing) is calculated. The average PSNR of these data are formulated as shown in Table 4.1 (a) and (b) and Table 4.2 (a) and (b). It can be seen



2	3	4	5	6	7	8	9	10
97.3	112.98	115.89	114.27	118.99	124.29	119.56	117.57	119.23
93.92	112	116.49	118.77	117.22	117.7	119.11	118.15	120.2
99.24	114.54	116.95	117.74	115.05	116.65	117.2	123.63	117.59
94.65	111.82	114.5	115.69	117.87	115.81	119.42	119.32	123.65
106	113.74	116.03	118.46	116.88	122.43	116.46	116.65	120.21
95.96	113.81	115.44	119.47	116.26	120.08	120.2	116.46	118.12
102.7	114.2	114.55	117.47	117.11	123.36	120.73	117.95	116.49
97.03	113.77	115.96	117.02	118.21	116.01	121.76	117.23	116.64
96.86	111.81	115.87	117.29	115.76	118.87	118.63	123.36	120.42
96.14	112.23	114.1	115.96	117.19	115.97	122.39	119.61	119.62
0.25	111.43	114.38	118.46	117.8	118.15	113.34	120.78	118.94
96.59	112.07	115.65	114.7	116.54	116.01	123.89	116.62	118.36
95.83	114.59	116.88	114.43	114.09	115.64	118.91	120.01	120.69
97.3	114.29	115.78	117.94	117.89	118.32	118.11	116.28	120.53
95.04	113.15	113.92	118.66	117.76	120.84	118.07	119.13	117.74
94.05	112.41	114.31	118.2	118.4	120.75	119.6	118.72	118.62
96.36	114.55	113.27	117.88	117.19	119.51	120.43	119.16	120.29
95.43	113.33	115.07	116.95	113.08	117.27	114.31	117.54	118.33
97.03	113.58	112.65	114.68	113.71	121.47	118.18	115.8	120.57
100.67	113.37	113.9	115.12	117.82	119.61	119.26	122.13	124.24
103.46	113.53	114.62	116.6	116.56	118.84	116.9	118.45	117.59
99.36	113.09	117.02	114.22	116.75	118.28	120.8	120.44	124.2
98.21	113.87	115.07	116.42	118.75	117.16	116.69	122.19	117.36
96.26	113.45	116.42	115.73	115.63	117.92	116.35	120.63	121.55
101.86	112.6	115.72	116.31	118.49	117.11	117.37	122.92	120.65
<b>97.9</b>	<b>113.21</b>	<b>115.22</b>	<b>116.74</b>	<b>116.84</b>	<b>118.72</b>	<b>118.71</b>	<b>119.23</b>	<b>119.67</b>

*Table 4.1(a) Training Data: Determination of number of hidden neurons; average PSNR in respect of the data used for training with the number of hidden neurons varying from  $J = 2$  to 10. Training is done with 25 new initializations of network weights and the network is trained for 1000 iterations for each initialization. Last row gives the average PSNR values for each number of hidden nodes*

11	12	13	14	15	16	17
122.89	118.23	119.29	119.66	119.11	120.65	116.4
120.28	119.59	121.35	119.36	126.48	119.89	117.83
121.44	119.98	118.57	118.96	123.29	120.58	117.7
118.87	118.03	121.9	119.85	118.72	121.51	125.56
117.07	121.5	125.71	118.91	124.04	125.46	118.88
118.78	116.65	119.23	119.11	119.39	118.72	124.95
123.98	119.09	116.14	124.84	121.71	122.47	119.53
117.83	115.97	121.07	116.46	118.81	116.5	120.68
118.58	121.5	119.66	121.68	120.29	117.5	118.93
118.78	119.67	122.17	117.33	121.87	115.46	115.78
122.86	123.75	118.84	118.08	120.78	118.4	119.73
116.49	118.01	123.33	118.8	121.81	116.96	120.31
117.65	118.51	117.03	120.33	118.88	119.64	121.52
122.67	121.77	120.73	117.69	118.61	118.82	120.14
120.32	120.81	119.58	118.68	117.71	122.2	120.47
125.19	118.23	119.29	120.07	117.89	122.17	122.73
125.41	119.48	116.06	121.66	118.2	119.31	118.06
124.25	118.52	121.43	122.17	123.55	119.22	122.77
121	118.81	120.15	120.43	117.63	118.25	118.99
120.55	121.66	122.29	119.83	119.32	120.57	119.75
121.9	124.9	117.64	119.7	120.69	123.21	120.66
118.7	121.25	119.67	123.33	120.72	117.59	119.95
120.58	122.44	120.86	114.97	120.67	125.57	122.14
122.69	121.54	120.01	119.72	118.13	122.45	119.47
119.01	125.99	120.33	120.09	120.77	119.85	120.2
<b>120.71</b>	<b>120.24</b>	<b>120.09</b>	<b>119.67</b>	<b>120.36</b>	<b>120.12</b>	<b>120.13</b>

*Table 4.1(b) Training Data: Determination of number of hidden neurons; average PSNR in respect of the data used for training with the number of hidden neurons varying from  $J = 10$  to 17. Training is done with 25 new initializations of network weights and the network is trained for 1000 iterations for each initialization. Last row gives the average PSNR values for each number of hidden nodes*

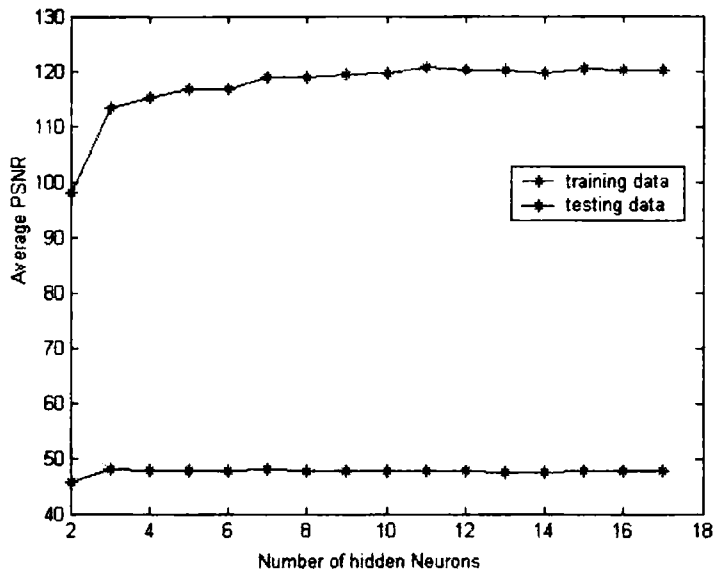
2	3	4	5	6	7	8	9	10
46.18	47.84	47.72	47.8	47.78	49.7	47.16	47.32	47.15
45.62	47.83	47.87	47.28	47.25	47.77	47.84	48.27	47.99
46.35	48.08	48.07	48.34	47.74	46.78	48.09	49.27	48.21
45.62	47.73	47.78	48.04	47.73	47.93	48.11	46.38	47.56
45.42	47.92	48.01	48.13	47.94	47.96	47.23	47.31	47.65
46.15	47.9	47.95	48.5	48	48.02	48.56	47.84	47.95
44.1	48.08	48.25	47.83	47.8	49.67	49.09	47.79	46.77
46.18	47.88	47.86	47.79	48.15	47.66	49.26	47.51	47.29
46.02	47.9	47.95	47.99	46.84	47.2	48.2	47.1	49.15
46.15	47.95	47.92	48.1	47.36	46.92	46.83	47.06	47.7
43.33	48	48	47.6	47.97	48.37	48.24	47.1	47.28
45.87	48.07	47.87	47.78	47.69	48.08	46.77	47.5	48.25
45.96	47.91	48.06	47.7	47.65	47.68	47.71	47.44	46.6
46.21	47.93	47.03	47.98	48.06	48.77	47.63	47.75	47.82
45.64	47.53	47.96	47.94	48.36	48.19	47.51	47.8	47.8
45.59	47.84	47.76	46.94	47.99	49.87	48.13	48.09	47.39
46.04	47.83	47.77	47.47	47.7	48.41	48.28	48.5	48.71
45.75	48.05	47.86	47.31	47.43	48.06	47.74	47.27	47.72
46.13	47.87	47.51	47.94	47.1	47.79	48.22	47.36	48.76
42.79	47.81	47.88	47.9	47.79	48.51	46.76	47.17	47.63
46.35	47.8	47.58	48.03	47.92	47.93	47.16	47.77	47.84
46.4	47.99	47.84	48.04	47.95	48.25	48.01	48.34	48.02
46.26	48.09	48.09	47.98	48.54	47.97	47.94	49.6	46.59
46.22	47.97	48.04	47.86	47.27	47.8	46.31	47.67	47.41
46.22	47.78	48.27	47.96	47.71	47.73	47.99	48.16	47.33
45.7	47.9	47.88	47.85	47.75	48.12	47.79	47.73	47.7

**Table 4.2 (a) Testing Data: Determination of number of hidden neurons; average PSNR for the data used for testing for the number of hidden neurons varying from  $J = 2$  to 10. For each of the network weights obtained with the training data is correspondingly tested. Last row gives the average PSNR.**

11	12	13	14	15	16	17
47.97	47.61	48.17	47.61	48.66	46.76	47.22
47.71	47.49	47.9	47.93	49.49	47.36	47.66
48.05	47.75	48.08	47.12	47.57	47.18	47.59
47.11	47.93	48.22	48.19	47.03	48.91	47.35
46.92	47.13	47.26	47.54	47.8	47.17	48.74
47.96	47.64	46.32	47.54	47.3	47.8	48.66
48.41	48.32	47.31	47.61	47.06	47.74	48.31
47.45	47.54	48.25	47.97	48.13	47.65	47.47
47.73	46.59	47.94	46.59	47.85	47.45	47.75
48.26	47.26	47.77	48.08	47.45	48.55	47.82
49.11	49.29	47.31	47.55	48.74	47.05	48.09
47.35	48.33	47.27	47.21	47.46	46.74	48.48
47.21	46.72	47.38	47.06	47.37	48.39	49.02
47.54	47.97	47.8	47.5	47.69	47.3	48.7
46.93	47.41	47.27	47.39	46.93	47.36	47.45
48.59	48.73	48.12	47.29	47.64	49.47	48.19
49.5	47.93	46.91	49.03	48.19	47.48	48.01
47.4	47.75	47.2	47.45	48.27	47.3	47.75
47.21	48.66	47.47	48.14	47.98	47.95	46.67
47.93	47.13	47.31	47.11	47.63	47.47	47.1
47.86	48.2	47.73	47.51	47.64	48.03	46.85
46.92	48.56	47.37	47.72	48.43	47.29	46.98
46.62	47.9	47.71	47.73	47.16	48.84	46.26
50.04	47.63	47.32	47.62	47.85	48.5	47.24
47.91	49.15	48.24	47.24	47.06	47.46	47.63
47.83	47.86	47.58	47.59	47.78	47.73	47.72

*Table 4.2 (b) Testing Data: Determination of number of hidden neurons; average PSNR for the data used for testing for the number of hidden neurons varying from  $J = 11$  to 17. For each of the network weights obtained with the training data is correspondingly tested. Last row gives the average PSNR.*

from table 4.1 (a) and (b), as the number of hidden layer neurons increases the performance of the network becomes better and better. With a large number of hidden layer neurons, the learning becomes less tedious. But the response of the neural network to untrained data will be poor. In short, the generalization capability of the network will be first increasing and then it shows a decreasing trend with the increase in the number of hidden layer neurons. This fact is evident in the table 4.2. Here the PSNR is determined for the test data, for various weight factors obtained for the various training of the neural network. The data in table 4.1(a) and (b) and table 4.2(a) and (b) is shown in Fig. 4.5. Also as the



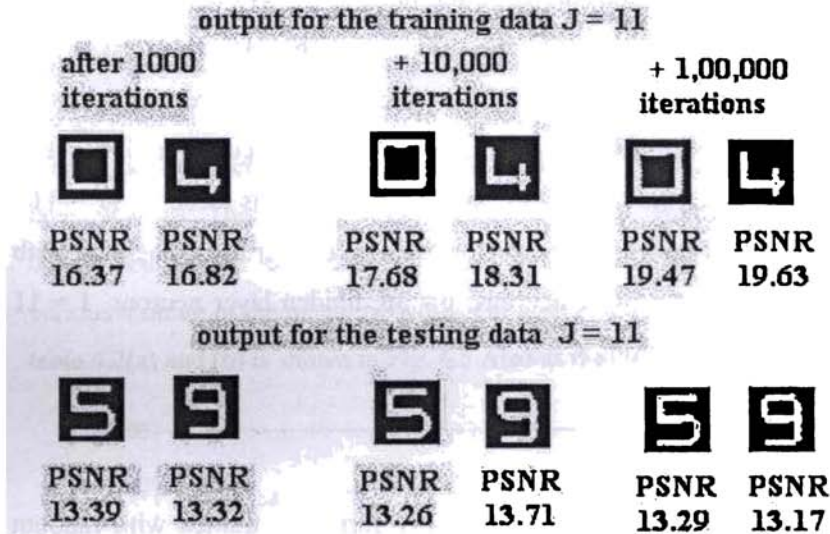
**Fig.4.5 Plot showing the variation in average PSNR for the training and testing data with the number of hidden neurons**

number of hidden layer neurons increases, the hardware requirement of the computer also increases. More the number of neurons, the more will be the memory requirement. More the number of neurons, the more will be computation complexity, the more will be the time taken for processing. So it is a trade off between the hardware, computation complexity, performance, time etc. With all these considerations, it is decided to select the number of hidden layer neurons to be 11 ( $J = 11$ ). Now the neural network is ready for training and further processing with the number of input layer neurons,  $I = 36$ , hidden layer neurons,  $J = 11$  and output layer neurons  $K = 9$ .

## 4.5 The Output

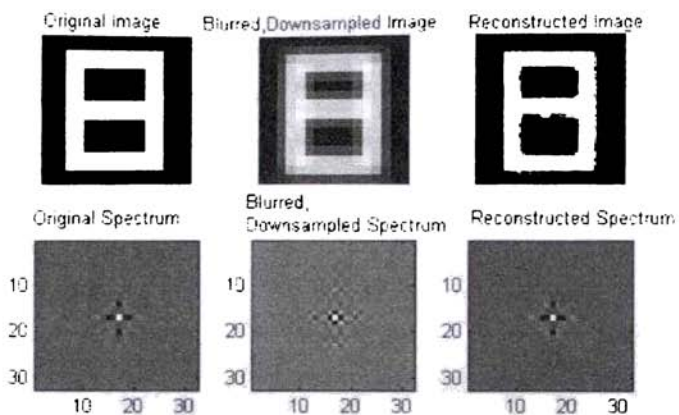
The neural network in the I-J-K format is trained with random initialization. The activation function is so chosen that for the hidden layer neurons sigmoid function is used and for the output layer neuron purelinear activation function is employed. The neural network is trained step by step. Each step consists of a specific number of iterations. At the end of each step the PSNR is tested. A neural network cannot be trained for a large time or for a large number of iterations as shown in Fig.4.6. Initially, as the number of iterations increases, the PSNR also increases. After specific number iterations, which have to be found by constant testing, the neural network gets saturated. The PSNR for the training set of data tend to increase with increasing number of iterations, whereas the generalizing capability of the network tends to decrease. When saturation is attained, further training will lead the neural network to confusion or it is said to be over trained. Hence, it is necessary to stop the training of the

neural network properly. After getting a proper termination for the neural network training, the network can be saved and can be tested.

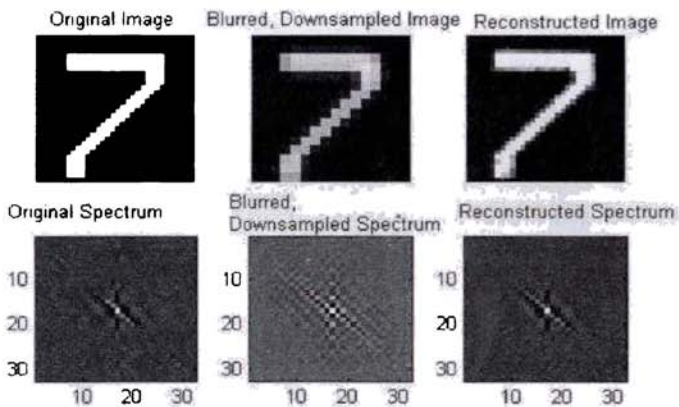


**Fig.4.6 Variation of the PSNR with the number of iterations**

The simulations are done with the binary image of numbers 0-9. The results of these simulations are compared with interpolate techniques like nearest neighbour, bilinear, bicubic and the spline interpolation methods that are available. The peak signal to noise ratio (PSNR) is used as an indicator for the image comparison as given in Eq.4.7. The training and testing set consist of numbers from 0-9 and lines of different inclinations as shown in Fig.4.2. For the multilayer perceptron (MLP) the simulation results are as shown in Fig.4.7. The simulation result shown in Fig.4.7, is for an unseen, new image for the MLP. The MLP is trained with numbers 0-4 and some slanting lines. The slanting lines are used because it became difficult for the MLP to recognize number 7 as it



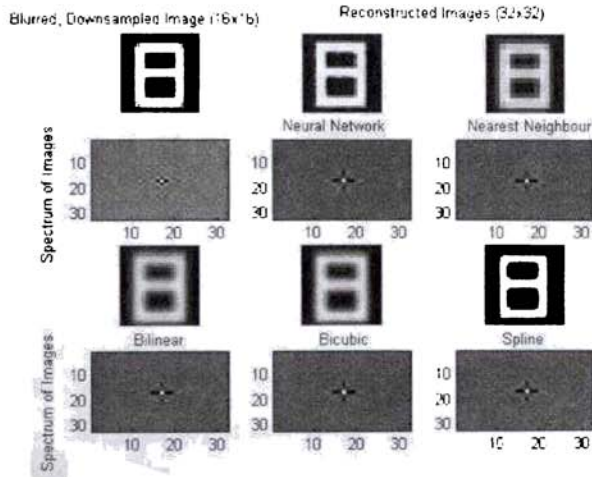
**Fig.4.7 Image reconstruction done for number 8 with their spectra.**



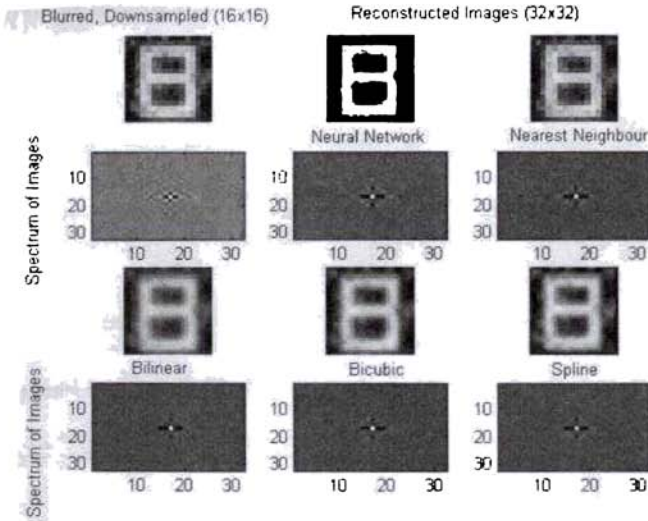
**Fig.4.8 Image reconstruction done for number 7 with their spectra.**

contains inclined lines. Number 8 is not in the training set and the MLP is able to reconstruct the image. This shows the generalization capability of the MLP. This can be very well appreciated in Fig.4.8 also. Here the reconstruction of number 7 is given. The performance of the MLP is





(a)



(b)

**Fig.4.9(a) Image reconstruction done for number 8 (b) with added noise using various methods like Neural Network, Nearest Neighbour, Bilinear, Bicubic and Spline interpolation methods with their spectra.**

compared with the existing techniques like nearest neighbour, bilinear, bicubic and spline interpolation methods. As shown in Fig.4.9 (a) and (b), the performance of the neural network is much better.

The PSNR related to each technique is calculated and compared. Details of comparison are shown as in table 4.3. Also the average PSNR is calculated and it is clear as can be seen that the performance of the neural network is comparable

Numbers	Neural Network	Nearest Neighbour	Bilinear	Bicubic	Spline
1	14.713	12.256	12.55	12.705	14.824
2	15.591	10.978	11.396	11.585	13.87
3	16.808	11.814	11.645	11.941	13.271
4	18.609	15.651	14.191	14.6	15.189
5	13.433	9.9562	10.63	10.793	12.981
6	13.667	10.612	11.136	11.321	13.343
7	16.026	16.022	14.359	15.009	15.216
8	14.073	10.609	11.055	11.342	12.573
9	13.67	10.483	10.755	10.899	13.335
0	17.915	16.63	13.012	13.556	13.023
Lines 45 <sup>0</sup>	22.816	13.753	13.963	14.08	16.049
135 <sup>0</sup>	22.8	15.364	15.087	15.883	15.842
30 <sup>0</sup>	21.529	15.622	15.262	15.792	16.033
150 <sup>0</sup>	23.947	15.612	15.251	15.766	16.124
60 <sup>0</sup>	21.33	15.789	15.687	16.263	16.398
120 <sup>0</sup>	20.616	14.622	14.502	14.728	16.231
Average	17.971	13.486	13.155	13.517	14.644

**Table 4.3** The PSNR of the numbers and lines compared with the various methods

## Summary

A neural network in the I-J-K format is set up with the number of input layer neuron I = 36, hidden layer neuron J = 11 and output layer

neuron  $K = 9$ . The neural was able to reconstruct the frequencies above the cut off frequency. The performance of the network is evaluated. The efficiency of the network is compared with the existing interpolating techniques and was found better.

## References:

- [1] Abbiss J B, B J Brames, J S Bailey and M A Fiddy, 1988. Super-resolution and Neural Computing, High Speed Computing, D P Casasent, ed., Proc. SPIE 880, 100-106
- [2] Abbiss J B, B J Brames, and M A Fiddy, 1991. Super-resolution algorithms for a modified Hopfield neural network, IEEE Trans. Signal Processing, 39, 1516-1523
- [3] Bilgen M and H S Hung, 1994. Neural Network for restoration of signals blurred by a random shift-invariant impulse response function, Opt. Eng. 33, 2723-2727
- [4] Byrne C L, R M Fitzgerald, M A Fiddy, T J Hall and A M Darling, 1983. Image Restoration and resolution enhancement, J. Opt. Soc. Am. 73, .1481-1487
- [5] Condocia Frank M and Jose C Principe, 1999. Super-Resolution of Images Based on Local Correlations, IEEE Trans. on neural networks, vol.10 No. 2, . 372-380
- [6] Chen T, Chen H, and R Liu, 1995. Approximation capability in  $C(R^n)$  by multilayer feedforward networks and related problems, IEEE Trans. Neural Networks, 6, 25-30

- [7] Darling A M, T J Hall and M A Fiddy, 1983. Stable, noniterative object reconstruction from incomplete data using a priori knowledge, *J. Opt. Soc. Am.* 73, .1466-1469
- [8] Davila C A and B R Hunt, 2000. Super-Resolution of Binary Images with a Nonlinear Interpolative Neural Network, *Applied Optics*, vol.39, No.14, . 2291-2299.
- [9] Davila C A and B R Hunt, 2000. Training of a Neural Network for Image Super-Resolution Based on a Nonlinear Interpolative Vector Quantiser , *Applied Optics*, vol.39, No.20, . 3473-3485.
- [10] Fig.ueiredo M and J Leitão, 1994. Sequential and Parallel Image Restoration: Neural network implementations, *IEEE Trans. Image Process.* 3, 789-801
- [11] Frieden B R and J J Burke, 1972. Restoring with maximum entropy II: superresolution of photographs of diffraction blurred impulses, *J. Opt. Soc. Am.* 62 . 1202-1210
- [12] Gerchberg R W, 1974. Super-Resolution through Error Energy Reduction, *Opt. Acta*, 21, 709-720
- [13] Gonzalez R C and Richard E Woods, 2002. *Digital Image Processing, Second Edition*, Pearson Edn.
- [14] Hagan, Martin T Howard B Demuth and Mark Beale, 2002 *Neural Network Design*, first ed., Boston, Thomson Learning
- [15] Haykin, S., 2003. *Neural Networks: A Comprehensive Foundation. Second Edition*, Pearson Education
- [16] Hopfield, J.J., 1984. Neurons with graded response have collective computational properties like those of two-state neurons. *Proc. Natl. Acad. Sci.* 81, .3088–3092.

- [17] Hopfield, J.J., 1982. Neural Networks and Physical Systems with emergent collective computational abilities,. Proc. Natl. Acad. Sci. 79, 2554-2558.
- [18] Hunt B R, 1974. Imagery Super-Resolution: emerging prospects, Applications of Digital Image Processing XIV, A G Tescherm, Proc. SPIE 1567 600-608
- [19] Kris Jensen and Dimitris Anastassiou, 1995. Subpixel Edge Localization and the Interpolation of Still Images, IEEE Trans. on Image Processing, vol.4, . 285-295.
- [20] Lucy L B , 1974. An iterative technique for the Rectification of observed distributions, Aston. J. 79 . 745-754
- [21] Nathalie Plaziac, 1999. Image Interpolation Using Neural Networks, IEEE Trans. Image Processing, vol.8, No.11, . 1647-1651.
- [22] Paik J K and A K Katsagelos, 1992. Image Restoration using a Modified Hopfield network, IEEE Trans. Image Proccss. 1, 49-63
- [23] Papoulis A, 1975. A New Algorithm in Spectral Analysis and Band Limited Extrapolation, IEEE Trans. circuits syst. 22, . 735-742
- [24] Perry S W and L Guan, 1995. Neural network restoration of images suffering space-variant distortion, Electron. Lett. 31 1358-1359.
- [25] Pham T Q, L J Van Vliet and K Schutte, 2006. Resolution Enhancement of Low Quality Videos using a High Resolution Frame, Visual Communications and Image Processing, Proc. of SPIE vol. 6077, 2006, .123-141. chapter 6

- [26] Richardson W H, 1972. Bayesian – based iterative method of image restoration, *J. Opt. Soc. Am.* 62, 55-59
- [27] Salari E, Zhang S , 2003. Integrated recurrent neural network for image resolution enhancement from multiple image frames. *IEEE Proc* 150:299–305
- [28] Sheppard D G, A Bilgin, M S Nadar, B R Hunt and M W Marcellin, 1998. A Vector Quantizer for Image Restoration, *IEEE Trans. Image Process.* 7, .119-124
- [29] Sivakumar K and U B Desai, 1993. Image Restoration Using a Multilayer Perceptron with Multilevel Sigmoidal Function, *IEEE Trans. Signal Processing* 14, No.5, . 2018-2022.
- [30] Sun Y, J G Li and S Y Yu, 1995. Improvement on performance of modified Hopfield neural network for image restoration, *IEEE Trans. Image Process.* 4, 688-692.
- [31] Tsagaris V, Panagiotopoulou A, Anastassopoulos V, 2004. Interpolation in multispectral data using neural networks. *Proc SPIE* 5573:460–470
- [32] Walsh D O and P A Nielsen-Delaney, (1994). Direct Method for Super-resolution, *J. Opt. Soc. Am. A* 11, .572-579
- [33] Zhou Y T, R Chellappa, A Vaid and B K Jenkins, 1988. Image Restoration using Neural Networks, *IEEE Trans. Acoust., Speech, Signal Processing* vol. ASSP-36, 1141-1151
- [34] Zhang W, K Itoh, J Tanida and Y Ichioka, 1991. Hopfield model with multistate neurons and its optoelectronic implementation, *Appl. Opt.* 30, 195-200



## **CHAPTER 5**

# **RESTORATION OF GRAY LEVEL IMAGES WITH DISCRETE COSINE TRANSFORMS**

### **5.1 Introduction**

In the previous chapter with binary image restoration, the variation of the neural network output with the number of hidden layer neurons, the input weight initialization and number of iterations is discussed. These are not only the factors that have to be considered for design of an efficient neural network. The performance of the neural network is affected by the variation of the activation function, the selection of the input data given for training, the selection of proper training algorithms etc. Here an attempt is done to illustrate the performance variation of the neural network with these parameters. Also the capability of the neural network in image restoration with discrete cosine transform is mentioned. A comparison of the performance of neural network with other image interpolation techniques is also done.

### **5.2 Shrinking and Zooming of image**

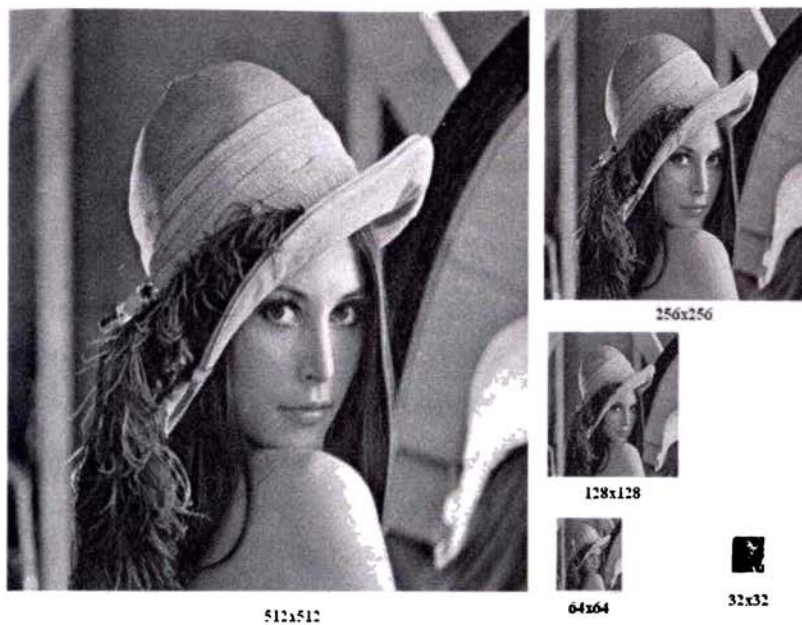
Sampling is the principal factor determining the spatial resolution of an image. Basically, spatial resolution is the smallest discernible detail in an image. Gray level resolution refers to the smallest discernible change in gray level. There are considerable discretion regarding the



number of samples used to generate a digital image, but this is not true for the number of gray levels. Due to hardware considerations, the number of gray levels is usually an integer power of 2. The most common number is 8 bits, with 16 bits being used in some applications where enhancement of specific gray-level ranges is necessary. When an actual measure of physical resolution relating pixels and the level of detail they resolve in the original scene are not necessary, it is not uncommon to refer to an  $L$  – level digital image of size  $M \times N$  as having a spatial resolution of  $M \times N$  pixels and a gray-level resolution of  $L$  levels. (Gonzalez and Woods, 2002)

Figure 5.1 shows an image of size 512 x 512 pixels whose gray levels are represented by 8 bits. The other images shown are the results of subsampling the 512 x 512 image. The subsampling was accomplished by deleting appropriate number of rows and columns from the original image. This is also referred as shrinking of the image. These images show the dimensional proportions between various sampling densities, but their size differences make it difficult to see the effects resulting from a reduction in the number of samples. The simplest way is to compare their effects is to bring all the subsampled images upto the size 512 x 512 as shown in Fig.5.2

When considering subsampling or shrinking of an image, equal importance must be given to enlargement or zooming of images. When subsampled, the details of the image were lost. Enlargement or zooming focuses on reconstructing this lost details in the image. When this lost information is added, a 32 x 32 image can be zoomed to 64 x 64 image. Zooming requires two steps: the creation of new pixel locations, and the assignment of gray levels to those new locations.



**Fig. 5.1** A 512 x 512, 8 bit image subsampled to a size of 32 x 32 pixels. The number of allowable gray levels was kept at 256.



**Fig.5.2** Resampled images.

There are many methods to perform this kind of assignment. These are commonly referred to as the image interpolation techniques. The most common interpolation techniques are discussed in chapter 4. The simplest of these technique is the nearest neighbour interpolation method. In this

method, in order to perform gray-level assignment for any point in the overlay, the closest pixel in the original image is considered and assigns its gray level to the new pixel in the grid. When this is done with all points in the overlay grid, a zoomed image will be obtained (Gonzalez and Woods, 2002).

Although nearest neighbour interpolation is fast, it has the undesirable feature that it produces a check board effect that is particularly objectionable at high factors of magnification. In computer vision and image processing, bilinear interpolation, another method of interpolation, is one of the basic resampling techniques. It is a texture mapping technique that produces a reasonably realistic image, also known as bilinear filtering or bilinear texture mapping. An algorithm is used to map a screen pixel location to a corresponding point on the texture map. A weighted average of the attributes (colour, alpha, etc.) of the four surrounding texels is computed and applied to the screen pixel. This process is repeated for each pixel forming the object being textured (Gonzalez and Woods, 2002).

Commonly, magnification is accomplished through convolution of the image samples with a single kernel—typically the bilinear, bicubic (Netravali and Haskell, 1995) or cubic B-spline kernel (Unser M et.al.,1991) . The mitigation of aliasing by this type of linear filtering is very limited. Magnification techniques based on a priori assumed knowledge are the subject of current research. Directional methods (Bayrakeri and Mersereau, 1995 and Jensen and Anastassiou, 1995) examine an image's local edge content and interpolate in the low frequency direction (along the edge) rather than in the high-frequency

direction (across the edge). Multiple kernel methods typically select between a few ad hoc interpolation kernels (Darwish and Bedair, 1996). Orthogonal transform methods focus on the use of the discrete cosine transform (DCT) (Martucci, 1995 and Shinbori and Takagi, 1994) and the wavelet transform (Chang et. al., 1995). Variational methods formulate the interpolation problem as the constrained minimization of a function (Karayiannis and Venetsanopoulos, 1991 and Schultz and Stevenson, 1994). It should be noted that these techniques make explicit assumptions regarding the character of the analog image. Most of the super resolution algorithms are discussed in chapter 4.

Another approach of resolution improvement in remotely sensed images is that of a fully inter connected NN model (Valdes and Inamura, 2000). The specific single-hidden-layer neural network, being trained by the backpropagation algorithm, is required to enhance the resolution of diffraction-limited, binary images. Moreover, a high-resolution, multi-neural network, based on the local variance is proposed Sekiwa and Taguchi, 2001. This specific network is composed of two neural networks, namely the NN for low local variance and the NN for high local variance. The weighted sum of the two NN outputs represents the enlarged image. A novel image interpolation scheme, using an artificial neural network, is described by Pan and Zhang, 2003. A single frame interpolation algorithm is joined together with an adaptive, linear, single-layer neural network that models the residual errors between the interpolated image and the respective original one. A novel image interpolation algorithm by means of a feedforward neural network, based upon classification, is thoroughly considered by Hu et.al. 2004. A HVS-

oriented, adaptive interpolation scheme for natural images by means of neural networks is proposed Pu et. al. 2003. A Hopfield-network-based algorithm, serving for the resolution enhancement of discrete targets taking up more space than the sample spacing of an image, is dealt with by Collins and Jong, 2004. Multilayer neural networks have also been used to perform document resizing by Ahmed et. al. 2001. Moving from a high-resolution image to a lower-resolution one, a group of pixels is replaced by one pixel. In theory, this replacement is determined by the scanner characteristics (Craubner, 2002 and Shen and Xin, 2004). A novel method of improving the spatial resolution of scanned images, by means of neural networks, is presented by Antigoni and Vassilis, 2008.

Typically bilinear interpolation can be used where perfect image transformation, matching and imaging is impossible so that it can calculate and assign appropriate image values to pixels. Unlike other interpolation techniques such as nearest neighbour interpolation and bicubic interpolation, bilinear interpolation uses the 4 nearest pixel values which are located in diagonal direction from that specific pixel in order to find the appropriate color intensity value of a desired pixel.

There are other sophisticated mathematical techniques for image interpolation like bicubic interpolation, spline interpolation etc. In image processing, bicubic interpolation is often chosen over bilinear interpolation or nearest neighbor in image resampling, when speed is not an issue. Images resampled with bicubic interpolation are smoother and have fewer interpolation artifacts.

Spline interpolation is preferred over polynomial interpolation because the interpolation error can be made small even when using low

degree polynomials for the spline. Using polynomial interpolation, the polynomial of degree  $n$  which interpolates the data set is uniquely defined by the data points. The spline of degree  $n$  which interpolates the same data set is not uniquely defined, and we have to fill in  $n-1$  additional degrees of freedom to construct a unique spline interpolant.

With the rapid increase in available computing power, coupled with great strides in image feature analysis, model-based, often highly nonlinear, interpolative techniques have become a viable alternative to classic linear methods and have received increasing attention recently. Several examples of model-based approaches to spatial image interpolation can be found in Jensen and Anastassiou (1995), Jensen and Anastassiou (1990), Martinez and Lim (1989), Wang and Mitra (1991) and Condociu and Principe (1989). Each of these papers utilizes the concept of an edge in a different fashion to enhance interpolation results.

### **5.3 Neural Networks and Image Interpolation**

The multilayer perceptron is one of the most common feedforward architectures. It consists of at least three layers: an input layer, which simply distributes the inputs to the next layer; a hidden layer; and an output layer, which collects the hidden layer outputs and computes the final output. The many powerful properties of the multilayer perceptron make it an attractive candidate for the image restoration and super resolution problems (Davila and Hunt, 2000, Nathalie Plaziac, 1999). It is well known that the multilayer neural networks can be used to approximate almost any function, if there are enough neurons in the hidden layers. Due to this property of neural networks, they are sometimes

regarded as a universal approximator in the sense that it can approximate an input-output mapping to any degree of approximation, given a sufficient number of hidden units (Hagan, et. al., 2002). Since it is a universal approximator, it can even extrapolate a band-limited signal over its pass band. So the neural networks can be used to interpolate a digital image.

A lot of research has been done in the area of image interpolation. Most of the researchers rely on sophisticated mathematical equations for the purpose. Very few researchers have directed their vision in the direction of neural networks for image interpolation. Of these, most of them had directed to Hopfield network or other competitive networks, where unsupervised learning is made use of. Few of the researchers are oriented towards multilayer perceptron or supervised learning algorithms and exploited the approximation capability of the neural networks. These works are capable of on line processing since most of the computational complexities are met in the training phase. In the implementation phase only the desired format input is fed to the neural network and the network approximate the function accordingly.

Here, in this work an attempt is done to interpolate digital image with multilayer perceptrons using discrete cosine transforms. In 1975, Athanasios Papoulis proposed a method which can be used to extrapolate a band-limited function. The algorithm is a simple iteration involving only the fast Fourier transform. In the proposed algorithm, the effect of noise and the error due to aliasing are determined and it is shown that they can be controlled by proper termination of the iteration.

## 5.4 Neural Network Design and Training

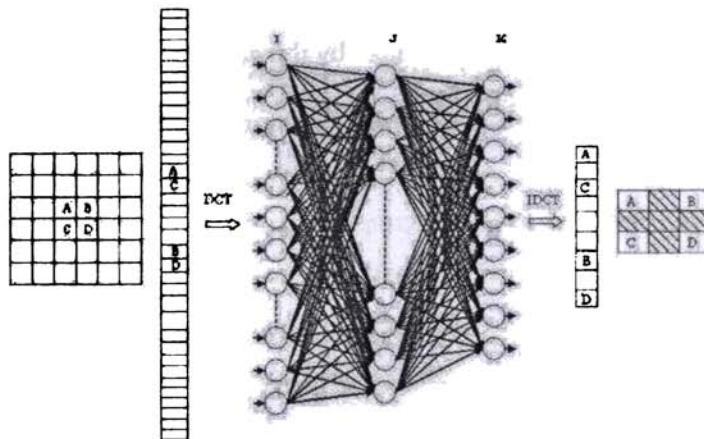
The problem here is to design a neural network trained with backpropagation algorithm for the reconstruction of the gray level image above a cut off frequency  $\rho_c$ . The neural network used for training has the I-J-K format, where I is the number of neurons in the input layer, J, that in the hidden layer and K, the number of neurons in the output layer. Design of a neural network consists of the determination of suitable I, J and K for a given problem so as to get a better performance for the network. Usually I and K are determined by the problem itself. The image is blurred using a low pass filter (lpf) with filter function given as:

$$lpf = \frac{1}{16} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (5.1)$$

The image chosen for training is an image of size 256x256. This is blurred by the low pass filter given in Eq.5.1. The image is now down sampled to a size of 128x128. It is then split into overlapping sections of 6x6 (36) pixels each as shown in Fig.5.3. This is then lexicographically arranged to form a matrix of size 36x1. The discrete cosine transform (DCT) of this matrix is taken and is then fed to input of the neural. Thus the number of input nodes for the neural network is now 36. As shown in Fig.5.3, the in between lines of the centre pixels A,B, C, D are interpolated. When these lines are interpolated, the image gets zoomed and the output is as shown. The shaded portion in the output is the interpolated pixels; the output is thus 9. So the number of the output neurons is 9. But the DCT of the enlarged image is got. The inverse discrete cosine transform (IDCT) is



taken and is arranged to get a portion of the image of size  $3 \times 3$ . Since backpropagation is used the target must be given for training. The original  $256 \times 256$  image is now split into overlapping sections of size  $3 \times 3$ , and lexicographically arranged to a matrix of size  $9 \times 1$ , which is given as the target for the network. The number of hidden layer neurons is found by actual training and testing. The performance of the neural network is found to be better when the number of hidden layer neuron is 4.



**Fig.5.3** Neural network trained with backpropagation algorithm with the input and the interpolated output (shaded pixels are interpolated)

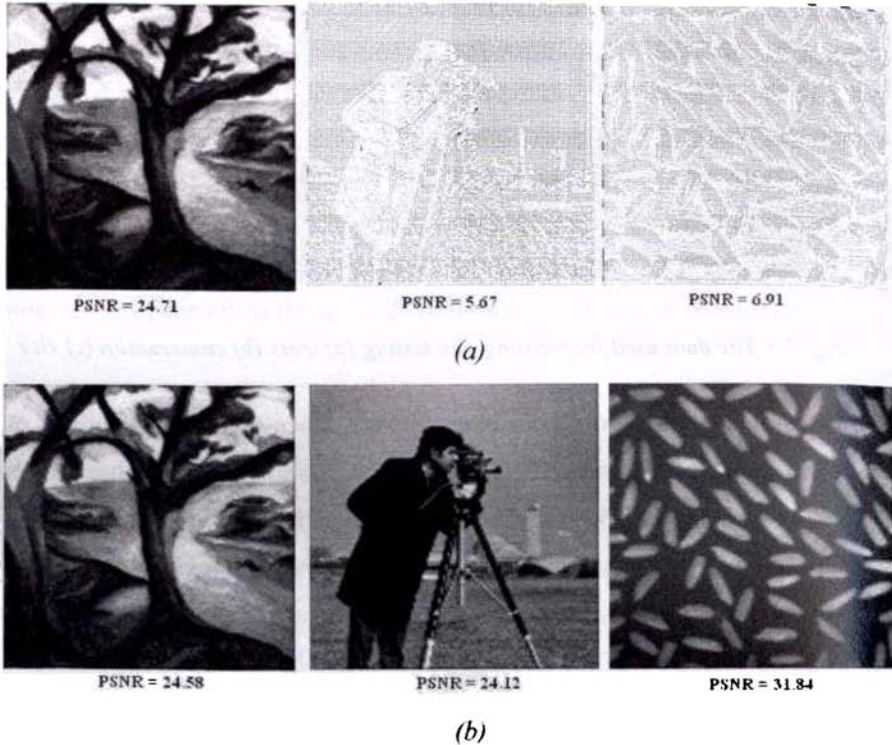
Now, the network is set up with  $I=36$ ,  $J=4$  and  $K=9$ . The images used for training and testing is shown in Fig.5.4. The data used for training the network is the data derived from the Fig. 5.4 (a) trees. For testing the network, data derived from the Fig. 5.4 (b) and (c), cameraman and rice, is used. Trees data consist of values varying between 0 and 1, where 0 corresponds to black and 1 corresponds to white.



**Fig. 5.4** The data used for training and testing (a) trees (b) cameraman (c) rice

Trees do not correspond to a binary image, since there are variations of value in between 0 and 1. But the data of cameraman and rice has values varying from 0 to 255, where 0 corresponds to black and 255 to white. As illustrated in Fig.5.3, the image should be downsampled and split into overlapping sections of 6x6 and the must be arranged lexicographically to 36x1 and the DCT of the pixels are then taken. When this process is done with trees image, the value of DCT is varying in between -1.8549 and 6, whereas the variation for cameraman is from -399.7156 to 1.1410e+003 and that for rice is from -305.6827 to 1.1677e+003.

Now the neural network with 36 input neurons, 4 hidden layer neurons and 9 output neurons is set for training and testing. Firstly, the selection of the activation function for the network can be encountered. The performance of the network is measured as the PSNR given in Eq. 4.7. Here the value of  $M$  in Eq.4.7 is 1 for the tree image and 255 for the cameraman and the rice image. The activation functions for a problem are selected depending upon the data available for the neural network. As discussed in chapter 1, there are a number of activation functions or squashing functions.



**Fig. 5.5** Variation of the output of the neural network with activation function and the obtained PSNR for each image after 1000 iterations. (a) with hyperbolic tan in the hidden layer and purelinear in the output layer (b) with purelinear in both the layers.

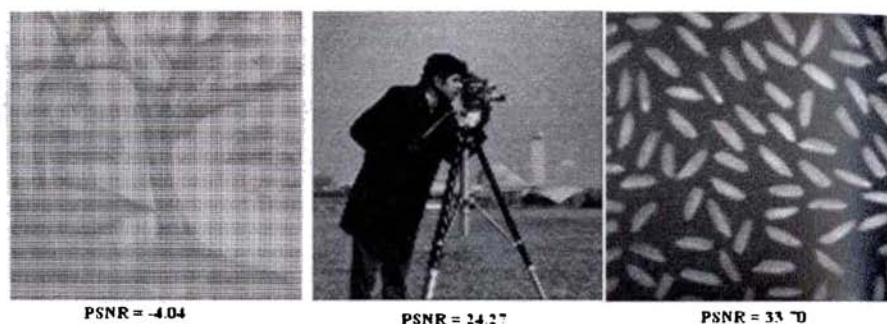
The purpose of these functions is to squash the output to the desired level. Usually, the sigmoid functions for data varying positively and hyperbolic tangent function for data varying both sides are selected. The value of sigmoid function is varying from 0 to 1 and that of hyperbolic tan is from -1 to 1. But for the problem suggested here, there are wide variation for the data as mentioned earlier. Therefore, it was suggested to take the

hyperbolic tangent for the hidden layer neurons and the linear activation function for the output layer. The neural network is trained for 1000 iterations with data derived from the trees image and tested with that of the cameraman image and rice image. The result is illustrated in Fig. 5.5 (a). The PSNR for the tree image is appreciable whereas for the cameraman and rice images the value is very low. This is because the variation of data for trees is very small compared to the others. When the variation in data is low, it corresponds to a better output than those with large variations. Now it is desired to change the activations in order to better the result. A linear activation function is recommended for both the layers. The network is again trained with the training data and tested with the testing data. Fig.5.5(b) shows the result and make the conclusion that both layers must have a linear activation function.

A neural network is well-known for its generalization capability. Even though it is trained with a certain set of data, it can give better outputs even for unseen inputs. The performance of a neural network is measured in terms of its generalization capability. A good neural network is one which can identify almost all patterns fed to it irrespective of the kind of data. Even if, it is trained with a particular type of data, it must be possible for the network to give better outputs for any kind of data given to it. So the selection of input data for training is important. This is illustrated in Fig.5.6. In Fig.5.6 (a), the cameraman image is taken as the training input and the trees and rice images are given as test data. The neural network is trained for 1000 iterations with purelinear activation functions and tested. The generalization of the neural network is poor for the trees data. The same happened when trained with rice image also as in



(a)



(b)



(c)

**Fig. 5.6** Variations of the output of the neural network with the selection of the training data. (a) training data is the cameraman and the others the test data (b) rice is the training data and (c) tree, the training data.

Fig. 5.6 (b). But with the trees data as the training data the generalization was very good as shown in Fig.5.6 (c). So the trees data is selected for further training. This is because the variation for the values of the trees data is smaller and they can catch with the activation functions easily. So in the testing phase they can very well adjust to the input fed and produce a reasonable output. But in the first two cases the activation functions cannot catch up with the input variation and hence the generalization was not good. However trees image was chosen for training the network.

## **5.5 Variations in Backpropagation Algorithms**

The algorithm chosen for training the network is the backpropagation algorithm (Hagan et. al, 2002, Haykin, 2003). When the basic backpropagation algorithm is applied to a practical problem the training may take days or weeks of computer time. So variations of backpropagation algorithms, which are faster, came. The literature says the convergence of these algorithms with the variation of their parameters like learning rate, momentum etc. Also, the literature emphasis on the need for the careful selection of values for these parameters. Here, a discussion is done on the selection of a particular backpropagation algorithm for a problem.

There are different methods to improve the speed of the algorithm. One of the simplest technique is the batch mode processing. In batch mode the weights and biases of the network are updated only after the entire training set has been applied to the network. The faster algorithms fall into two main categories. The first category uses heuristic

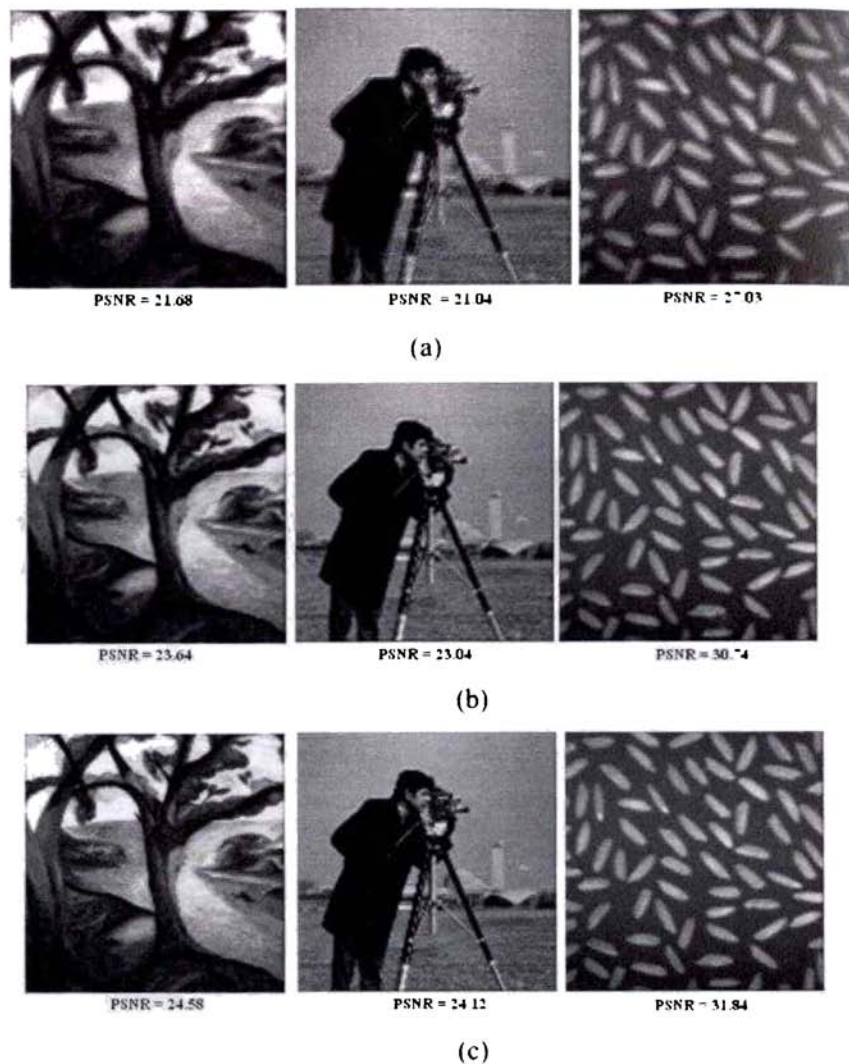
techniques, which were developed from an analysis of the performance of the standard steepest descent algorithm. One heuristic modification is the momentum technique, more heuristic techniques are: variable learning rate backpropagation and resilient backpropagation

With standard steepest descent, the learning rate is held constant throughout training. The performance of the algorithm is very sensitive to the proper setting of the learning rate. If the learning rate is set too high, the algorithm may oscillate and become unstable. If the learning rate is too small, the algorithm will take too long to converge. It is not practical to determine the optimal setting for the learning rate before training, and, in fact, the optimal learning rate changes during the training process, as the algorithm moves across the performance surface. The performance of the steepest descent algorithm can be improved if we allow the learning rate to change during the training process. An adaptive learning rate will attempt to keep the learning step size as large as possible while keeping learning stable. The learning rate is made responsive to the complexity of the local error surface. An adaptive learning rate requires some changes in the training procedure. First, the initial network output and error are calculated. At each epoch new weights and biases are calculated using the current learning rate. New outputs and errors are then calculated. As with momentum, if the new error exceeds the old error by more than a predefined ratio (typically 1.04), the new weights and biases are discarded. In addition, the learning rate is decreased. Otherwise, the new weights, etc., are kept. If the new error is less than the old error, the learning rate is increased. This procedure increases the learning rate, but only to the extent that the network can learn without large error increases.

Thus, a near-optimal learning rate is obtained for the local terrain. When a larger learning rate could result in stable learning, the learning rate is increased. When the learning rate is too high to guarantee a decrease in error, it gets decreased until stable learning resumes. Certain algorithms combines adaptive learning rate with momentum training (Hagan et. al., 2002).

Multilayer networks typically use sigmoid transfer functions in the hidden layers. These functions are often called "squashing" functions, since they compress an infinite input range into a finite output range. Sigmoid functions are characterized by the fact that their slope must approach zero as the input gets large. This causes a problem when using steepest descent to train a multilayer network with sigmoid functions, since the gradient can have a very small magnitude; and therefore, cause small changes in the weights and biases, even though the weights and biases are far from their optimal values. The purpose of the resilient backpropagation training algorithm is to eliminate these harmful effects of the magnitudes of the partial derivatives. Only the sign of the derivative is used to determine the direction of the weight update; the magnitude of the derivative has no effect on the weight update. The size of the weight change is determined by a separate update value. The update value for each weight and bias is increased by a factor whenever the derivative of the performance function with respect to that weight has the same sign for two successive iterations. The update value is decreased by a factor whenever the derivative with respect to that weight changes sign from the previous iteration. If the derivative is zero, then the update value remains the same.





*Fig. 5.7 Variations in the PSNR for the different training algorithms. (a) an algorithm in which adaptive learning rate is used. (b) an algorithm which combines both momentum and the adaptive learning rate. (c) resilient backpropagation training algorithm*

Whenever the weights are oscillating the weight change will be reduced. If the weight continues to change in the same direction for several iterations, then the magnitude of the weight change will be increased.

Figure 5.7 illustrates the variation of the output of the neural network for the above discussed training algorithms for 1000 iterations of each algorithm. Figure 5.7 (a) describes the PSNR of the neural network for the train data and test data for the algorithm in which adaptive learning rate is employed. For most of the problems, the algorithm which combines momentum and adaptive learning rate is suitable as shown in Fig.5.7 (b). But for this problem, resilient backpropagation algorithm is found to give more performance than the others as seen in Fig.5.7(c).

## 5.6 Simulation Results

The neural network is now set up for training and testing. The configuration of the neural network is such that it has 36 input neurons, I, 11 hidden layer neurons, J and 9 output layer neurons, K. It is now decided to have purelinear functions for the hidden layer and output neurons as activation functions, the trees image can be used as the data for training and resilient backpropagation algorithm can be used for training. Fig.5.8 shows the image used for training and testing the network. Fig.5.8 (a) is the actual image and Fig. 5.8 (b) is the downsampled image. Fig.5.8 (b) is split into overlapping sections each of 6x6 pixels size and is lexicographically arranged to a 36x1 matrix, whose DCT is taken and fed to the neural network for training as well as for testing. Fig. 5.8 (a) is the image which is split into overlapping sections of 3x3 pixels size and is arranged to a 9x1 matrix.



(a)

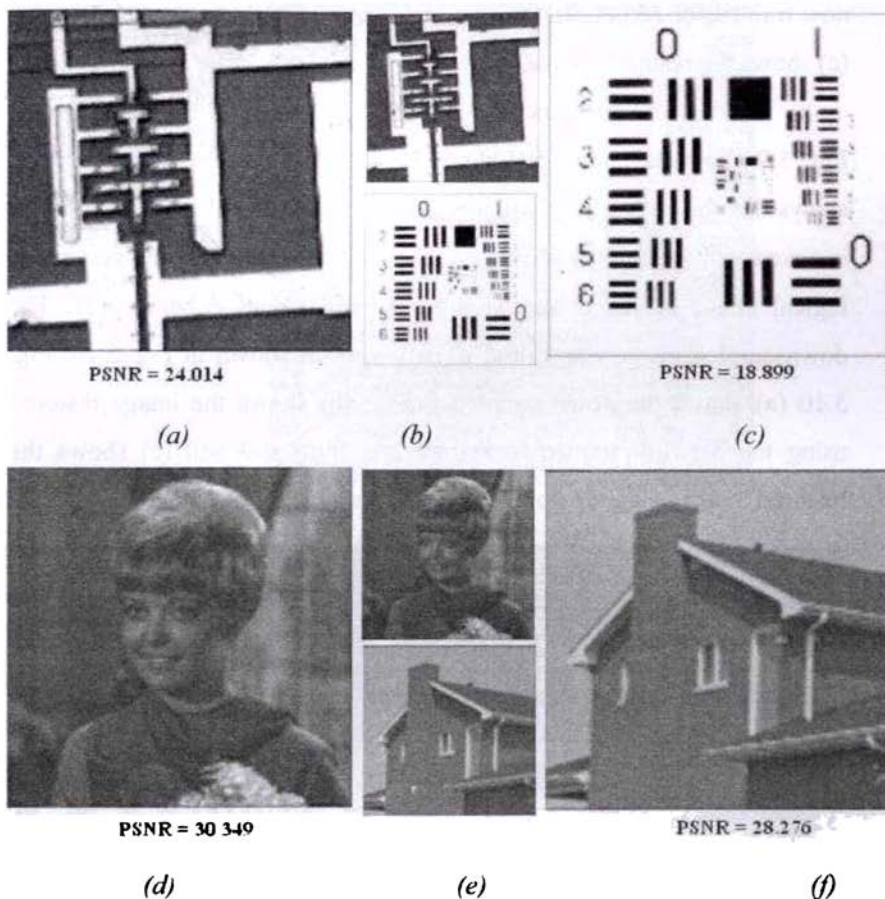


(b)



(c)

**Fig.5.8** The data for training and testing (a) actual data 256 x 256 image (b) down sampled image 128 x 128 image. (c) the restored image using neural network

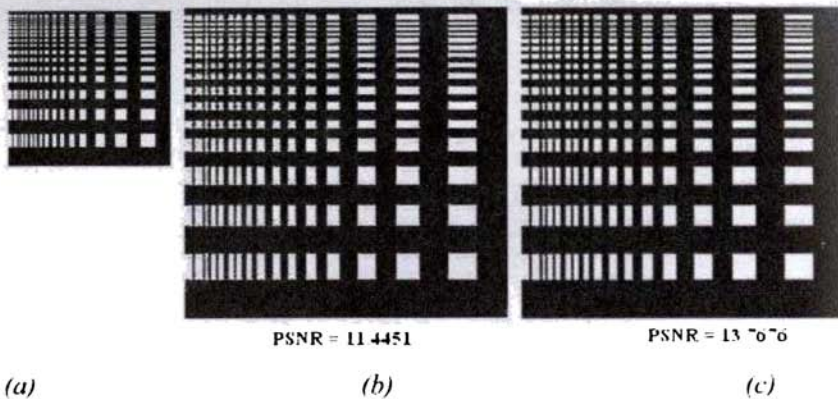


**Fig. 5.9** Various restored images (a) and (c) show the restored images (256x256) of the downsampled images (128x128) shown in (b). Similarly (d) and (f) show the restored images of the downsampled image shown in (e)

For the trees image which is used for training the 9x1 image is used as the target image and others are used to find the PSNR of the test image, which is a measure of the performance of the network. The neural network is

now trained for 25,000 iterations and found to function properly. Fig. 5.8 (c) shows the restored image.

The neural network is now ready to test with different images. Fig. 5.9 illustrates more results. A high PSNR for the untrained data shows that the network is functioning properly. So far the neural network is tested with gray level images. A logical image is fed to the system. A logical image is that image having the pixel values either 0 or 1. The downsampled image is restored in two ways as shown in Fig. 5.10. Fig. 5.10 (a) shows the downsampled image, (b) shows the image restored using the network trained to restore the digits 0-9 and (c) shows the restored image using the now discussed neural network.



**Fig. 5.10 (a) the downsampled image, (128 x 128) (b) the image restored using the network trained to restore the digits 0-9 and (c) the restored image using the now discussed neural network, (256 x 256)**



PSNR = 31.369

(a)

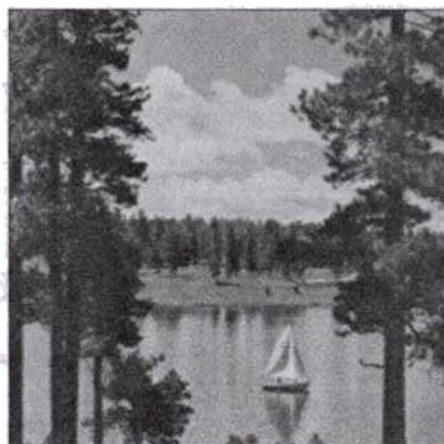


(b)



PSNR = 27.812

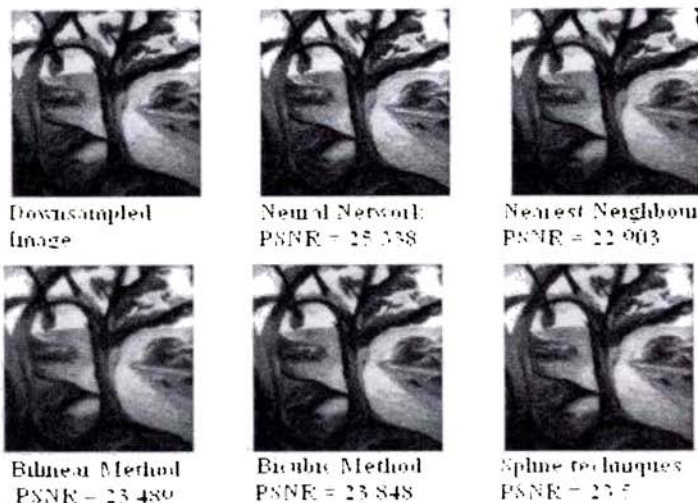
(c)



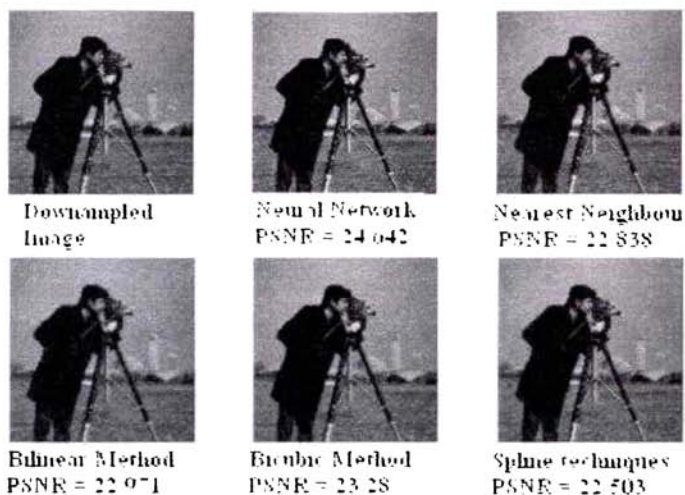
(d)

**Fig. 5.11 (a) and (c) are the downsampled image of size 256x256 and (b) and (d) are the restored image of size 512x512**

It is now proved that the performance of the network is satisfactory for all the  $128 \times 128$  image fed to it. Now it is required to test whether the neural network is able to restore a  $256 \times 256$  image to a  $512 \times 512$  image. It is worthwhile, to note that the current network is trained to restore  $128 \times 128$  to  $256 \times 256$ . The  $512 \times 512$  image is downsampled to  $256 \times 256$  and is split into overlapping sections of size  $6 \times 6$ . It is lexicographically arranged to matrix of  $36 \times 1$  and DCT is taken and is fed to the neural network. The IDCT of the output of the neural network is taken which is a  $9 \times 1$  matrix and is arranged to a  $3 \times 3$  matrix and the image is restored. The output obtained with such a restoration is given in Fig.5.11.



**Fig. 5.12 (a) restoration of image using various interpolation techniques- a comparison  $128 \times 128$  image is restored to  $256 \times 256$  image**



**Fig. 5.12 (b) restoration of image using various interpolation techniques - a comparison 128 x 128 image is restored to 256 x 256 image**

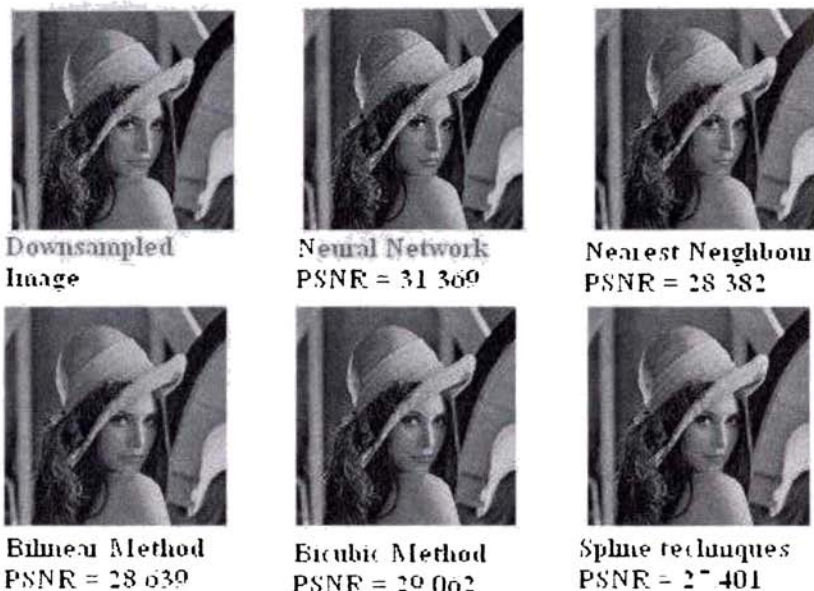
The PSNR of the restored image shows that the restoration is better. Also it shows that it is possible to restore an image of any size to double its size irrespective of the training pattern given to train the neural network. The performance of the neural network is compared with the various existing interpolating methods like nearest neighbour, bilinear, bicubic and spline techniques. The downsampled image is restored using the various interpolating techniques. Fig.5.12 illustrates a comparison.

Table 5.1 gives a comparison of the performance of the different interpolating techniques to the different downsampled images. There about 8 test images and the results show that the image restoration with neural networks is superior to the different techniques available. The

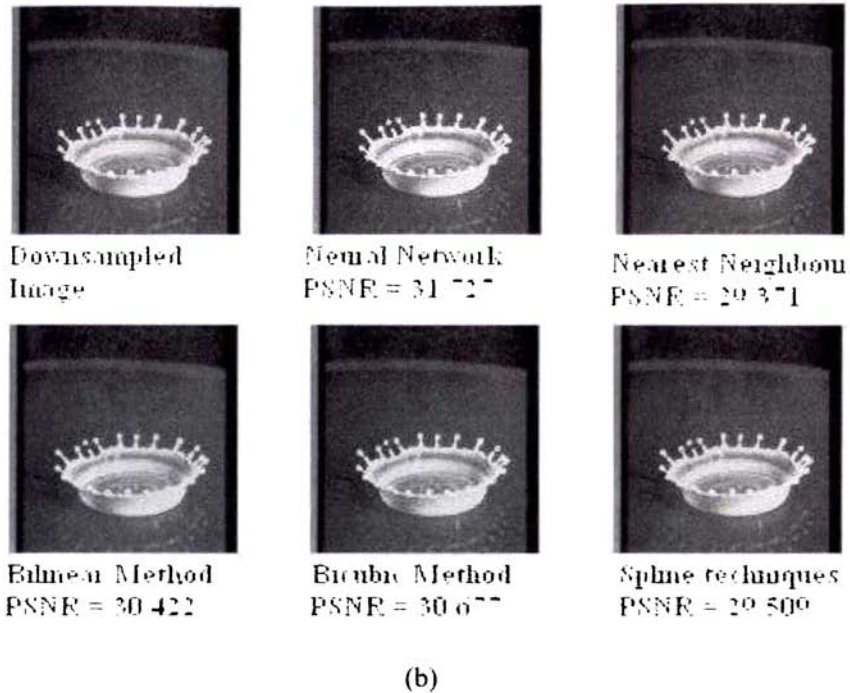


Test Image	Neural Network	Nearest Neighbour	Bilinear Method	Bicubic Method	Spline Techniques
Camerman	24.642	22.838	22.971	23.28	22.503
Rice	33.536	28.271	28.498	29.105	27.668
Trees	25.338	22.903	23.489	23.848	23.5
Test pattern	13.768	12.133	12.168	12.142	13.097
IC image	24.014	21.535	21.587	22.057	20.664
Girl	30.349	28.625	28.637	28.94	26.033
House	28.276	25.776	26.754	27.071	24.942
Pattern	18.899	16.774	16.745	17.086	16.499

**Table 5.1** A comparison of the PSNR of different images with the various interpolation methods 128 x 128 image is restored to 256 x 256 image



**Fig. 5.13 (a)** restoration of image using various interpolation techniques- a comparison 256 x 256 image is restored to 512 x 512 image



**Fig. 5.13 (b) restoration of image using various interpolation techniques- a comparison 256 x 256 image is restored to 512 x 512 image**

network tested was trained to restore an image of size 128 x 128 to 256 x 256.

It is now decided to test the performance of the network to restore a 256 x 256 image to 512 x 512. The same network is used for the restoration. The data is preprocessed as discussed and is fed for restoration. When preprocessed the image is split into overlapping sections of 36 pixels each. The network restored the image and is found to be better in performance compared with the other restoring techniques. Fig. 5.13 gives an illustration of this comparison.

Test Image	Neural Network	Nearest Neighbour	Bilinear Method	Bicubic Method	Spline Techniques
Lenna	31.369	28.382	28.639	29.062	27.401
Boat	27.812	25.116	25.561	25.946	25.685
Baboon	22.673	21.467	21.537	21.688	22.179
Splash	31.727	29.371	30.422	30.677	29.509

*Table 5.2 A comparison of the PSNR of different images with the various interpolation methods 256 x 256 image is restored to 512 x 512 image*

The aforementioned results show that the neural network trained is able to restore an image to twice its size. The work is now directed to the enlargement of the image to four times its original size. When enlarge to four times, the image is first doubled to its original, then from the restored image of double size is again enlarged to double its size. This technique is applied to all the interpolation algorithms. It can be seen that when such restoration is done for 2 or 3 times, the quality of the image gets degraded. This is true with all interpolation algorithms. The neural network trained for interpolating the image to double its size is employed for restoring the image to 4 times its original size. When interpolated to 4 times, the quality of the image is degraded. Here a 128 x 128 image is restored to 512 x 512 image. Fig. 5.14 shows the result of this restoration.

The restoration result with neural network is compared with the various available techniques. The PSNR of each is technique is compared with that of the neural network. Fig. 5.15 shows the result of this comparison. Table 5.3 illustrates the variation of the PSNR of each technique. From these results we conclude that restoration done by neural

network is superior to the commonly used techniques. The neural networks can be considered as a candidate for further works in this field.



(a)



(b)

*Fig. 5.14 (a) and (b) restored images to size 512 x 512 with the images of size 128 x 128 and the intermediate image of size 256 x 256.*



*Fig. 5.15 Restoration of image using various interpolation techniques- a comparison 128 x 128 image is restored to 512 x 512 image with intermediate image of size 256 x 256*

Test Image	Lenna		Boat	
	X 2	X 4	X 2	X 4
Neural Network	26.929	26.265	23.569	22.592
Nearest Neighbour	24.844	23.662	21.703	20.612
Bilinear method	24.949	23.894	22.006	21.089
Bicubic method	25.215	23.989	22.283	21.221
Spline techniques	23.806	23.573	21.892	21.601

*Table 5.3 A comparison of the PSNR of different images with the various interpolation methods 128 x 128 image is restored to 512 x 512 image with the PSNR of the intermediate image of size 256 x 256.*

## Summary

A neural network is trained to restore the image to twice its original size. The quality of the restored image is comparable with that obtained with the existing interpolation methods. The trained neural net can be used to restore any image to double its size, irrespective of the size of the image used for training. The same network can be used to enlarge the image to 4 times its original size. The results of the investigation are promising to consider neural networks as candidate for image restoration applications.

## Reference:

- [1] Ahmed MN, Cooper BE, Love ST, 2001 Document resizing using a multi-layer neural network. NIP17: Intern Conf Dig Print Techn:792-796

- [2] Antigoni Panagiotopoulou and Vassilis Anastassopoulos, 2008 Scanned images resolution improvement using neural networks, Journal Neural Computing & Applications. Pp 39-47
- [3] Athanasios Papoulis, 1975. A New Algorithm in Spectral Analysis and Band Limited Extrapolation, IEEE Trans. circuits syst. 22, . 735-742
- [4] Bayrakeri S D and R. M. Mersereau, 1995. A new method for directional image interpolation, Proc. Int. Conf. Acoust., Speech, Signal Processing, vol. 4, 2383–2386.
- [5] Chang S G, Z. Cvetkovic, and M. Vetterli, 1995. Resolution enhancement of images using wavelet transform extrema extrapolation, Proc. Int. Conf. Acoust., Speech, Signal Processing, vol. 4, 2379–2382
- [6] Collins M, Jong M 2004 Neuralizing target superresolution algorithms. IEEE Geosc Rem Sens Lett 1:318–321
- [7] Condocia Frank M and Jose C Principe, 1999. Super-Resolution of Images Based on Local Correlations, IEEE Trans. on neural networks, vol.10 No. 2, . 372-380
- [8] Craubner S, 2002. Optoelectronic image scanning with high spatial resolution and reconstruction fidelity. Opt Eng 41:381–394
- [9] Darwish A M and M. S. Bedair, 1996. An adaptive resampling algorithm for image zooming Proc. SPIE, vol. 2666, 131–144.
- [10] Davila C A and B R Hunt, 2000. Super-Resolution of Binary Images with a Nonlinear Interpolative Neural Network, Applied Optics, vol.39, No.14, . 2291-2299.

- [11] Davila C A and B R Hunt, 2000. Training of a Neural Network for Image Super-Resolution Based on a Nonlinear Interpolative Vector Quantiser , *Applied Optics*, vol.39, No.20, . 3473-3485.
- [12] Gonzalez R C and Richard E Woods, 2002. *Digital Image Processing, Second Edition*, Pearson Edn.
- [13] Hagan, Martin T Howard B Demuth and Mark Beale, 2002 *Neural Network Design*, first ed., Boston, Thomson Learning
- [14] Haykin, S., 2003. *Neural Networks: A Comprehensive Foundation. Second Edition*, Pearson Education
- [15] Hu H, Hofman PM, Haan G, 2004 Image interpolation using classification-based neural networks. *IEEE Intern Symp Consum Electr* 1:133–137
- [16] Jensen K and D. Anastassiou, 1995 . Subpixel Edge Localization and Interpolation of still Images, *IEEE Trans. on Image Processing*, vol. 4, no.3 285-295
- [17] Jensen K and D. Anastassiou, 1990 .Spatial resolution enhancement of images using non-linear interpolation, in *Proc. IEEE Int. Conf: Acoust. Speech. and Signal Processing* (Albuquerque, NM).
- [18] Karayiannis N B and A. N. Venetsanopoulos, 1991. Image interpolation based on variational principles, *Signal Processing*, vol. 25, 259–288.
- [19] Martinez D M and J. S. Lim, 1989. Spatial interpolation of interlaced television pictures in *Proc. IEEE Int. Conf. Acoust Speech. Signal Processing Scotland*, 1886-1889.



- [20] Martucci S A, 1995. Image resizing in the discrete cosine transform domain, Proc. Int. Conf. Image Processing, vol. 2, 244–247.
- [21] Nathalie Plaziac, 1999. Image Interpolation Using Neural Networks, IEEE Trans. Image Processing, vol.8, No.11, . 1647-1651.
- [22] Netravali A N and B. G. Haskell, 1995. Digital Pictures: Representation, Compression and Standards, 2nd ed. New York: Plenum.
- [23] Pan F, Zhang L, 2003. New image super-resolution scheme based on residual error restoration by neural networks. Opt Eng 42:3038–3046
- [24] Pu H-C, Lin C-T, Liang S-F, Kumar N, 2003. A novel neural network-based image resolution enhancement. IEEE Intern Conf. Fuzz Syst 2:1428–1433
- [25] Schultz R R and R. L. Stevenson, 1994. A Bayesian approach to image expansion for improved definition, IEEE Trans. Image Processing, vol.3, 233–242, 1994.
- [26] Sekiwa D and Taguchi A, 2001. Enlargement of digital images by using multi-neural networks. Electr. Commun Jpn 84:61–69
- [27] Shen H-L, Xin JH, 2004. Spectral characterization of a color scanner by adaptive estimation. J Opt Soc Am 21:1125–1130
- [28] Shinbori E and M. Takagi, 1994. High-quality image magnification applying the Gerchberg-Papoulis iterative algorithm with DCT, Syst. Comput. Japan, vol. 25, no. 6, 80–90.

- [29] Sivakumar K and U B Desai, 1993. Image Restoration Using a Multilayer Perceptron with Multilevel Sigmoidal Function, IEEE Trans. Signal Processing 14, No.5, . 2018-2022
- [30] Unser M, A. Aldroubi, and M. Eden, 1991. Fast B-spline transforms for continuous image representation and interpolation, IEEE Trans. Pattern Anal. Machine Intell., vol. 13, 277–285.
- [31] Valdes M del C, Inamura M (2000) Spatial resolution improvement of remotely sensed images by a fully interconnected neural network approach. IEEE Trans Geosc Rem Sens 38:2426–2430
- [32] Wang. Y and S. K. Mitra, 1991. Motion/pattern adaptive interpolation of interlaced video sequences, Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing (Toronto, Canada). 2829-2832.



## CHAPTER 6

# RECONSTRUCTION OF IMAGES FROM NOISE EMBEDDED DATA

### 6.1 Introduction

The principal sources of noise in digital images arise during image acquisition (digitization) and/or transmission. The performance of imaging sensors is affected by a variety of factors, such as environmental conditions during image acquisition, and by the quality of the sensing elements themselves. For instance, in acquiring images with a CCD camera, light levels and sensor temperature are major factors affecting the amount of noise in the resulting image. Images are corrupted during transmission principally due to interference in the channel used for transmission. Frequency properties of the noise refer to the frequency content in the Fourier sense. When the Fourier spectrum of noise is constant, the noise is usually called white noise. Noise may be considered as random variations or variables characterized by a probability density function (PDF). The following are among the most common PDFs found in image processing applications:

#### **Gaussian Noise**

Because of its mathematical tractability in both the spatial and frequency domains, Gaussian (also called normal) noise models are used

frequently in practice. The PDF of a Gaussian random variable,  $z$ , is given by

$$\rho(z) = \frac{1}{\sqrt{2\pi\sigma}} e^{-(z-\mu)^2/2\sigma^2} \quad (6.1)$$

where  $z$  represents gray level,  $\mu$  is the mean of average value of  $z$ , and  $\sigma$  is its standard deviation and  $\sigma^2$  is called the variance of  $z$ .

### Rayleigh Noise

The PDF of Rayleigh noise is given by

$$\rho(z) = \begin{cases} \frac{2}{b}(z-a)e^{-(z-a)^2/b} & \text{for } z \geq a \\ 0 & \text{for } z < a \end{cases} \quad (6.2)$$

The mean and variance of this density are given by

$$\begin{aligned} \mu &= a + \sqrt{\pi b/4} \\ \sigma^2 &= \frac{b(4-\pi)}{4} \end{aligned} \quad (6.3)$$

### Erlang (Gamma) Noise

The PDF of Erlang noise is given by

$$\rho(z) = \begin{cases} \frac{a^b z^{b-1}}{(b-1)!} e^{-az} & \text{for } z \geq 0 \\ 0 & \text{for } z < 0 \end{cases} \quad (6.4)$$

where the parameters are such that  $a > 0$ ,  $b$  is a positive integer. The mean and variance of this density are given by

$$\begin{aligned}\mu &= \frac{b}{a} \\ \sigma^2 &= \frac{b}{a^2}\end{aligned}\tag{6.5}$$

### Exponential Noise

The PDF of exponential noise is given by

$$\rho(z) = \begin{cases} a e^{-az} & \text{for } z \geq 0 \\ 0 & \text{for } z < 0 \end{cases}\tag{6.6}$$

The mean and variance of this density are given by

$$\begin{aligned}\mu &= \frac{1}{a} \\ \sigma^2 &= \frac{1}{a^2}\end{aligned}\tag{6.7}$$

This PDF is a special case of Erlang PDF with  $b = 1$ .

### Uniform Noise

The PDF of uniform noise is given by

$$\rho(z) = \begin{cases} \frac{1}{b-a} & \text{if } a \leq z \leq b \\ 0 & \text{otherwise} \end{cases}\tag{6.8}$$

The mean and variance of this density are given by

$$\begin{aligned}\mu &= \frac{a+b}{2} \\ \sigma^2 &= \frac{(b-a)^2}{12}\end{aligned}\tag{6.9}$$

### Impulse (salt-and-pepper) Noise

The PDF of (bipolar) impulse noise is given by

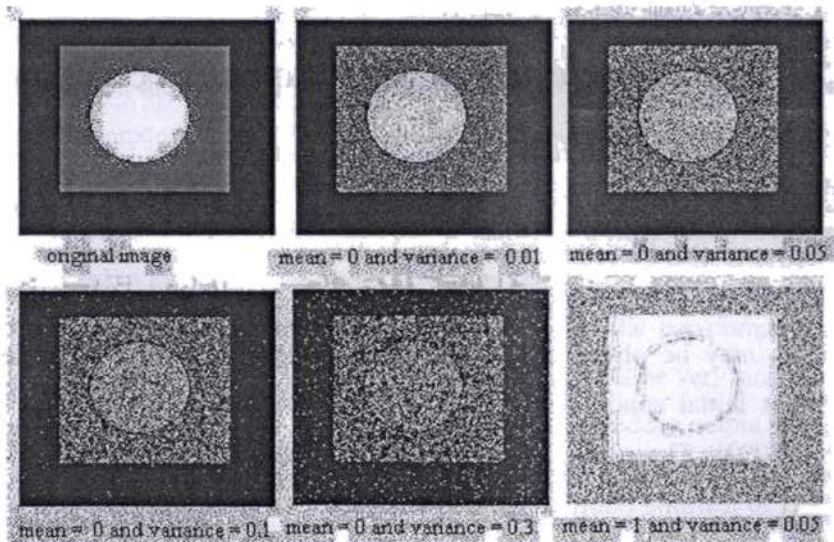
$$\rho(z) = \begin{cases} P_a & \text{for } z = a \\ P_b & \text{for } z = b \\ 0 & \text{otherwise} \end{cases}\tag{6.10}$$

If  $b > a$ , gray-level  $b$  will appear as a light dot in the image. Conversely, level  $a$  will appear like a dark dot. If either  $P_a$  or  $P_b$  is zero, the impulse noise is called unipolar. If neither probability is zero, and especially if they are approximately equal, impulse noise values will resemble salt-and-pepper granules randomly distributed over the image. For this reason, bipolar impulse noise is also called salt-and-pepper noise (Gonzalez and Woods, 2002).

As a group, the preceding PDFs provide useful tools for modeling a broad range of noise corruption situations found in practice. For example, Gaussian noise arises in an image due to factors such as electronic circuit noise and sensor noise due to poor illumination and/or high temperature. The Rayleigh density is helpful in characterizing noise phenomena in range imaging. The exponential and gamma densities find application in laser imaging. Impulse noise is found in situations where

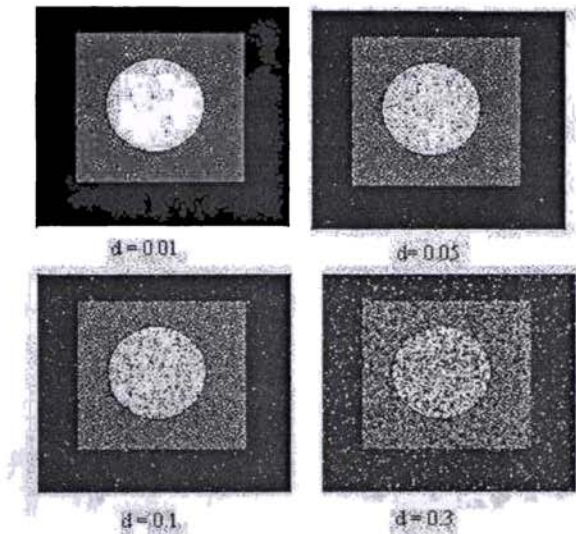
quick transients, such as faulty switching, take place during imaging. The uniform density is perhaps the least descriptive of practical situations.

The most common types of noise that are found affecting the digital images are the Gaussian noise and the impulse noise (salt and pepper noise). The Gaussian noise is characterized by its mean,  $\mu$  and variance,  $v$ . By changing these parameters, the Gaussian distribution is changing according to equation (6.1). Usually the mean is assumed to be zero and the variance is changed. The variation in the digital image with the variation of the mean and the variance is as shown in Fig. 6.1. The salt and pepper noise is characterized by its density. As the density of the noise increases, the image becomes worse. The effect of salt and pepper noise with varied density on an image is as given in Fig. 6.2.



**Fig. 6.1** Variation in image due to Gaussian noise of various mean and variance





*Fig. 6.2 Variation in image due to salt and pepper noise of various densities*

## 6.2 Noise Immunity of Multilayer Perceptrons

Artificial neural networks (ANNs) are not inherently fault tolerant (Segee & Carter, 1994; Pathak & Koren, 1995). In the case of multilayer perceptrons (MLPs), it can be observed that for a fixed structure (a particular number of layers and neurons per layer), different sets of weights may be obtained if the training process is applied by using different initial conditions or parameters of the learning rule (Choi & Choi, 1992). These solutions may present a similar performance with respect to learning (similar mean squared error or classification error) but differ with respect to fault tolerance. In this way, some configurations of weights present a higher tolerance against weight perturbations than others, and similar conclusions can be obtained with respect to tolerance

to input noise or the generalization ability of the MLP (Bernier et. al., 2000).

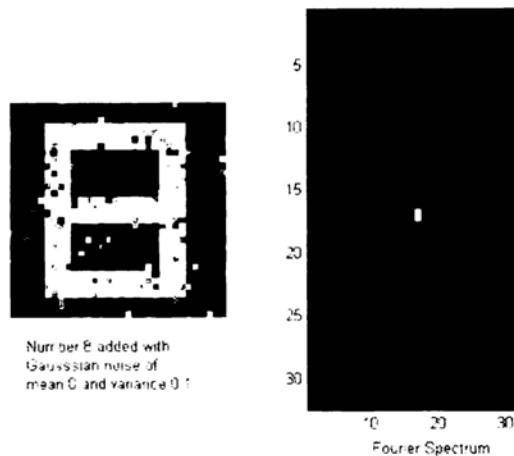
Moreover, when the training process is carried out in a conventional computer and the weights obtained must be mapped onto a physical implementation, the differences between the precision used during training and the accuracy of the implementation can seriously affect the learning performance simulated in the computer. Moreover, in the case of analog implementations, the weight values may vary within a tolerance margin, which also affects the expected performance (Edwards & Murray, 1998).

It is well known that MLPs are robust to noise contamination in inputs and/or weights, including the case of quantization. MLPs have these properties in two ways: First, the orthogonal property among the output values of the hidden nodes reduces the noise effect. It is well known that the hidden weight vectors tend to be near orthogonal through learning procedure for efficient extraction of input patterns (Xue and Hu, 1990). Thus, after successful training, the weighted sums to hidden nodes are much less correlated even when a pattern with correlated noise is presented to the input layer. Also, the magnitude of correlation coefficient between the weighted sums decreases under sigmoidal transformations. Therefore, the correlations among hidden nodes should be very small. As a result, the noise effects are averaged out when the hidden output values are summed through the output weights (Haykins, 2003, Hagan, et. al., 2002). Second, noise immunity of MLPs can be explained in the information theoretic point of view (Abu-Mostafa, 1989). It is reported that MLPs have hierarchical information extraction capabilities acquired

through learning (Lee and Song, 1993). It is argued there that the input pattern set has inter-class information as well as the intra-class variation. The inter-class information is the information content that an input pattern belongs to a specific class, and the intra-class variation is a measure of the average variations within the classes including noise contaminations. After learning, each layer of MLPs tries to keep the inter-class information and to minimize the intra-class variation as possible. When a noisy pattern is presented to the input, MLPs extract the inter-class information and suppress the noise components, yielding noise immunity of MLPs (Youngjik Lee and Sang-Hoon Oh, 1994).

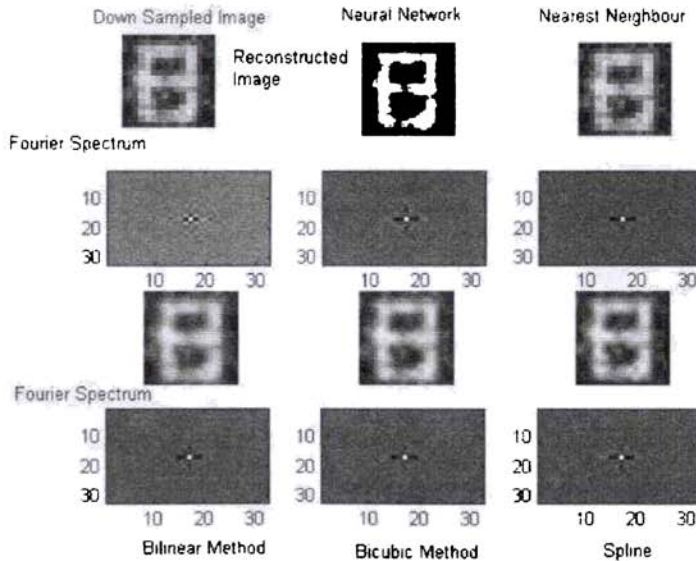
## 6.3 Effect of Noise on Binary Images

### 6.3.1 Effect of Gaussian Noise



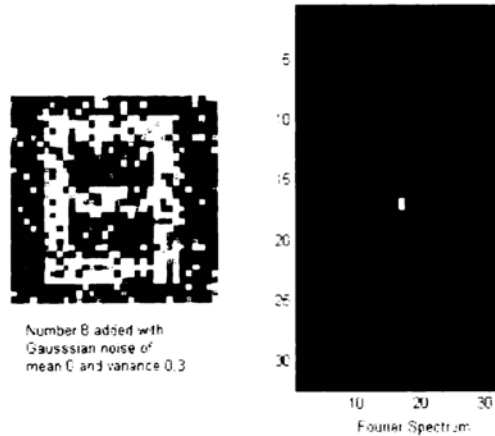
*Fig. 6.3 Number 8 and its Fourier spectrum with added Gaussian noise of mean 0 and variance 0.1*

As we have already discussed in chapter 4, a neural network is trained to superresolve a 16 x 16 binary images of numbers 0-9 to a size of 32 x 32. The neural network is trained with some slanting lines and numbers 0-4 and is tested with numbers 5-9. It is worthwhile to note here that the training images are not contaminated by noise of any kind. They are only filtered by a low pass filter. The noise immunity of the neural network is tested by adding Gaussian noise of mean = 0, and variances ranging from 0 to 0.3. When the variance is increased beyond 0.3, the image is much corrupted and it is difficult to distinguish the original image.



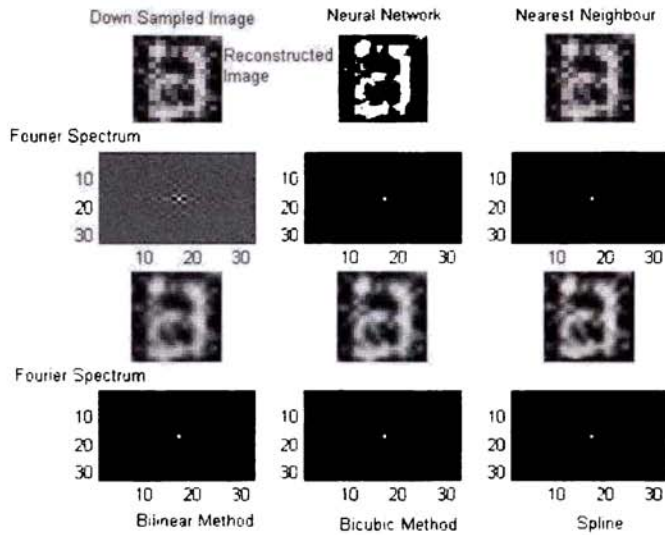
**Fig. 6.4** Down sampled image of number 8 in Fig.6.3 and its Fourier spectrum and the reconstructed image by various interpolation techniques and the corresponding Fourier spectra

For studying the effect of noise on the neural network performances, number 8 and its spectrum is chosen. Fig. 6.3 and Fig. 6.5 shows the number 8 and its Fourier spectrum for an added Gaussian noise of mean 0 and variance 0.1 and variance 0.3

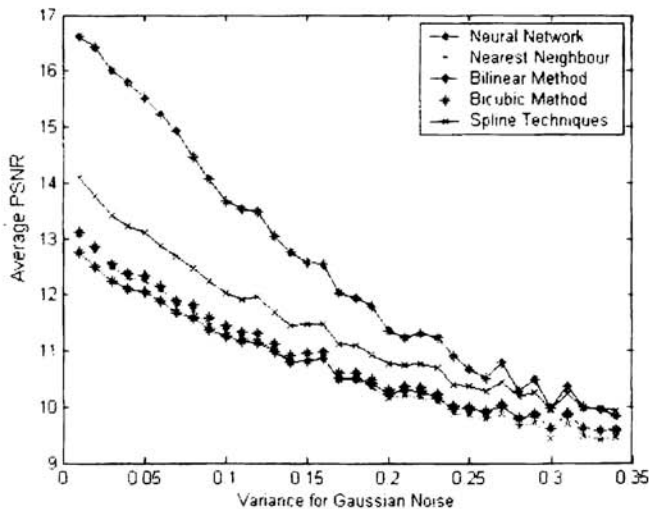


**Fig. 6.5** Number 8 and its Fourier spectrum with added Gaussian noise of mean 0 and variance 0.3

The noise added image of size 32 x 32 is downsampled to a size of 16 x 16. It then reconstructed to a size of 32 x 32 by neural networks and also by the various interpolation methods. Fig. 6.4 and Fig. 6.6 gives the comparison of the results obtained for the interpolation. From these Figures, it can be concluded easily that the performance of the neural network is much better than any other techniques. Gaussian noise of various variances ranging from 0 to 0.34 with mean 0 is added to the number 0-9 and the average PSNR (peak signal to noise ratio) for these numbers for each interpolating technique is compared with that of the neural network. Fig. 6.7 shows the result of this comparison.



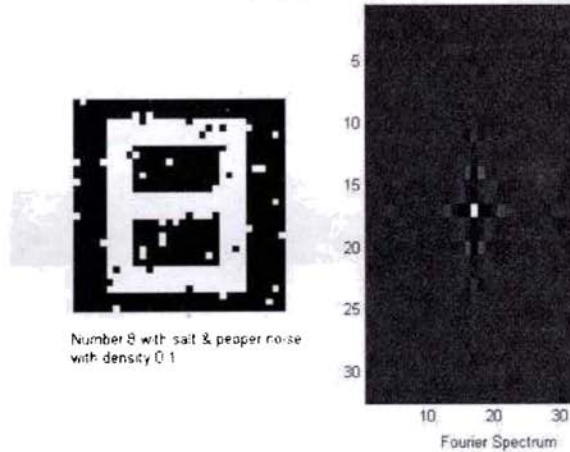
**Fig. 6.6** Down sampled image of number 8 in Fig.6.5 and its Fourier spectrum and the reconstructed image by various interpolation techniques and the corresponding Fourier spectra



**Fig. 6.7** Plot showing the variation of the average PSNR of numbers 0-9 with variation in variance of Gaussian noise with mean 0.

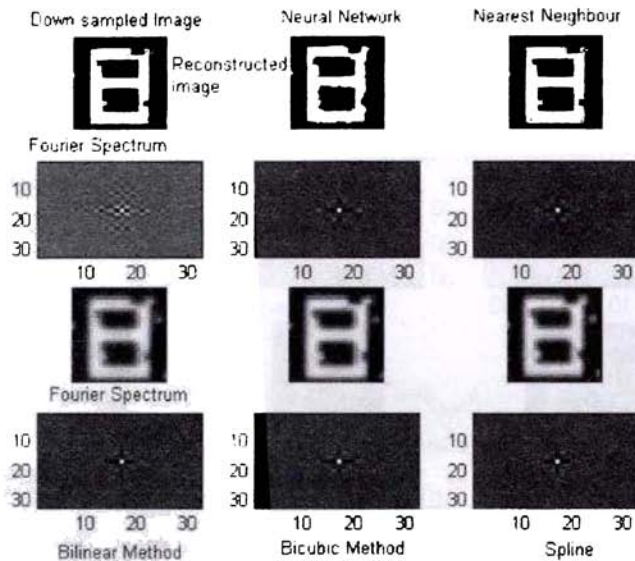
From Fig.6.7, it is clear that the super resolution done with the neural network is better to other interpolation techniques. This result is attributed to the noise rejection capability of the neural network.

### 6.3.2 Effect of salt and pepper noise

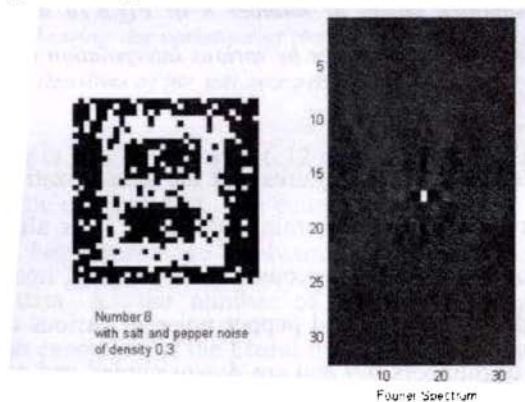


*Fig. 6.8. Number 8 and its Fourier spectrum with added salt and pepper noise of density 0.1*

The neural network to super resolve a binary image of 16x16 size to 32x32 size, is tested with salt and pepper noise of various densities. Fig. 6.8 and Fig. 6.10 show number 8 and the corresponding Fourier spectrum with added salt and pepper noise of density 0.1 and 0.3 respectively. The noise added images in Fig.6.8 and Fig.6.10 are downsampled and reconstructed with the trained neural network and the various interpolation techniques. Fig. 6.9 and Fig. 6.11 show the result of these comparisons. The neural network is giving a response for a pattern which is unknown or unseen by it.

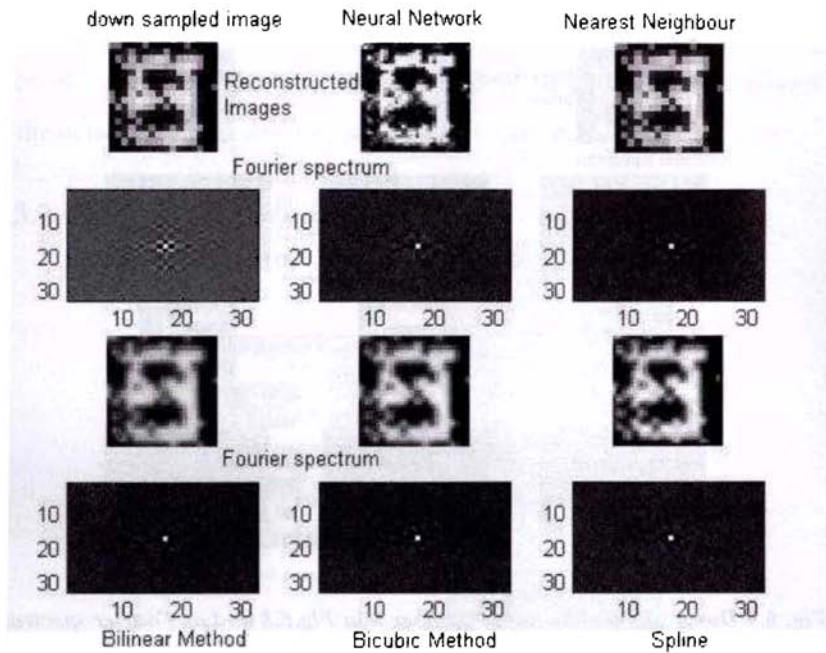


**Fig. 6.9** Down sampled image of number 8 in Fig.6.8 and its Fourier spectrum and the reconstructed image by various interpolation techniques and the corresponding Fourier spectra.



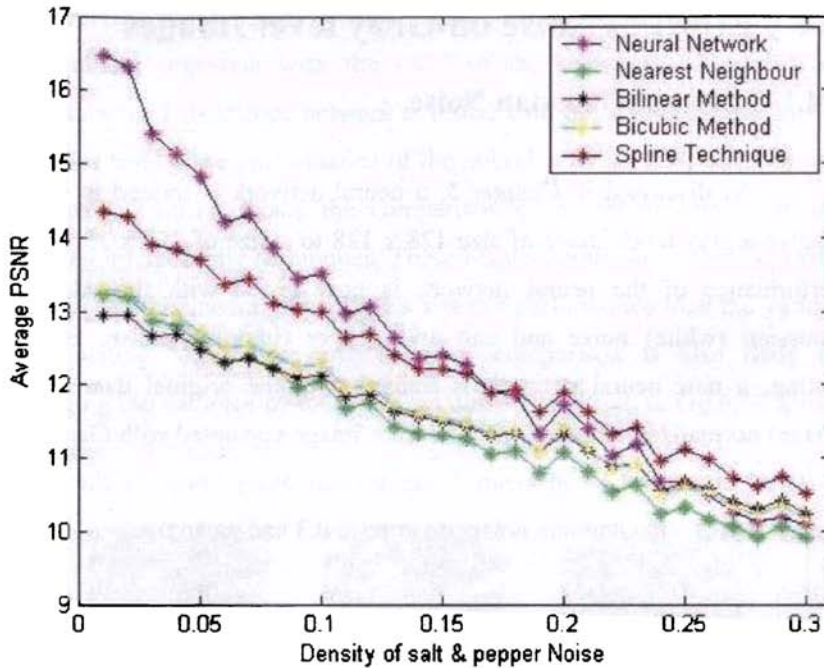
**Fig. 6.10.** Number 8 and its Fourier spectrum with added salt and pepper noise of density 0.3





**Fig. 6.11** Down sampled image of number 8 in Fig.6.10 and its Fourier spectrum and the reconstructed image by various interpolation techniques and the corresponding Fourier spectra

It can be evaluated from these Figures that the generalization capability of the neural network is better, if trained properly. This also shows the ability of the neural networks to reconstruct the original image from the noise corrupted data. The salt and pepper noise of various densities are added to images of numbers 0-9 and are downsampled and reconstructed with the various interpolation methods and neural network. The average PSNR of these data are taken and Fig.6.12 shows the result of these comparisons.



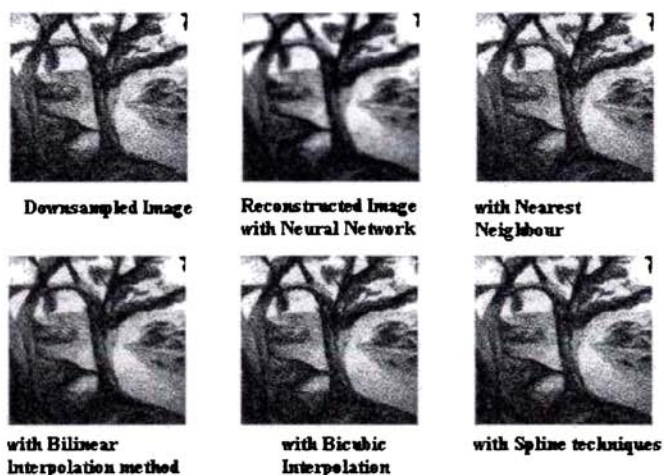
**Fig. 6.12** Plot showing the variation of the average PSNR of numbers 0-9 with variation in the densities of the salt and pepper noise.

Results shown in Fig. 6.7 and Fig. 6.12 are obtained with a neural network trained with the original data. The neural network is not trained with the noisy data. A better result can be obtained if the neural network is trained with noisy data. As the number of training patterns increases, the generalization capability of the neural network also increases. Smaller the size of the training set, the poorer is the generalization capability (Basheer and Hajmer, 2000). Better performance can be expected with a neural network trained by considering all variations in the data.

## 6.4 Effect of Noise on Gray level Images

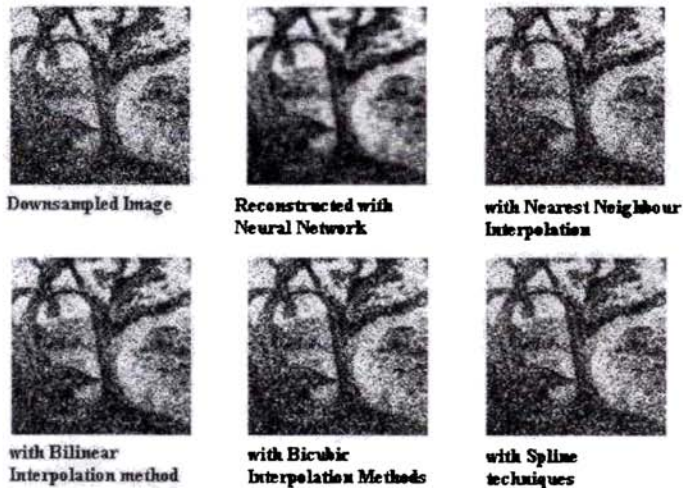
### 6.4.1 Effect of Gaussian Noise

As discussed in Chapter 5, a neural network is trained to super resolve a gray level image of size  $128 \times 128$  to a size of  $256 \times 256$ . The performance of the neural network is now tested with the addition Gaussian (white) noise and salt and pepper (impulse) noise. Before testing, a new neural network is trained with the original data (trees image) normalized and also with the trees image corrupted with Gaussian noise of variance 0.1 and mean 0 and another noisy image of the same image added with Gaussian noise of variance 0.3 and mean 0.

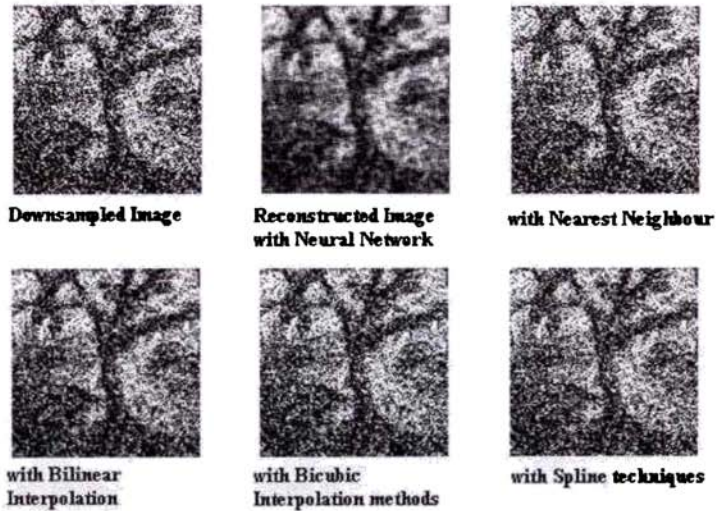


*Fig. 6.13 Downsampled image with Gaussian noise of variance 0.01 and mean 0 and the reconstructed image with the various interpolating techniques.*

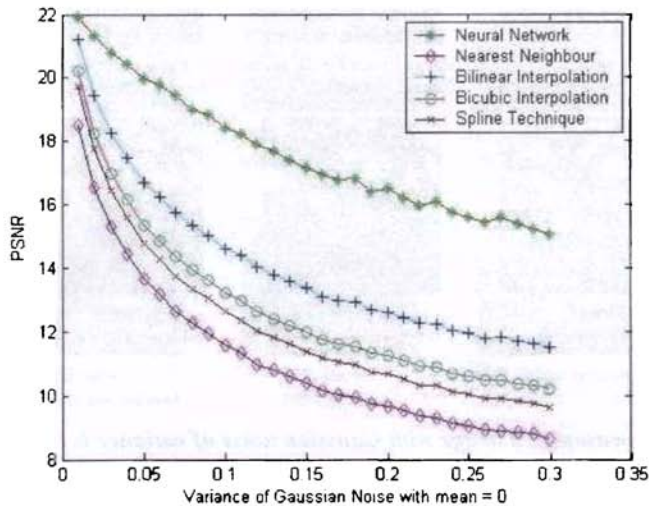
The training of the neural network is done as mentioned in Chapter 5, as overlapping segments with the DCT of the segments taken and are interpolated. This trained network is tested with the testing images and is used for testing the performance of the neural network with added noise. Figures 6.13-6.15 shows the comparison of neural network with the existing interpolating techniques. These results points out to the fact that a properly trained neural network has a better performance than the various interpolating techniques now used. A comparison is also done by changing the variance of the Gaussian noise with mean 0. Fig.6.16 shows the plot for this comparison.



**Fig. 6.14** Downsampled image with Gaussian noise of variance 0.1 and mean 0 and the reconstructed image with the various interpolating techniques.

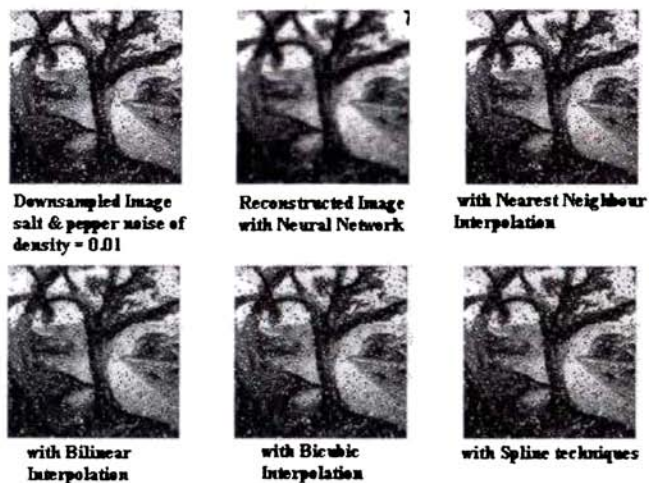


**Fig. 6.15** Downsampled image with Gaussian noise of variance 0.3 and mean 0 and the reconstructed image with the various interpolating techniques.

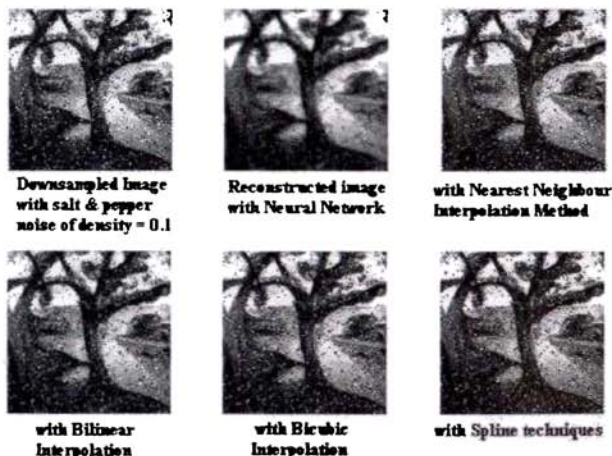


**Fig. 6.16** Plot showing the variation of the PSNR for the normalized trees image with variation in variance of Gaussian noise with mean 0.

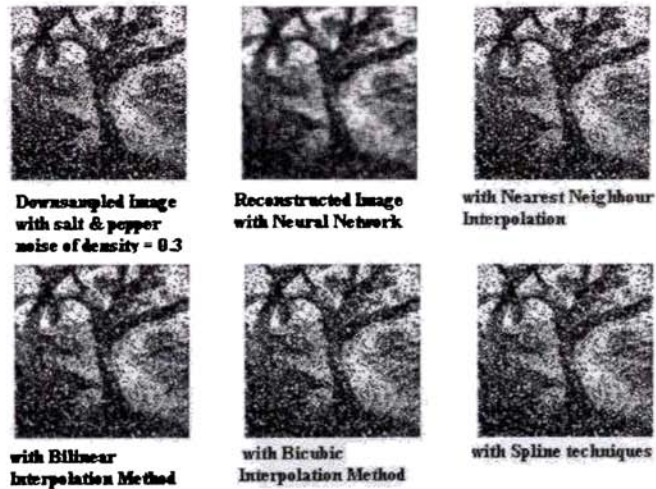
### 6.4.1 Effect of Salt and Pepper Noise



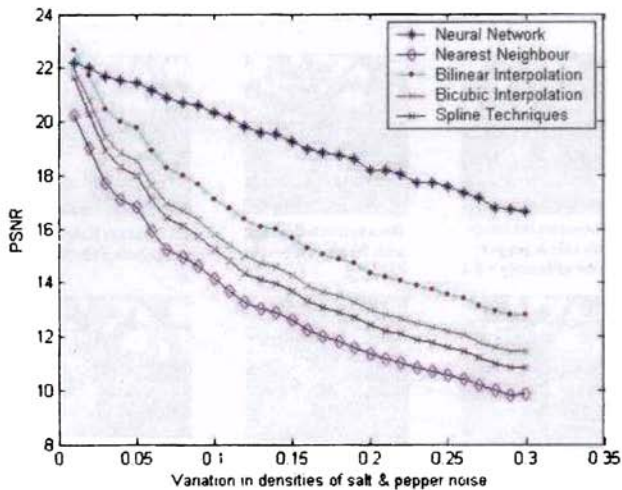
*Fig. 6.17 Downsampled image with salt and pepper noise of density 0.01 and the reconstructed image with the various interpolating techniques.*



*Fig. 6.18 Downsampled image with salt and pepper noise of density 0.1 and the reconstructed image with the various interpolating techniques.*



**Fig. 6.19** Downsampled image with salt and pepper noise of density 0.3 and the reconstructed image with the various interpolating techniques.



**Fig. 6.20** Plot showing the variation of the PSNR for the normalized trees image with variation in densities of salt and pepper noise.

The neural network trained is now tested with impulse noise of varied densities. The neural network is trained only with added Gaussian noise and not with salt and pepper noise. So the image added with salt and pepper noise is an unseen image for the neural network. Here again the performance of the neural network is evaluated against the various interpolating methods. Fig. 6.17-Fig. 6.19 shows the improved performance of the neural network.

The noise rejection capability of the neural network is tested with varied densities of the salt and pepper noise and is compared with the other interpolation methods. As shown in Fig.6.20, the noise immunity of neural networks is much higher. A multilayer perceptron can be trained to super resolve binary and gray level images. The advantage of using a neural network is that most of the complex computations are encountered in the training phase and when an input is presented to the network, it is only a mapping between the input and output. Also the noise immunity of such a network is very high.

## Summary

Two neural networks are trained: one to super resolve a binary image and another to super resolve a gray level image. The two network paradigms are tested with added Gaussian noise and salt and pepper noise. A comparison of the neural networks with the available interpolation algorithms is done. The performance of the neural network is found to be better than the existing interpolation methods.



## References:

- [1] Y S Abu-Mostafa, 1989. Information theory, complexity and neural networks, *IEEE Communication Magazine*, 22-28
- [2] Basheer I A and M Hajmeer, 2000 Artificial neural networks: fundamentals, computing, design and application, *Journal of microbiological methods* 43 pp. 3-31
- [3] Bernier J L, J Ortega, E Ros, I Rojas, A Prieto, 2000. A Quantitative Study of Fault Tolerance, Noise Immunity , and Generalization Ability of MLPs, *Neural Computation* 12, 2941-2964.
- [4] Choi, J. Y., & Choi, C. 1992. Sensitivity analysis of multilayer perceptron with differentiable activation functions, *IEEE Trans. on Neural Networks*, 3(1), 101–107.
- [5] Edwards, P. J.,& A. F.Murray. 1998.Fault tolerance via weight noise in analog VLSI implementations ofMLPs—A case study with EPSILON. *IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing*, 45(9), 1255–1262.
- [6] Gonzalez R C and Richard E Woods, 2002. *Digital Image Processing, Second Edition*, Pearson Edn.
- [7] Hagan, Martin T Howard B Demuth and Mark Beale, 2002 *Neural Network Design*, first ed., Boston, Thomson Learning
- [8] Haykin, S., 2003. *Neural Networks: A Comprehensive Foundation*. Macmillan, New York.
- [9] Pathak, D. S., & Koren, I. 1995. Complete and partial fault tolerance of feedforward neural nets. *IEEE Transactions on Neural Networks*, 6(2), 446–456.

- [10] Segee, B. E., & Carter, M. J. 1994. Comparative fault tolerance of parallel distributed processing networks. *IEEE Trans. on Computers*, 43(11), 1323–1329.
- [11] O.Xue, Y.Hu and W J Tompkins, 1990. Analyses of the hidden units of backpropagation model by singular value decomposition, *Proc. IJCNN'90 San Diego*, vol.1 739-742
- [12] Y Lee and H K Song, 1993. Analysis on the efficiency of pattern recognition layers using information measures, *Proc. IJCNN'93 Nagoya*, vol III, 2129-2132
- [13] Youngjik Lee and Sang-Hoon Oh, 1994. Noise Immunity of Multilayer Perceptrons, *ETRI Journal*, Vol. 16, No.1 35-43



## FUTURE SCOPE

Neural Network is a very effective computational tool. It finds applications in almost every field of signal processing. In this thesis, two applications of neural network are dealt with. In the field of spectroscopic analysis, neural network is found to be very effective. As a further development, a neural network is trained to identify the elements present in a sample irrespective of the spectra taken by any type of spectrometer. Also a neural network can be trained to find the concentration of each element present in the sample. The number of spectral lines obtained will depend on the concentration of each element. Only the persistent lines are obtained with low concentrations.

Neural network has found applications in image processing also. Multilayer perceptrons are good function approximators. But they are not widely used in super resolution algorithms. Here a neural network trained with backpropagation algorithms are found to be very good in super resolving images. As advancement in the studies, networks can be trained with wavelet transforms instead of DCT. In image hiding techniques artificial neural networks find an extensive applications. Several image processing applications like fingerprint identification, face recognition, cryptography etc. neural networks can be employed. Future studies are directed to these fields to employ neural networks as effective computational tools.