APPLICATION OF ARTIFICIAL NEURAL NETWORKS TO DIGITAL IMAGE COMPRESSION

A thesis Submitted to the Graduate School of Natural and Applied Science of
Dokuz Eylül University
In Partial Fulfillment of the requirements for the Degree of Master of Science in
Electrical and Electronics Engineering

98318

by
Müjdat AYBAR

November, 2000
İZMİR

T.C. YÜKSEKÖĞRETİM KURULU
DOKÜMANTASYON MERKEZİ

Ms.Sc. THESIS EXAMINATION RESULT FORM

We certify that we have read the thesis, entitled "Application of Artificial Neural Networks to Digital Image Compression" completed by Müjdat AYBAR under supervision of Yard. Doç. Dr. Yavuz ŞENOL and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

.....

Yrd. Doç. Dr. Yavuz ŞENOL

(Supervisor)

.....

Doç. Dr. Betir KARLIK

(Committee Member)

.....

Yrd. Doç. Dr. Mehmet KUNTALI

(Committee Member)

Approved by the

Graduate School of Natural and Applied Sciences

Prof. Dr. Cahit Helvacı

(Director)

ACKNOWLEDGEMENTS

I want to thank to my advisor assistant professor Yavuz ŞENOL for his invaluable guidance and to research assistant İlker KILIÇ for sharing me his expertise on digital image compression.

I want to thank to my friend Özgür SAĞLAM for his helps durig preparation of this thesis.

Müjdat AYBAR

ABSTRACT

The basic objective of this thesis is to investigate the use of artificial neural networks for digital image compression. Compression of digital images has been a topic of research for many years and a number of image compression standards has been created for different applications. The role of compression is to reduce bandwidth requirements for transmission, and memory requirements for storage of all forms of data. While today more than ever before new technologies provide high speed digital communications and large memories, image compression is still of major importance. Because along with the advances in technologies there is increasing demand for image communications, as well as demand for higher quality image printing and display.

Current major trends in image compression include the traditional transform based methods, vector quantization, and wavelet based approaches. In transform coding, an image is subdivided into non-overlapping blocks of nxn pixels. Each block can be considered as a N-dimensional vector, N=nxn, in N-dimensional space. The transform coding is a transform which maps this set of vectors into another M-dimensional space (M<N) such that the inverse transformation to the original N-dimensional space leads to minimum deviation from the original. For the linear transformation coding, the transformation can be written as an *MXN* matrix *W*. The compressed vector *y* can be calculated as $y = Wxo$ and the reconstructed vector can be calculated as $o1 = W^T xy$. The problem here is to find appropriate W that for specified *M*, minimizes the deviation between *o* and *o1*.

In vector quantization, the blocks in the image are entered into a training algorithm. Using this training algorithm, a codebook which can represent the image is obtained. The compression is achieved by using this codebook.

Neural networks have recently been employed for image compression mainly for two reasons: their inherent adaptivity and their intrinsic parallel structure. Massively parallel implementations of neural networks have gained interest due to their capability of processing data fast enough to satisfy real time requirements (as the one imposed by video broadcasting or video storage).

In our thesis we focus on applications of artificial neural networks to digital image compression. First we used a two-layered neural network model with the number of neurons in the first layer smaller than that in the second layer. The model is trained with a vector set as an input and as an output at the same time using standard supervised neural network training techniques. After training the resulting first layer weight matrix can be taken as W and the resulting second layer weight matrix will be W^T, and this model can be used as a data compressor. The compression is achieved by applying input vector to the first layer of trained network and using the outputs of the hidden layer as compressed data. Exploiting the generalization capability of the neural network which allows the same network to be used for untrained data, the same W and W^T can be used for untrained images. The performance of the compression system is further improved by using a preprocessing stage of quad-tree based image segmentation. Moreover, DCT-based image technique is also applied to different test images. The performance results for DCT-based and artificial neural network based compression methods are compared.

ÖZET

Bu tezin temel amacı yapay sinir ağlarının dijital görüntülerin sıkıştırılması için kullanımının araştırılmasıdır. Dijital görüntülerin sıkıştırılması uzun zamandır araştırma konusu olmuş ve değişik uygulamalar için birtakım görüntü sıkıştırma standartları oluşmuştur. Sıkıştırmanın rolü her tür bilgi için iletişim bant genişliğinin ve hafıza gereksiniminin azaltılmasıdır. Yeni teknolojiler bugün geçmişten daha çok yüksek hızlı dijital bilgi iletişimi ve yüksek kapasiteli hafızalar sağlamakta ve görüntü sıkıştırması hala büyük önem taşımaktadır. Çünkü teknolojik gelişmelerle paralel olarak görüntü iletişimi ve yüksek kaliteli görüntü baskısı ve gösterimi için artmakta olan bir talep bulunmaktadır.

Şu an için görüntü sıkıştırmada ana trendler geleneksel transform tabanlı yöntemler, vektör kuantalama ve wavelet tabanlı yaklaşımlardan oluşmaktadır. Dönüşüm tabanlı yöntemlerde, sıkıştırılacak görüntü üst üste binmeyen nxn pixellik bloklara ayrışılır. N-boyutlu uzayda her blok N-boyutlu bir vektör olarak düşünülebilir. Dönüşüm kodlaması bu vektör setini başka M-boyutlu bir uzaya (M<N) dönüştüren ve orijinal N-boyutlu uzaya geri dönüşüm sonucunda orijinal bilgiden en az sapmanın olduğu bir dönüşümdür. Lineer dönüşüm kodlaması için dönüşüm MxN boyutlu W matrisi olarak yazılabilir. Sıkıştırılmış vektör y, $y = Wxo$ olarak ve geri dönüştürülmüş vektör de $o1 = W^T xy$ olarak hesaplanabilir. Buradaki problem belirli bir M değeri için o ve $o1$ arasındaki sapmayı minimize edecek W matrisinin bulunmasıdır.

Vektör kuantalamada. görüntü blokları bir eğitim algoritmasına tabi tutulup sonuçta resmi temsil edebilecek bir vektör grubu oluşturulur. Sıkıştırma işlemi bu vektör grubunun kullanımıyla yapılır.

Son zamanlarda yapay sinir ağları görüntü sıkıştırmasında genel olarak iki amaç için kullanılmaya başlandı; doğal adaptiviteleri ve gerçek paralel mimarileri. Yapay sinir ağlarının ağır paralel uygulamaları, onların gerçek zaman ihtiyaçları için gerekli olan, bilgiyi yeterince hızlı işleyebilmeleri açısından (video yayını veya video bilgisinin saklanmasında gerektiği gibi) ilgi görmektedirler.

Tezimizde yapay sinir ağlarının dijital görüntülerin sıkıştırılmasında kullanımı üzerinde çalışılmıştır. İlk olarak birinci katmanındaki nöron sayısı ikinci katman nöron sayısından az olan bir yapay sinir ağı kullanılmıştır. Bu model giriş ve çıkış olarak bir vektör seti ve standart denetlenmiş sinir ağı öğrenme tekniği kullanılarak eğitildi. Eğitimden sonra elde edilen birinci katman katsayı matrisi W ve ikinci katman katsayı matrisi W^T olarak alınıp bu model veri sıkıştırıcısı olarak kullanılabilir. Sıkıştırma, vektörün eğitilmiş ağın ilk katmanına uygulanması ve saklı katmanın çıkışlarının sıkıştırılmış bilgi olarak kullanılması ile elde edilmektedir. Aynı ağın öğrenme setinde olmayan bilgiler için kullanımına izin veren, yapay sinir ağlarının genelleme yeteneklerini kullanarak aynı W ve W^T öğrenme setinde olmayan görüntüler için de kullanılabilirler. Sistemin performansı, görüntünün, quad-tree tabanlı görüntü segmentasyon yöntemi ile işlendikten sonra sıkıştırılması ile artırılmıştır. Ayrıca DCT tabanlı sıkıştırma yöntemi değişik dijital görüntülere uygulanmış ve test sonuçları nöral network tabanlı sıkıştırma için elde edilen sonuçlarla karşılaştırılmıştır.

CONTENTS

Page

CONTENTS.....VI

LIST OF TABLES.....IX

LIST OF FIGURES X

Chapter One

INTRODUCTION TO DIGITAL IMAGE PROCESSING

1.1 Digital Image Representation.....2

1.2 Resolution 4

1.2.1 Spatial Resolution.....4

1.2.2 Brightness Resolution.....5

1.2.3 Color Resolution.....6

1.3 Discrete Fourier Transform.....6

1.4 Region Oriented Segmentation.....10

1.4.1 Basic Formulation.....10

1.4.2 Region Growing by Pixel Aggregation.....11

1.4.3 Region Splitting and Merging.....12

Chapter Two

ARTIFICIAL NEURAL NETWORKS

2.1 Neural Network Learning Rule.....19

2.2 The Mathematics.....22

Page

2.3 Multilayer Perceptron Algorithm.....25
2.4 Visualizing Network Behavior.....26
2.5 Generalization.....27
2.6 Fault Tolerance.....28
2.7 Learning Difficulties.....28

Chapter Three

DIGITAL IMAGE COMPRESSION TECHNIQUES

3.1 Need for Data Compression.....31
3.2 Historical Background on Image Compression and Compression standards.....30
3.3 Bandwidth, Memory, Computations and Compression.....32
3.4 Image Quality Metrics.....33
3.5 General Principles Behind Compression.....35
3.6 Different Classes of Compression Techniques.....35
3.6.1 Lossless vs. Lossy compression.....35
3.6.2. Predictive vs. Transform coding.....36
3.6.3 A Typical Image Coder.....36
3.6.4 Source Encoder (or Linear Transformer) 37
3.6.5 Quantizer.....37
3.6.6 Entropy Encoder.....37
3.7 Discrete cosine Transform Coding.....38
3.8 JPEG : DCT-Based Image Coding Standard.....40
3.9 Vector Quantization.....41
3.9.1 LBG Algorithm.....43
3.10 Neural Networks and Image Data Compression.....44
3.10.1 The Use of Multilayer Feedforward ANN's as Image Data Compressor..46

Chapter Four

SIMULATIONS AND RESULTS

Page

4.1 Performance Results for DCT-Based Compression.....48

4.2 Simulations with Neural Networks.....51

4.2.1 Training.....51

4.2.2 Compression.....53

4.2.3 Decompression.....53

4.2.4 Preprocessing with Region-Based Segmentation.....57

4.2.5 Fault tolerance test.....61

Chapter Five

CONCLUSIONS

5.1 Future work 65

REFERENCES.....66

LIST OF TABLES

Page

Table 3.1 Multimedia data types and uncompressed storage space, transmission bandwidth, and transmission time required.....31

Table 4.1 PSNR values for different compression ratios, DCT-based method tested on image "Couple".....48

Table 4.2 Number of hidden layer nodes/ DCT coefficients used in decompressed image versus PSNR value for image "Zelda".....54

Table 4.3 PSNR values for different images, a network with 8 hidden layer nodes is trained with image "Zelda".....55

Table 4.4 PSNR values for compression systems using preprocessing stage the system without preprocessing stage.....59

LIST OF FIGURES

Page

Figure 1.1 The Image Coordinate System Definition....3

Figure 1.2 Sampling a Continuous Function....8

Figure 1.3 Example of Region Growing using known starting points....11

Figure 2.1 The basic features of a biological neuron. 16

Figure 2.2 Outline of the basic model. 17

Figure 2.3 A layered structure of perceptrons....20

Figure 2.4 Sigmoidal threshold function....21

Figure 2.5 The energy function in one dimensional, as we vary one weight, for a fixed pattern. 27

Figure 3.1 A Typical Lossy Signal/Image Encoder....37

Figure 3.2 DCT–Based Compression Block Diagram....39

Figure 3.3 DCT-Based Decompression Block Diagram....39

Figure 3.4 Zig-Zag sequence....40

Figure 3.5 (a) JPEG Encoder Block Diagram....40

Figure 3.5 (b) JPEG Decoder Block Diagram....41

Figure 3.6 Block Diagram of simple VQ....42

Figure 3.7 Feedforward artificial neural network with a single hidden layer....47

Figure 3.8 Feedforward artificial neural network used as compressed data link....47

Figure 4.1 DCT-Based compression applied to image "Zelda" for different compression ratios....49

Figure 4.2 DCT-Based compression applied to image "Zelda" for different compression ratios....50

Page

Figure 4.3 The sum squared error vs. number of training cycles for a neural network having 9 hidden layer nodes with 100 training vectors.....52
Figure 4.4. Reconstructed images, network is trained by image 'Zelda', number of hidden layer nodes is changed.....56
Figure 4.5 Reconstructed images "Zelda" for compression system without preprocessing and for the hybrid compression system, neural networks having 9 hidden layer nodes and trained with the same image are used.....60
Figure 4.6 The effect of removing one hidden layer node output for reconstruction of the image "Zelda".....62

CHAPTER ONE
INTRODUCTION TO DIGITAL IMAGE PROCESSING

Uncompressed multimedia (graphics, audio and video) data requires considerable storage capacity and transmission bandwidth. Despite rapid progress in mass-storage density, processor speeds, and digital communication system performance, demand for data storage capacity and data-transmission bandwidth continues to outstrip the capabilities of available technologies. The recent growth of data intensive multimedia-based web applications have not only sustained the need for more efficient ways to encode signals and images but have made compression of such signals central to storage and communication technology.

Neurocomputing is a new and stormily developing area of science and technology. Scientific activity in this field manifests itself in the large number of international conferences (up to 100 Per year) and specialized journals (exceeding 50) [15]. Neural network products market grows by approximately 40% annually and now approaches the billion dollar level, typical for the beginning of mass production [29].

Neural network algorithms combine high parallelism of computations with relatively simple and flexible programming. Traditional programming is replaced with learning from examples. Such data driven algorithms are capable of real-time-self-adaptation. In this work we focus on applications of artificial neural networks to image data compression.

The chapters of this thesis are organized as follows. In this chapter basics of digital image processing; digital image representation, image resolution, Discrete Fourier transform, and region oriented image segmentation are presented.

In the second chapter artificial neural networks are discussed. Neural network learning rule, the general properties of multilayer neural networks, and learning difficulties of multilayer neural networks are explained.

Two of the major conventional approaches to image compression; Discrete cosine transform coding, and vector quantization are briefly reviewed in chapter three. Different classes of image compression techniques; lossles vs. lossy, predictive vs. transform based, are explained. Artificial neural network based image compression technique which is an alternative to these approaches in some ways is also discussed. JPEG which is based on discrete cosine transform coding and is an international standard for digital image compression and LBG, the basic algorithm for the codebook design of vector quantization based compression are explained. The relation between artificial neural networks and digital image processing and the use of multilayer artificial neural networks as an image data compressor are covered.

In the fourth chapter simulations for Discrete cosine transform based method and neural network based compression method are presented. A hybrid compression system that consist of a feedforward artificial neural network with quad-tree based image segmentation preprocessing stage is simulated. It is seen that this approach improves the performance of the compression system.

1.1 DIGITAL IMAGE REPRESENTATION

The term gray level image or simply image, refers to a two dimensional light intensity function $f(x,y)$,where x and y denote spatial coordinates and the value of f at any point (x,y) is proportional to the brightness (or gray level) of the image at that point. Figure 1.1 illustrates the axis convention used throughout this thesis. Sometimes viewing an image function in perspective with the third axis being the

brightness is useful. Figure 1.1 viewed in this way appears as a series of active peaks in the regions or plateaus where the brightness levels varied little or were constant. Assigning proportionately higher values to brighter areas will make the height of components in the plot proportional to the corresponding brightness.

A digital image is an image, $f(x,y)$ that is discretized both in the spatial coordinates and brightness. A digital image can be considered as a matrix whose row and column indices identify a point in the image and the corresponding matrix element value represents the gray level at that point. The elements of such a digital array are called image elements or pixels.

Images can have either digital or analog representation. In the digital representation of gray-level images, the image is represented as a two dimensional array of numbers . Each gray level is presented by 8 bits (1 byte), then the gray level may have any of 256 possible values. These levels are assigned integer values ranging from 0 to 255, with 0 representing the darkest intensity level. In a colored image, the representation is similar except the number represents three primary colors; red, green and blue at each location of the matrix. For a 24 bit color representation, the number is divided into three 8 bit segments. Each segment represents the intensity of the color primaries.

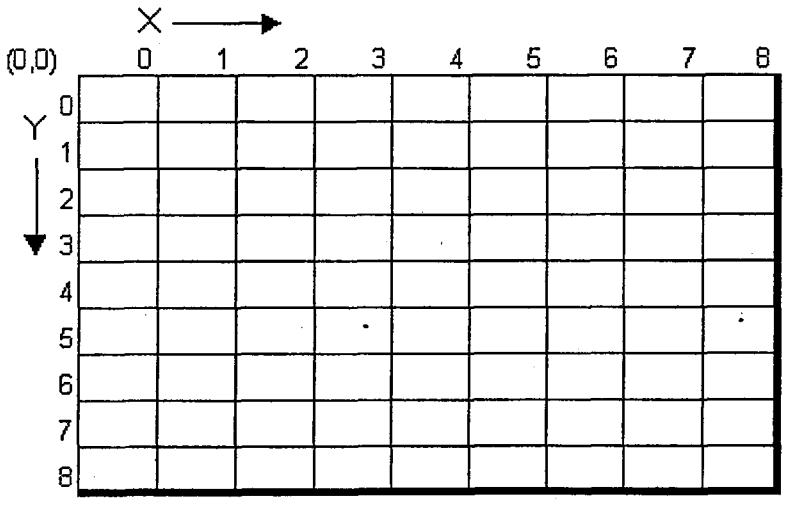


Figure 1.1 The Image Coordinate System Definition

In analog form, images are usually presented as horizontal raster lines. Each line is basically an analog signal carrying the continuous variations of light intensities along a horizontal line in the original scene. For example; images on television sets are displayed through raster scanning.

1.2 RESOLUTION

The quality of the digital image is directly related to the number of pixels (M and N for an MxN image). These aspects are known as image resolution. Image resolution is the capability of the digital image to resolve the elements of the original scene.

For digital images, the resolution characteristics can be broken into two primary groups as spatial and brightness resolution (or color resolution for color images).

1.2.1 Spatial resolution

Spatial resolution is described as how many pixels comprise a digital image. The more pixels in the image, the greater its spatial resolution. The continuous tone image like a photograph or a video camera image, is broken into enough discrete pixels so that the digitized image contains all the information of the original. This means that the displayed digital image would look identical to the original continuous-tone image to an observer.

The concept of spatial frequency explains how finely the image is sampled. The sampling rate must be determined so that the digital image adequately resolves all the spatial details of the original continuous-tone image. Mathematically, the theorem is classical sampling theorem that is used to represent fully the spatial details of the original continuous-tone image. The image is sampled at a rate at least twice as fast as the highest frequency contained in it. This guarantees that both the dark and the light portions of the detail are sampled. The highest spatial frequencies that can be contained in the resulting digital image will not exceed one half of the sampling rate. This frequency is named as Nyquist rate.

If the sampling occurs at a slower rate than required by the sampling theorem, high spatial frequency details will be missed in the digital image. However, the digital image will appear to have less spatial resolution than the original image. Because, in this digital image, there are not enough pixels to represent all of the spatial details of the original image. On the other hand, if sampling occurs at a rate higher than that required by sampling theorem, extra pixels will be created.

1.2.2 Brightness resolution

Every pixel in a digital image represents the intensity of the original image at the spatial location where it was sampled. The digital brightness of pixel represents the intensity of the original image.

The intensity refers to the magnitude of light energy actually reflected from a physical scene. The term brightness refers to the measured intensity after it is acquired, sampled, quantized, displayed and observed. The brightness of a pixel accounts for all the effects induced by the entire imaging system.

The quantization process converts the continuous-tone intensity to a digital brightness value at the sample points. The accuracy of the digital value is directly dependent upon how many bits are used. The brightness can be converted to one of eight gray-levels. In this case, gray-level "0" represents black, gray level "7" represents white and gray levels "1" through "6" represents the ascending gray tones between black and white. The eight gray levels comprise what is called gray-scale, or in this case, the 3-bit gray scale.

The number of bits used in quantizing an image depends on how the image will be used. 8-bit quantization is the most common way used in many applications and generally sufficient.

1.2.3 Color resolution

For color images, the same concepts of sampling, quantization, spatial and brightness resolution are valid. But instead of a single brightness value, color digital images have pixels that are generally quantized using three brightness components. In displaying color, three independent color emitters are used.

At a color video display screen , if it is a cathode ray tube (CRT), liquid crystal display (LCD) or other type, individual dots will be noticed. These dots emit light in the colors of red, green, and blue.

All the colors in the spectrum can be created with the primary red, green, and blue components of each pixel of image. Generally, each component must be quantized at a resolution rate equivalent to the brightness resolution used in gray-scale image.

1.3 DISCRETE FOURIER TRANSFORM

Linear transforms, especially Fourier and Laplace transforms, are widely used in solving problems in science and engineering. The Fourier transform is used in linear systems analysis, antenna studies, optics, random process modeling, probability theory, quantum physics, and boundary-value problems. The Fourier transform, a pervasive and versatile tool, is used in many fields of science as a mathematical or physical tool to alter a problem into one that can be more easily solved.

The Fourier transform, in essence, decomposes or separates a waveform or function into sinusoids of different frequencies which sum to the original waveform. It identifies or distinguishes the different frequency sinusoids and their respective amplitudes [28]. The Fourier transform of f(x) is defined as,

$$F(u) = \int_{-\infty}^{+\infty} f(x)\exp(-j2\pi ux)dx \tag{1.1}$$

where $j = \sqrt{-1}$

Given $F(u)$, $f(x)$ can be obtained by using the inverse Fourier transform.

$$f(x) = \int_{-\infty}^{+\infty} F(u)\exp(j2\pi ux)du \tag{1.2}$$

Equations 1.1 and 1.2, called Fourier transform pair, exist if $f(x)$ is continuous and integrable. These conditions are almost always satisfied in practice.

We are concerned throughout this thesis with functions that are real. The Fourier transform of a real function, however, is generally complex; that is,

$$F(u) = R(u) + I(u) \tag{1.3}$$

where $R(u)$ and $I(u)$ are real and imaginary components of $F(u)$ respectively. It is often convenient to express Equation 1.3 in exponential form, that is,

$$F(u) = |F(u)|\exp(j\phi(u)) \tag{1.4}$$

where

$$|F(u)| = \sqrt{(R^2(u) + I^2(u)} \tag{1.5}$$

and

$$\phi(u) = \tan^{-1}\left[\frac{I(u)}{R(u)}\right] \tag{1.6}$$

The magnitude function $F(u)$ is called the Fourier spectrum of and $\phi(u)$ its phase angle. The square of the spectrum,

$$\begin{aligned} P(u) &= |F(u)|^2 \\ &= R^2(u) + I^2(u) \end{aligned} \tag{1.7}$$

is commonly referred to as the power spectrum of $f(x)$. The term spectral density also is commonly used to denote the power spectrum.

Discrete Fourier transform is obtained by the following method;

Assume that a continuous function $f(x)$ is discretized into a sequence

$$\{f(x_0), f(x_0 + \Delta x), f(x_0 + 2\Delta x), \ldots, f(x_0 + [N-1]\Delta x)\} \tag{1.8}$$

by taking N samples Δx units apart, as shown in Figure 1.2. It will be convenient to use x either a discrete or continuous variable, depending on the context of the discussion. To do so requires defining

$$f(x) = f(x_0 + x\Delta x) \tag{1.9}$$

where x assumes the discrete values 0, 1, 2, 3, ..., N-1. In other words, the sequence $\{f(0), f(1), f(2), ..., f(N-1)\}$ represents any N uniformly spaced samples from a corresponding continuous function. With the above notation, the discrete Fourier transform pair that applies to sampled functions is given by

$$F(u) = \frac{1}{N}\sum_{x=0}^{N-1} f(x)\exp[-j2\pi ux/N] \tag{1.10}$$

for u=0, 1, 2, 3, ..., N-1 and

$$f(x) = \sum_{u=0}^{N-1} F(u)\exp[j2\pi ux/N] \tag{1.11}$$

for x=0, 1, 2, 3, ..., N-1.

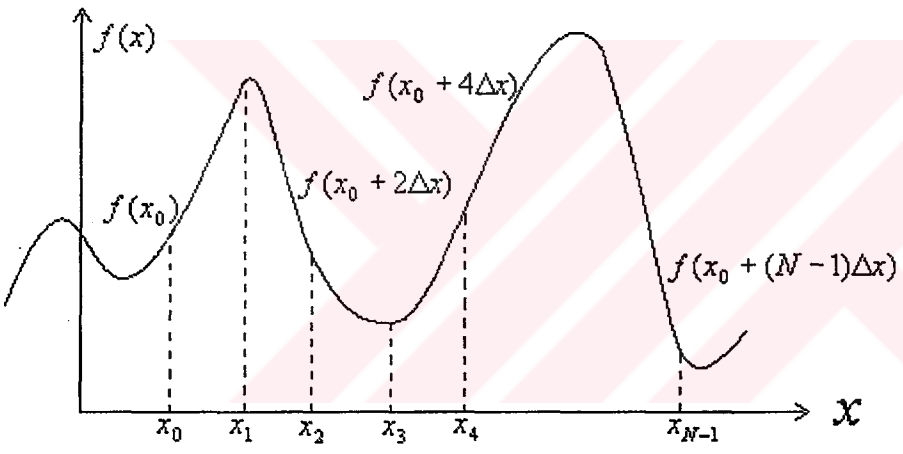


Figure 1.2 Sampling a Continuous Function

The values u=0, 1, 2, ..., N-1 in the discrete Fourier transform (Equation 1.10) correspond to samples of the continuous transform at values 0, Δu, 2Δu, ...,(N-1)Δu. In other words, $F(u)$ represents $F(u\Delta u)$. This definition is similar to that used for the discrete $f(x)$, except that the samples of $F(u)$ start at the origin of the frequency axis. The terms Δu and Δx are given by the expression

$$\Delta u = \frac{1}{N\Delta x} \tag{1.12}$$

In the two dimensional case the discrete Fourier transform pair is;

$$F(u,v) = \frac{1}{MN}\sum_{y=0}^{M-1}\sum_{x=0}^{N-1} f(x,y)\exp[-j2\pi(ux/M + vy/N)] \tag{1.13}$$

for u=0, 1, 2, 3, ..., M-1, v=0, 1, 2, ..., N-1, and

$$f(x,y) = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)\exp[j2\pi(ux/M + vy/N)] \tag{1.14}$$

for x=0, 1, 2, ..., M-1 and y=0, 1, 2, ,,,, N-1

Sampling of a continuous function is now in a two dimensional coordinate system with division of width Δx and Δy in the x and y axis, respectively. In the 1-D case the discrete function $f(x,y)$ represents the samples of the function $f(x_0 + \Delta x, y_0 + \Delta y)$ for x=0, 1, 2, ..., M-1 and y=0, 1, 2, ..., N-1. Similar technique is applied to $F(u,v)$. The sampling increments in the spatial and the frequency domains are given by,

$$\Delta u = \frac{1}{M\Delta x} \tag{1.15}$$

and

$$\Delta v = \frac{1}{N\Delta v} \tag{1.16}$$

When images are sampled in a square array, M=N and

$$F(u,v) = \frac{1}{N}\sum_{y=0}^{N-1}\sum_{x=0}^{N-1} f(x,y)\exp[-j2\pi(ux + vy)/N] \tag{1.17}$$

for u,v=0, 1, 2,...,N-1 and

$$f(x,y) = \frac{1}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1} F(u,v)\exp[j2\pi(ux + vy)/N] \tag{1.18}$$

for x, y=0, 1, 2,...,N-1. Note that 1/N term is in both Equations 1.17 and 1.18. Because $F(u,v)$ and $f(x,y)$ is a Fourier transform pair, the grouping of these constant terms is arbitrary. In practice, images are digitized in square arrays, so we will mostly use the Fourier transform pair in Equations 1.17 and 1.18.

1.4 REGION ORIENTED SEGMENTATION

The main idea behind the segmentation procedure is to obtain a (minimal) set of regions, such that each region is homogenous in relation to one or more pre-defined properties, and differs from all of its neighboring regions regarding those same properties.

1.4.1 Basic formulation

Let R represents the entire region. We may view segmentation as a process that partitions R into $R_1, R_2, \ldots, R_{n1}$, such that

a) $U_{i=1}^{n} R_i = R$,

b) R_i is a connected region, $i = 1,2,\ldots,n$

c) $R_i \cap R_j = \phi$ for all i and j, $i \neq j$,

d) $P(R_i) = TRUE$ for $i = 1,2,\ldots,n$ and

e) $P(R_i \cup R_j) = FALSE$ for all $i \neq j$,

where $P(R_i)$ is a logical predictive over the points in the set R_i and ϕ is the null set.

Condition a) indicates that the segmentation must be complete; that is, every pixel must be in a region. The second condition requires that points in a region must be connected [11]. Condition c) indicates that the regions must be disjoint. Condition d) deals with the properties that must be satisfied by the pixels in a segmented region – for example $P(R_i) = TRUE$ if all pixels in R_i have the same intensity. Finally condition e) indicates that regions R_i and R_j are different in the sense of predicate P.

1.4.2 Region growing by pixel aggregation

As its name implies, region growing is a procedure that groups pixels or subregions into larger regions. The simplest of these approaches is pixel aggregation, which starts with a set of "seed" points and from these grows region by appending to each seed point those neighboring pixels that have similar properties (such as gray level, texture, color). To illustrate this procedure let us consider Figure 1.3.(a), where the numbers inside the cells represent gray level values. Let the points with coordinates (3,2) and (3,4) be used as seeds. Using two starting points results in a segmentation consisting of, at most, two regions: R_1 associated with seed (3,2) and R_2 associated with seed (3,4). The property P to be used to include a pixel in either region is that the absolute difference between gray level of that pixel and the gray level of the seed be less than a threshold T. Any pixel that satisfies this property simultaneously for both seeds is (arbitrarily) is assigned to region R_1. Figure 1.3 (b) shows the result obtained by using $T = 3$. In this case, the segmentation consist of two regions, where the points in R_1 are denoted a's and the points in R_2 by b's. Note that any starting point in either of these two resulting regions would yield the same result. However, choosing $T = 8$, would result in a single region, as Figure 1.3 (c) shows.

0	0	5	6	7
1	1	5	8	7
0	1	6	7	7
2	0	7	6	6
0	1	5	6	5

(a)

a	a	b	b	b
a	a	b	b	b
a	a	b	b	b
a	a	b	b	b
a	a	b	b	b

(b)

a	a	a	a	a
a	a	a	a	a
a	a	a	a	a
a	a	a	a	a
a	a	a	a	a

(c)

Figure 1.3 Example of Region Growing using known starting points: (a) Original image (b) Segmentation result using an absolute difference of less than 3 between intensity levels; (c) result using an absolute difference of less than 8.

The preceding illustration, although simple in nature, points out some fundamental difficulties with region growing. Two immediate problems are the selection of initial seeds that properly represent regions of interest and the selection

of suitable properties for including points in the various regions during the growing process. Selecting a set of one or more starting points often can be based on the nature of the problem. For example, in military applications of infrared imaging, targets of interest generally are hotter (and thus appear brighter) than the background. Choosing the brightest pixels then is a natural starting point for a region-growing algorithm.

The selection of similarity criteria depends not only on the problem under consideration, but also on the type of image data available. For example the analysis of land-use satellite imagery depends heavily on the use of color. This problem would be significantly more difficult to handle by using monochrome images alone.

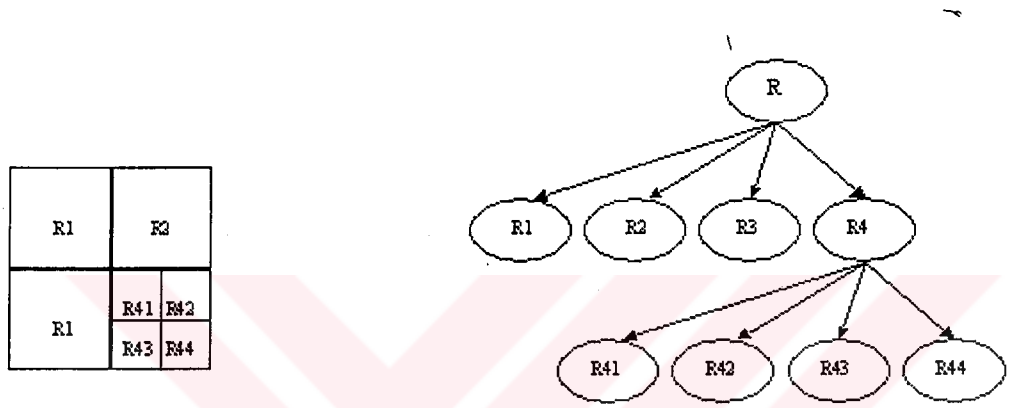
Another problem in region growing is the formulation of a stopping rule. Basically, growing a region should stop when no more pixels satisfy the criteria for inclusion in that region. Criteria such as intensity, texture, and color, are local in nature and do not take into account the "history" of region growth. Additional criteria that increase the power of a region growing algorithm utilize the concept of size, likeness between a candidate pixel and the pixel grown so far (such as a comparison of intensity of a candidate and the average intensity of the region) , and the shape of region being grown.

1.4.3 Region Splitting and Merging

The procedure just discussed grows regions from a set of seed points. An alternative is to subdivide an image internally into a set of arbitrary, disjointed regions and then merge and/or split the regions in an attempt to satisfy the conditions stated in section 1.4.1. A split and merge algorithm that iteratively works toward satisfying these constraints is as follows.

Let R represent the entire image region and select a predicate P. For a square image, one approach for segmenting R is to subdivide it successively into smaller and smaller quadrant regions so that, for any region R_i, $P(R_i) = TRUE$. That is, if

P is false for any quadrant, subdivide that quadrant into subquadrant an so on. This particular splitting technique has a convenient representation in the form of a so called *quadtree* (that is, a tree in which each node has exactly four descendants). as illustrated in Figure 1.3 (b). Note that the root of the tree corresponds to entire image and that each node corresponds to a subdivision. In this case, only R_4 has subdivided further.



a) Partitioned image b) Corresponding quadtree

Figure1.3 Partitioned and corresponding quadtree

If only splitting were used, the final portion likely would contain adjacent regions with identical properties. This drawback may be remedied by allowing merging as well as splitting. Satisfying the constraint of section 1.4.1 requires merging only adjacent regions whose combined pixels satisfy the predicate P; that is two adjacent regions R_j and R_k are merged only if $P(R_j \cup R_k) = TRUE$.

The preceding discussion may be summarized by the following procedure in which, at any step , we:

1) Split into four disjointed quadrants any region R_i where $P(R_i) = FALSE$;
2) Merge any adjacent regions R_j and R_k for which $P(R_i \cup R_k) = TRUE$; and
3) Stop when no further merging or splitting is possible.

Several variations of this basic approach are possible. For example, one possibility is to split the image initially into a set of square blocks. Further splitting is carried out as described, but merging is initially limited to groups of four blocks that are descendants in the quadtree representation and that satisfy the predicate P. When no further mergings of this type are possible, the procedure is terminated by one final merging of regions satisfying step 2. At this point, the merged regions may be of different sizes. The principal advantage of this approach is that it uses the same quadtree for splitting and merging, until the final merging step.

CHAPTER TWO
NEURAL NETWORKS

Mankind has always been interested in how the human brain actually works. From the ancient Greek philosophers to contemporary researchers, many attempts have been made to make a model of the human brain [13]. However, it is not yet impossible. The human brain is a very powerful instrument for all kinds of tasks. from vision and hearing to steering the muscles. Probably the most interesting feature is its capability to learn and to anticipate new tasks. Therefore, by modeling the brain using some artificial model, for example by a computer simulation, one might obtain this same functionality. In our research we do not aim at understanding the working of the human brain, but instead take advantage of the models which simulate the brain's functionality and apply them in solutions. We are interested in models which are able to learn and even generalize, i.e. able to cope with new environments. These models which are artificial constructs modeled in a computer environment are referred to as artificial neural network models.

In order to understand the development of artificial neural networks as used in current research, one has to have some understanding of how the biological brain works. This is necessary since the concepts of artificial neural nets are derived from biological research. The (human) brain consists of basic elements called *neurons*. Each of these neurons consists of a cell body (*soma*), dendrites, synapses and axons. Although in the brain many specialized neurons exist, only the basics will be explained here. The neuron, (see Figure 2.1) receives input signals through dendrites and synapses. These inputs come from the nerves or other neurons. Such signals arrive at a synapse which determines to what extent the signals are transmitted to the dendrites. Thus the signal is amplified by the synapse in some way. This may result in excitation, a positive contribution to the dendrite's signal, or inhibition, a negative

contribution. The dendrites transport the signals to the soma where they are accumulated in what is known as an action potential. When this action potential reaches a certain threshold an output spike is produced. This spike lasts about 1 millisecond and is transported along the axon to other neurons.

The brain consists of millions of these neurons, which may be specialized to some task or not. A powerful property of the brain is its ability to learn. This is assumed to be done by modifying the synaptic connections by growing new dendrites and lengthening the axon [4]. By modifying the synaptic connections the brain is able to learn new tasks or to cope with a loss of function elsewhere in the brain. This behavior of the brain inspired McCulloch and Pitts [25] in the early 1940s to devise an artificial neuron called the *perceptron*. This perceptron is the basis of all neural network models which have been developed. It is a simplified model of the biological neuron. Figure 2.2 shows the perceptron. .

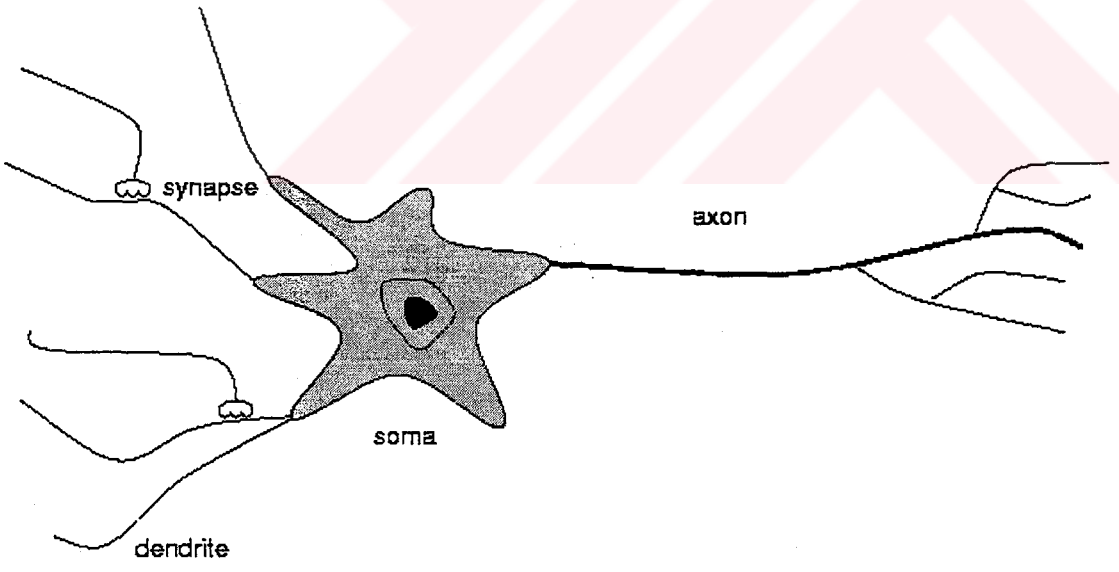


Figure 2.1 The basic features of a biological neuron.

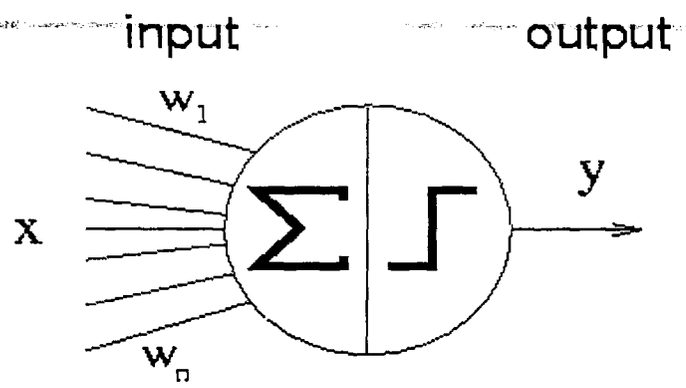


Figure 2.2 Outline of the basic model.

The dendrites and synapses are replaced by input lines that receive inputs from other neurons or the real world. With each of these lines a weight is associated which acts like a dendrite and a synapse. Input signals are multiplied by the weights. The soma is modeled by a body consisting of a summation followed by a thresholding function. This thresholding function is a step function, although in modern neural networks sigmoidal functions are usually used. Whenever the sum of input signals reaches the threshold, an output signal is given on the output line which is connected to other units.

Learning is usually done by looking at examples and extracting some general rules from them, or by using a teacher. The teacher gives examples and the brain receives rewards for correct behavior and penalties for faulty behavior according to the teacher's decision. For instance when children are taught to learn certain actions this is usually done by the mother/father who corrects the child for his/her action. This can also be simulated in the perceptron. Remember that the synapses have been replaced by weights, therefore by changing the weights the behavior of the perceptron changes.

Changing the weights according to some rule, *the learning rule*, using a set of examples is called learning. In general two types of example sets can be distinguished: those which have for each example a desired label and those which do not. These labels indicate what type of example one is dealing with, e.g. the label ``apple" indicates the example is an apple. With these labels, numbers are associated

in order to be able to train a network. These numbers are referred to as *targets*. Learning rules that use examples which have a corresponding label are called *supervised*. That is a teacher indicates whether the answer of a network (*output*) corresponds to the label of an example which was expected. Using only examples and no answers is referred to as *unsupervised learning*. Learning in the perceptron is done by adapting the weights in a supervised way such that the examples supplied to the perceptron produce the correct output. Consider a given set of examples. Each of the examples is labeled either $\mathcal{A}$ or $\mathcal{B}$. A perceptron should produce a 0 when an example with label A is supplied or a 1 when an example with label B is supplied. If a perceptron produces the incorrect output, e.g. a 0 ($\equiv \mathcal{A}$) when a 1 ($\equiv \mathcal{B}$) was expected, the output value of the perceptron is subtracted from the weight values (see Figure 2.2). In the case of a correct prediction the output value is added to the weight values [41]. By using an adaptive term when adding or subtracting only a part of the perceptron's output value is used for an update such that the next time the same example is applied the output will be correct. This type of rule is known as the Widrow-Hoff delta rule [41].

Using this basic building block, networks can be constructed. These constructions are usually referred to as artificial neural networks, or neural networks for short. The network structures which will be dealt with in this thesis are the so-called feed forward neural networks. This means that the input to the network is processed in a layered manner. Basically this allows a black-box approach. These networks consist of an input layer, i.e. the part of the network which "sees" the input from the real world, hidden layers and an output layer, i.e. the part of the network that produces output to the real world. The hidden layers are called hidden since they have no connection with input or output, i.e. they constitute the black-box and have no connection to the real world. These networks can quite easily be built using perceptrons. However, in the 1960s Minsky and Papert [26] showed in their famous book *Perceptrons* that single perceptrons have a fundamental restriction. They showed that a single perceptron cannot solve the XOR problem since it can only induce linear functions. Multi-layer structures, however, are able to solve XOR-like

problems but no adequate learning rules were available at that time. Minsky and Papert's book caused a major decrease of interest in artificial neural networks.

The interest increased significantly when the backpropagation rule was proposed by Werbos [40] and later on rediscovered by Rumelhart, Hinton and Williams [34]. This caused a major breakthrough since it enabled the training of the multi-layered structures. The backpropagation rule derived its name from the fact that the errors made by the network, deviations from the desired output, are propagated back through the network. This implies that the rule uses supervision since a teacher is needed to calculate the error. The weights of the hidden layers are adapted according to the error made by the network.

2.1 NEURAL NETWORK LEARNING RULE

The most basic building block in the neural network is the perceptron. It is characterized by several input lines with weights associated to them. The input is collected and summed and an output is given according to the output function f. The output is formally defined as follows:

$$o = f\left(\sum_{i=1}^{n} w_i \, \overline{x_i}\right) \tag{2.1}$$

where o is the output function and w_i the weight associated with input line i. The McCulloch-Pitts neuron has a step function as output function (see Figure 2.2). This neuron can be trained using the Widrow-Hoff delta rule [41]. Assume the feature values of the samples are real numbers, and their labels are either 0 or 1 denoting two different classes. The delta rule adapts the weights w_i according to the following formula:

$$w_i(t+1) = w_i(t) + \eta(y_i(t) - o(t))\overline{x_i}(t), \tag{2.2}$$

where the $y_i(t)$ is desired response, i.e. the real class label, and η an adjustable parameter called the *learning rate* which usually has a value between 0 and 1. The t is used to indicate the time step. Note that there is no change in the weights if the perceptron produces a correct answer, $y(t) - o(t) = 0$, in other cases the weights are

adapted. In fact the limitation of this perceptron is that, it cannot solve any problem that is linearly inseparable [4] . This makes it unsuited for solving the XOR problem, which need a nonlinear solution. It was theoretically shown that it can be solved only by constructing a layered architecture [26].

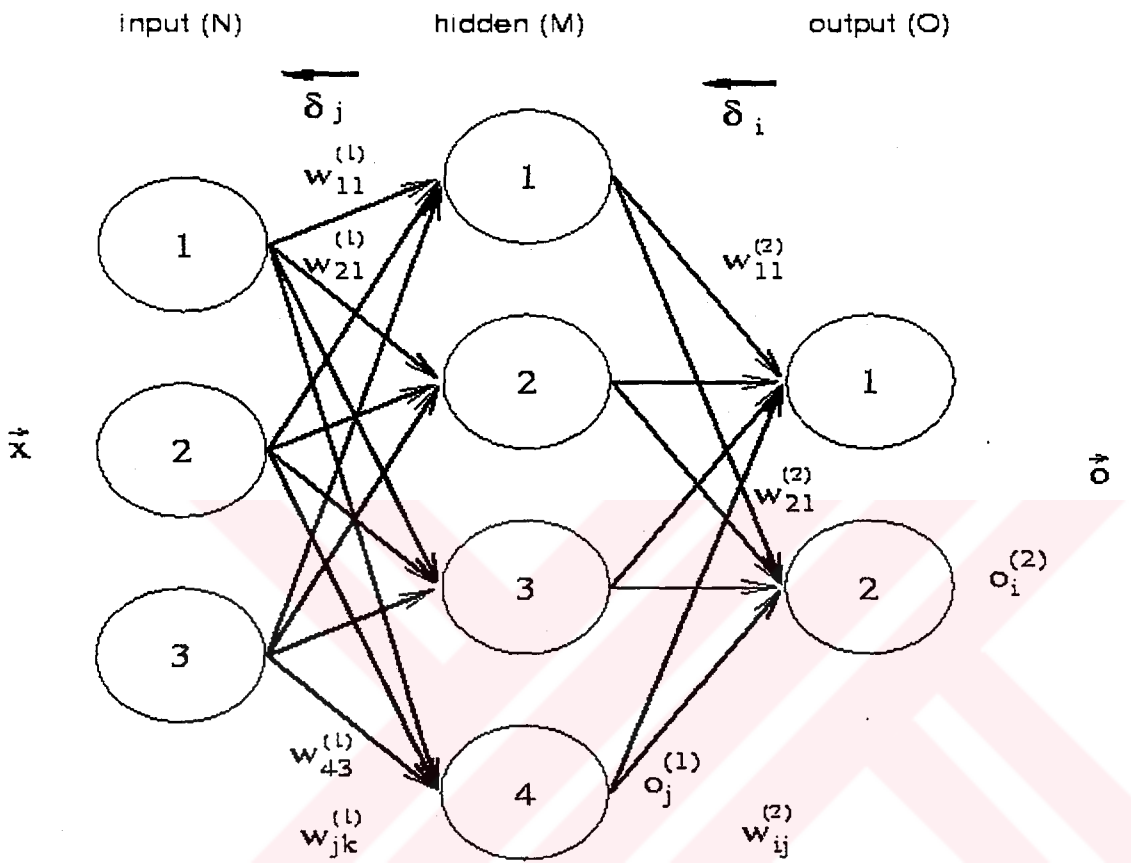


Figure 2.3 A layered structure of perceptrons.

This new model has three layers; an input layer, an output layer, and a layer in between, not connected directly to the input or the output, and so called the hidden layer. Each unit in the hidden layer and the output layer is like a perceptron, except that the thresholding function is the one shown in Figure 2.4, sigmoid function and not the step function. The units in the input layer serve to distribute the values they receive to the next layer, and so do not perform a weighted sum or threshold. Because we modified the single-layer perceptron by changing the nonlinearity from a step function to a sigmoid function, and added a hidden layer, we are forced to alter our learning rule as well.

The learning rule for multilayer perceptrons is called "generalized delta rule", or the "backpropogation rule", and was suggested in 1986 by Rumelhart, and McClelland, and Williams. The operation of the network is similar to that of the single-layer perceptron, in that we show the net a pattern and calculate its response. Comparison with the desired response enables the weights to be altered so that the network can produce a more accurate output next time. As we saw earlier in this chapter, the learning rule used in the single-layer perceptron will not work for multilayer networks. However, the use of the sigmoid function means that enough information about the output is available to the units in earlier layers, so that these units can have their weights adjusted so as to decrease the error next time.

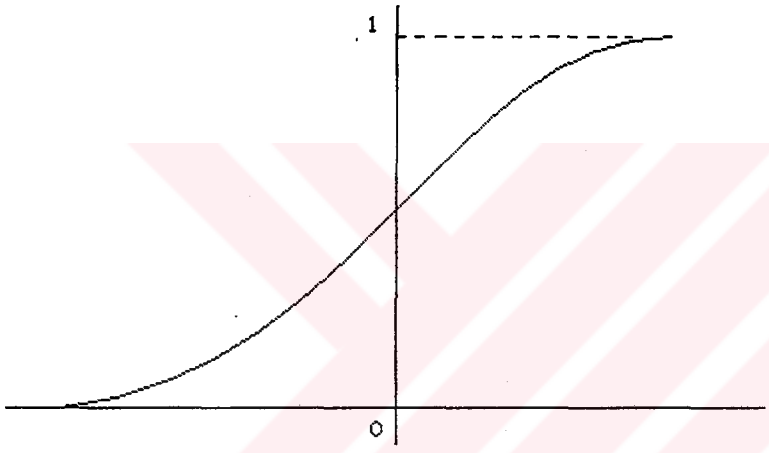


Figure 2.4 Sigmoidal threshold function

The learning rule is a little more complex than the previous one, however, and we can best understand it by considering how the net behaves as patterns are taught to it. When we show the untrained network an input pattern, it will produce any random output. We need to define an error function that represents the difference between the network's current output and the correct output that we want it to produce. Because we need to know the "correct" pattern, this type of learning is known as "supervised learning". In order to learn successfully we want to make the output of the net approach to the desired output, that is we want to continually reduce the value of this error function. This is achieved by adjusting the weights on the links between the units, and the generalized delta rule does this by calculating the value of the error

function for that particular input. and then back-propagating (hence the name!) the error from one layer to the previous one.

Each unit in the net has its weights adjusted so that it reduces the value of the error function; for the units actually on the output, their output and the desired output is known, so adjusting the weights is relatively simple, but for the units in the middle layer, the adjustment is not so obvious. Intuitively, we might guess that the hidden units that are connected to outputs with a large error should have their weights adjusted a lot, while units that feed almost correct outputs should not be altered much. In fact mathematics shows that the weights for a particular node should be adjusted in direct proportion to the error in the units to which it is connected.

2.2 THE MATHEMATICS

Let us define the error function to be proportional to the square of the difference between the actual and desired output, for all the patterns to be learnt.

$$E_p = \frac{1}{2}\sum_j (t_{pj} - o_{pj})^2 \tag{2.3}$$

Where E_P is the error function for pattern p, t_{pj} represents the target output for pattern p on node j, whilst o_{pj} represents the actual output at that node. w_{ij} is the weight from node i to node j.

The activation of each unit j, for pattern p, can be written as

$$net_{pj} = \sum_i w_{ij} o_{pi} \tag{2.4}$$

i.e. simply the weighted sum, as in the single layer perceptron.

The output from each unit j is the threshold function f_j acting on the weighted sum. In the perceptron, it is usually the sigmoid function, although any continuously differentiable monotonic function can be used.

$$o_{pj} = f(net_{pj}) \tag{2.5}$$

we can write

$$\frac{\partial E_p}{\partial w_{ij}} = \frac{\partial E_p}{\partial net_{pj}} \frac{\partial net_{pj}}{\partial w_{ij}} \tag{2.6}$$

looking at the second term in (2.6). and substituting in (2.5)

$$\frac{\partial net_{pj}}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \sum_k w_{kj} o_{pk}$$

$$= \sum_k \frac{\partial w_{jk}}{\partial w_{ij}} o_{pk}$$

$$= o_{pi} \tag{2.7}$$

since $\frac{\partial w_{ki}}{\partial w_{ij}} = 0$ except when $k = i$ when it equals 1.

We can define the change in error as a function of the change in the net inputs to a unit as

$$-\frac{\partial E_p}{\partial net_{pj}} = \delta_{pj} \tag{2.8}$$

and so (2.6) becomes

$$\frac{\partial E_p}{\partial w_{ij}} = \delta_{pj} o_{pi} \tag{2.9}$$

Decreasing the value of E_p therefore means making the weight changes proportional to $\delta_{pj} o_{pi}$,

i.e.

$$\Delta_p w_{ij} = \eta \delta_{pj} o_{pi} \tag{2.10}$$

we now need to know what δ_{pj} is for each of the units, if we know this, we can decrease E. Using (2.8) and chain rule, we can write

$$\delta_{pj} = -\frac{\partial E_p}{\partial net_{pj}} = -\frac{\partial E_p}{\partial o_{pj}} \frac{\partial o_{pj}}{\partial net_{pj}} \tag{2.11}$$

consider the second term, and from (2.5),

$$\frac{\partial o_{pj}}{\partial net_{pj}} = f'(net_{pj}) \tag{2.12}$$

consider now the first term in Equation 2.11. From Equation 2.3, we can differentiate E_p with respect to o_{pj}, giving

$$\frac{\partial E_p}{\partial o_{pj}} = -(t_{pj} - o_{pj}) \tag{2.13}$$

Thus

$$\delta_{pj} = f'(net_{pj})(t_{pj} - o_{pj}) \tag{2.14}$$

This is useful for the output units. since the target and output are both available, but not for the hidden units, since their targets are not known.

So, if unit j is not an output unit, we can write, by the chain rule again, that

$$\frac{\partial E_p}{\partial o_{pj}} = \sum_k \frac{\partial E_p}{\partial net_{pk}} \frac{\partial net_{pk}}{\partial o_{pj}}$$

$$= \sum_k \frac{\partial E_p}{\partial net_{pk}} \frac{\partial}{\partial o_{pj}} \sum_i w_{ik} o_{pi}$$

$$= -\sum_k \delta_{pk} w_{jk} \tag{2.15}$$

Using Equation 2.4 and Equation 2.8, and noticing that the sum drops out since the partial differential is non-zero for only one value, just as in Equation 2.7. Substituting Equation 2.15 in Equation 2.11, we get finally

$$\delta_{pj} = f'(net_{pj}) \sum_k \delta_{pk} w_{jk} \tag{2.16}$$

This equation represents the change in the error function, with respect to weights in the network. This provides a method for changing the error function so as to be sure of reducing it.

The function is proportional to the errors in the subsequent units, so the error has to be calculated in the output units given by Equation 2.14 and then passed back through the net to the earlier units to allow them to alter their connection weights. Equation 2.14 and Equation 2.16 together define how we can train our multilayer networks.

One advantage of using sigmoid function as the non-linear threshold function is that it is quite like step function, and so should demonstrate behavior of a similar nature. The sigmoid function is defined as

$$f(net) = 1/(1 + e^{-k.net}) \tag{2.17}$$

and has the range $0 < f(net) < 1$. k is a positive constant that controls the "spread" of the function: large values of k squash the function until as $k \longrightarrow \infty$, $f(net) \longrightarrow$ Heaviside function. It also acts as an automatic gain control, since for small input signals the slope is quite steep and so the function is changing quite rapidly, producing a large gain. For large inputs the slope and thus the gain is much less. This means that the network can accept large inputs and still remain sensitive to small changes.

A major reason for its use is that it has a simple derivative, however, and this makes the implementation of backpropogation system much easier. Given that the output of a unit, o_{pj} is given by

$$o_{pj} = f(net) = 1/(1+e^{-k.net}) \tag{2.18}$$

its derivative with respect to that unit, $f'(net)$ is given by

$$\begin{aligned} f'(net) &= ke^{-k.net}/(1+e^{-k.net})^2 \\ &= kf(net)(1-f(net)) \\ &= ko_{pj}(1-o_{pj}) \end{aligned} \tag{2.19}$$

the derivative is therefore a simple function of the outputs.

2.3 MULTILAYER PERCEPTRON ALGORITHM

The algorithm for the multilayer perceptron that implements the back-propagation training rule is shown below.

1. Initialize weights and thresholds.

   Set all weights and thresholds to small random values

2. Present input and desired output

   Present input $x_p = x_0, x_1, x_2, ..., x_{n-1}$ and target output $T_p = t_0, t_1, ..., t_{m-1}$ where n is the number of input nodes. Set w_0 to be $-\Theta$, the bias, and x_0 to be always 1.

3. Calculate actual output

   Each layer calculates

$$y_{pj} = f\left[\sum_{i=0}^{n-1} w_{ij} x_i\right]$$

and passes that as input to the next layer. The final layer outputs values o_{pj}.

4. Adapt weights.

Start from the output layer, and work backwards.

$$w_{ij}(t+1) = w_{ij}(t) + \eta \delta_{pj} o_{pj}$$

$w_{ij}(t)$ represents the weights from node i to node j at time t, η is a gain term, and δ_{pj} is an error term for pattern p on node j.

For output units

$$\delta_{pj} = ko_{pj}(1 - o_{pj})(t_{pj} - o_{pj})$$

For hidden units

$$\delta_{pj} = ko_{pj}(1 - o_{pj})\sum_k \delta_{pk} w_{jk}$$

where the sum is over the k nodes in the layer above node j.

2.4 VISUALIZING NETWORK BEHAVIOUR

This learning rule is not guarantied to produce convergence, however, and it is possible for the network to fall into a situation in which it is unable to learn the correct output. In order to understand this, we need a method of visualizing what is going on in the network. As we have seen, the network computes an error or energy function,

$$E_p = \frac{1}{2}\sum_j (t_{pj} - o_{pj})^2$$

Which represents the amount of the net differs from the required output. Since the output of net is related to the weights between the units and the input applied, the energy is therefore a function of the weights and inputs to the network. We can draw a graph of the energy function showing how varying the weights affects the energy, for a fixed input pattern.

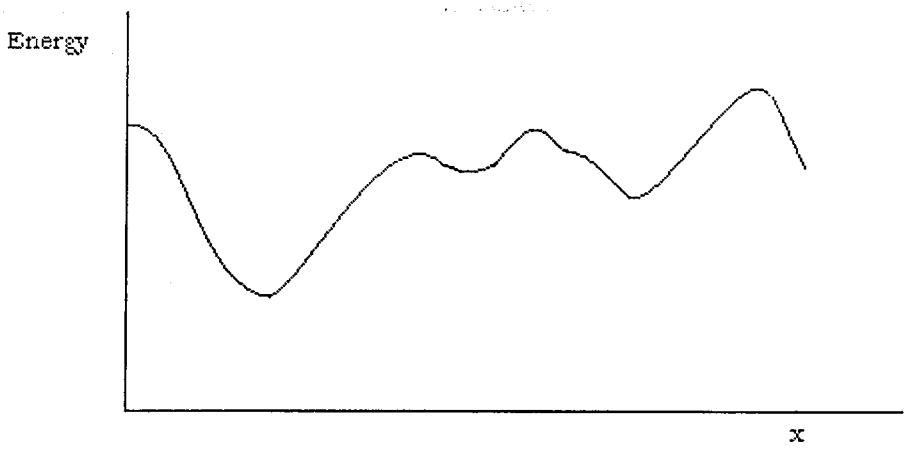


Figure 2.5 The energy function in one dimensional, as we vary one weight, for a fixed pattern.

The generalized delta rule aims to minimize the error function E by adjusting the weights in the network so that they correspond to those at which the energy surface is lowest. It does this by a method known as gradient descent, where the energy function is calculated, and energy changes are made in the steepest downward direction. This is guaranteed to find a solution in cases where the energy landscape is simple. Each possible solution is represented as a hallow, or a basin, in the landscape. These "basin of attraction", as they are known represent the solutions to the values of the weights that produce the correct output from a given input. Remember that these basins are actually many dimensional.

2.5 GENERALIZATION

One of the major features of neural networks is their ability to generalize, that is, to successfully classify patterns that have not been previously presented. Multilayer perceptrons generalize by detecting features of input pattern that have been learnt to be significant, and so coded into the internal units. Thus an unknown pattern is classified with others that share the same distinguishing features. This means that learning by example is a feasible proposition, since only a representative set of input have to be taught to the network, and the generalization properties will allow similar inputs to be classified as well.

2.6 FAULT TOLERANCE

Multilayer perceptron networks are intrinsically fault-tolerant, since they are distributed parallel processing elements, with each node contributing to the final output response. If a node or its weights are lost or damaged, recall is impaired in quality, but the distributed nature of the information means that damage has to be extensive before a networks response degrades badly. The network therefore demonstrates degradation in performance rather than catastrophic failure.

2.7 LEARNING DIFFICULTIES

Occasionally the network settles into a stable solution that does not provide the correct output. In these cases, the energy function is in a "local minimum". This means that in every direction in which we could move in the energy landscape, the energy is higher than at the current position. It may be that there is only a slight "lip" to cross before reaching an actual deeper minimum, but the network has no way of knowing this, since learning is accomplished by following the energy function down in the steepest direction, until it reaches the bottom of a well, at which point there is no direction to move in order to reduce the energy.

There are alternatives to minimizing these occurrences, which are outlined below.

1) **Lowering the gain term:** If the rate at which the weights are altered is progressively decreased, then the gradient descent algorithm is able to achieve a better solution. If the gain term η is made large to begin with, large steps are taken across the weight and energy space towards the solution. As the gain is decreased, the network weights settle into a minimum energy configuration without overshooting the stable position, as the gradient descent takes smaller downhill steps. This approach enables the network to bypass local minima at first, then hopefully locate, and settle in, some deeper minima without oscillating wildly. However, the reduction in the gain term will mean that the network will take longer to converge.

2) **Addition of internal nodes:** Local minima can be considered to occur when two or more disjoint classes are categorized as the same. This amounts to a poor internal representation within the hidden units, and so adding more units to this layer will allow a better recording of the inputs and lessen the occurrence of these minima.

3) **Momentum term**: The weight changes can be given some "momentum" by introducing an extra term into weight adaptation equation that will produce a larger change in the weight if the changes are currently large, and will decrease as the changes become less. This means that the network is less likely to get stuck in local minima early on, since the momentum term will push changes in local increases in the energy function, following the overall trend.

   The momentum term can be written as follows.

$$\delta_p w_{ji}(t+1) = w_{ji}(t) + \alpha(w_{ji}(t) - w_{ji}(t-1)) \tag{2.20}$$

   where α is the momentum factor, $0 < \alpha < 1$.

CHAPTER THREE
DIGITAL IMAGE COMPRESSION TECHNIQUES

In this chapter we will study digital image compression techniques and application of artificial neural networks to digital image compression. Applications dealing with image identification do not require high quality. Human beings can identify images very effectively, especially in noisy circumstances. Some information in these images can be removed while maintaining sufficient quality of identification. This is an application of still image data compression.

3.1 NEED FOR DATA COMPRESSION

The Table 3.1 shows the qualitative transition from simple text to full-motion video data and the disk space, transmission bandwidth, and transmission time needed to store and transmit such uncompressed data. The examples below clearly illustrate the need for sufficient storage space, large transmission bandwidth, and long transmission time for image, audio, and video data. At the present state of technology, the only solution is to compress multimedia data before its storage and transmission, and decompress it at the receiver for play back. For example, with a compression ratio of 15:1, the space, bandwidth, and transmission time requirements can be reduced by a factor of 15, with acceptable quality.

Table 3.1. Multimedia data types and uncompressed storage space, transmission bandwidth, and transmission time required. The prefix kilo- denotes a factor of 1000 rather than 1024.

Multimedia Data	**Size/ Duration**	**Bits/Pixel or Bits/Sample**	**Uncompressed Size (B for bytes)**	**Transmission Bandwidth (b for bits)**	**Transmission Time (using a 28.8K Modem)**
A page of text	11" x 8.5"	Varying resolution	4-8 KB	32-64 Kb/page	1.1 - 2.2 sec
Telephone quality speech	10 sec	8 bps	80 KB	64 Kb/sec	22.2 sec
Grayscale Image	512 x 512	8 bpp	262 KB	2.1 Mb/image	1 min 13 sec
Color Image	512 x 512	24 bpp	786 KB	6.29 Mb/image	3 min 39 sec
Medical Image	2048 x 1680	12 bpp	5.16 MB	41.3 Mb/image	23 min 54 sec
Full-motion Video	640 x 480, 1 min (30 frames/sec)	24 bpp	1.66 GB	221 Mb/sec	5 days 8 hrs

3.2 HISTORICAL BACKGROUND ON IMAGE COMPRESSION AND COMPRESSION STANDARDS

Image compression algorithms have been the subject of research both in academia and industry for many years. Today, while significantly improved algorithms have been achieved and compression performance is better than a decade ago, there is still room for new technologies. The first widely adopted international image compression standard was JPEG [17], [30] which was introduced in the late eighties. JPEG is based on DCT followed by entropy coding based on either Huffman coding

[14], [35], [8], or binary arithmetic coding [20], [33], [8], [42]. It has been widely used from the printing industry to internet applications. For example all high-end printers compress the image to be printed before they actually send it to the print engine, and most images transmitted through the internet are JPEG compressed. JPEG is intended for continuous tone images of more than one bit depth. Algorithms for binary images work in a different way, JBIG-1 and JBIG-2 are the standards covering this area. JPEG and JBIG are part of other standards, such as facsimile transmission standards [16], the Flash Pix file format [12], the TIFF file format [1], and page description languages like PDF.

3.3 BANDWIDTH, MEMORY, COMPUTATIONS AND COMPRESSİON

While the usefulness of digital images in communicating information is not questionable, the cost of storing and transmitting images is much larger compared to storage and transmission of text, so that for example image databases require more storage than document archives. While series of still images can form a video sequence, still images are treated in a different way than video images. In video we are dealing with a three dimensional space, with the third dimension being time. Video images need to be transmitted at a certain rate for real time display. The amount of data transmitted through the internet doubles every year, and a large portion of that data is images. Reducing the bandwidth needs of any given device will result in significant cost reductions and will make use of the device more affordable. Magnetic hard discs (HD), CD's, DVD's of larger capacity are released every year, in response to greater demand for storage of digital data. The use of digital cameras, both still and video, is becoming more widely accepted. Image compression offers ways to represent an image in a more compact way, so that one can store more images and transmit images faster. The advantages of image compression come at the expense of numerical computations, and therefore we can trade off computations for storage or bandwidth. Before storing or transmitting an image we process it in such a way that will require fewer bits for its representation. Technologies addressing increase of bandwidth of a given channel, increase of the size of a memory device and increase in the number of computations per second

performed by a CPU. have been developing simultaneously without any sign of saturation. Memory sizes are only limited by the physical size of the device, if we place two discs one next to the other we double the capacity of the system. The same is true for computations, as two processors have twice the computational power of a single processor. However the problem of just duplicating a processor or a memory unit is bandwidth. If we duplicate a processor we need to make sure that the two processors can communicate with each other at high enough rates and we can access all memory units through the same bus. It seems that bandwidth, memory and computations are closely interrelated, and which one is more important depends on the specific application. Compression algorithms can be thought of as means to trade off computation, memory and bandwidth. A compression algorithm tries to offer the best trade off among the three factors, for a given application. For example if we are limited in terms of memory we can spend more computational time to compress the image and make sure it fits into the given memory size. If we are computation limited we can store the image as is with no compression or with limited compression with a simple compression algorithm.

3.4 IMAGE QUALITY METRICS

Image compression and decompression operations are often called image coding operations, because the process uses data coding methods to represent an image in a new form.

The amount of compression is calculated by dividing the data size of the original image by the size of the compressed image. The result is called as compression ratio. The higher the compression ratio, the smaller the compressed image size.

$$C_R = \frac{n_1}{n_2}(pixel / pixel) \tag{3.1}$$

where n_1 is the data size of the original image and n_2 is the data size of the compressed image. The relative redundancy is defined as

$$R_D = 1 - \frac{1}{C_R}(pixel / pixel) \tag{3.2}$$

Another goal of data compression is also to reduce data redundancy in an image. A practical compression ratio such as 8 (or 8:1) means that the input data set has 8 information carrying units for every 1 unit in compressed output data set. The corresponding data redundancy of 0.875 means that 87.5 percent of the data in the input set is redundant.

The quality of an image can be evaluated through a subjective evaluation. An objective evaluation involves a peak signal-to-noise ratio (PSNR), as given by

$$PSNR = 10\log_{10}\left\{\frac{(peak - signal - value)^2}{MSE}\right\} \tag{3.3}$$

The mean-squared error is:

$$MSE = \frac{1}{MxN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}(f(x,y) - f^{'}(x,y))^2 \tag{3.4}$$

where,

$peak - signal - value$ = 255 for an 8 bits/pixel image.

MxN = number of pixels in image.

$f(x,y), f^{'}(x,y)$ = value of pixel (x,y) in the original and reconstructed images.

Notice that the argument of the log function is the highest signal-to-noise ratio possible, since the original signal is assumed to be at maximum brightness throughout, thus the name peak SNR. The PSNR reduces variability between other experimenter's measurements. For example, two pairs of images (each consisting of an original and a decompressed version) may have the same MSE. However, one of the original images may be brighter than the other, and this would give a misleading SNR.

Through out this thesis, results are given for two measures of compression performance: compression ratio and peak signal to noise ratio.

3.5 GENERAL PRINCIPLES BEHIND COMPRESSION

A common characteristic of most images is that the neighboring pixels are correlated and therefore contain redundant information. The foremost task then is to find less correlated representation of the image. Two fundamental components of compression are redundancy and irrelevancy reduction. Redundancy reduction aims at removing duplication from the signal source (image/video). Irrelevancy reduction omits parts of the signal that will not be noticed by the signal receiver, namely the Human Visual System (HVS). In general, three types of redundancy can be identified:

- **Spatial Redundancy** or correlation between neighboring pixel values.
- **Spectral Redundancy** or correlation between different color planes or spectral bands.
- **Temporal Redundancy** or correlation between adjacent frames in a sequence of images (in video applications).

Image compression research aims at reducing the number of bits needed to represent an image by removing the spatial and spectral redundancies as much as possible. Since we will focus only on grayscale still image compression, we will not worry about temporal redundancy and spectral redundancy.

3.6 DIFFERENT CLASSES OF COMPRESSION TECHNIQUES

Two ways of classifying compression techniques are mentioned here.

3.6.1 Lossless vs. Lossy compression:

In lossless compression schemes, the reconstructed image, is numerically identical to the original image after compression. However lossless compression can only achieve a modest amount of compression. An image reconstructed following lossy compression contains degradation relative to the original. Often this is because

the compression scheme completely discards redundant information. However, lossy schemes are capable of achieving much higher compression.

3.6.2 Predictive vs. Transform coding

Typically, images exhibit high degree of correlation among neighboring samples. A high degree of correlation implies a high degree of redundancy in the raw data. Therefore, if the redundancy is removed by decorrelating the data, a more efficient and hence compressed coding of data is possible. This can be accomplished through the use of predictive coding [9]. In predictive coding, information already sent or available is used to predict future values, and the difference is coded. Since this is done in the image or spatial domain, it is relatively simple to implement and is readily adapted to local image characteristics.

Another approach to image compression is the use of transformations that operate on an image to produce a set of coefficients. A subset of these coefficients is chosen and quantized for transmission across a channel or for storage. The goal of this technique is to choose a transformation for which such a subset of coefficients is adequate to reconstruct an image with a minimum of discernible distortion. This method provides greater data compression compared to predictive methods, although at the expense of greater computation.

3.6.3 A Typical Image Coder

A typical lossy image compression system is shown in Figure 3.1. It consists of three closely connected components namely (a) Source Encoder (b) Quantizer, and (c) Entropy Encoder. Compression is accomplished by applying a linear transform to decorrelate the image data, quantizing the resulting transform coefficients, and entropy coding the quantized values.

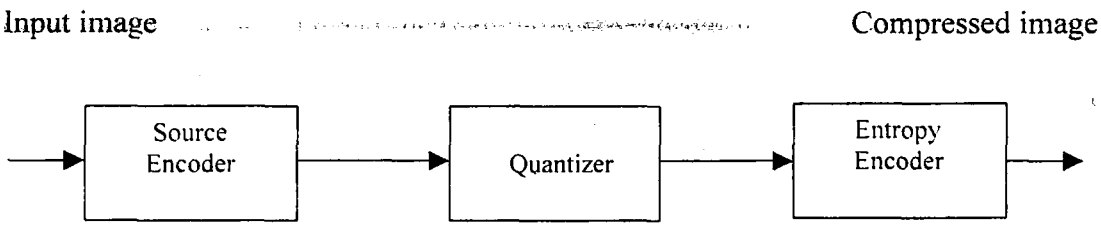


Figure 3.1 A Typical Lossy Signal/Image Encoder

3.6.4 Source Encoder (or Linear Transformer)

Over the years, a variety of linear transforms have been developed which include Discrete Fourier Transform (DFT), Discrete Cosine Transform (DCT) [2], Discrete Wavelet Transform (DWT) [6], [37], and many more, each with its own advantages and disadvantages.

3.6.5 Quantizer

A quantizer simply reduces the number of bits needed to store the transformed coefficients by reducing the precision of those values. Since this is a many-to-one mapping, it is a lossy process and is the main source of compression in an encoder. Quantization can be performed on each individual coefficient, which is known as Scalar Quantization (SQ). Quantization can also be performed on a group of coefficients together, and this is known as Vector Quantization (VQ). Both uniform and non-uniform quantizers can be used depending on the problem at hand.

3.6.6 Entropy Encoder

An entropy encoder further compresses the quantized values losslessly to give better overall compression. It uses a model to accurately determine the probabilities for each quantized value and produces an appropriate code based on these probabilities so that the resultant output code stream will be smaller than the input stream. The most commonly used entropy encoders are the Huffman encoder and the Arithmetic encoder, although for applications requiring fast execution, simple run-length encoding (RLE) has proven very effective.

It is important to note that a properly designed quantizer and entropy encoder are absolutely necessary along with optimum signal transformation to get the best possible compression.

3.7 DISCRETE COSINE TRANSFORM CODING

The discovery of DCT in 1974 is an important achievement for the research community working on image compression [2], [19]. The DCT can be regarded as a discrete-time version of the Fourier-Cosine series. It is a close relative of DFT, a technique for converting a signal into elementary frequency components. Thus DCT can be computed with a Fast Fourier Transform (FFT) like algorithm. Unlike DFT, DCT is real valued and provides a better approximation of a signal with fewer coefficients. The DCT of a discrete signal $f(x)$; $x=0, 1, .., N-1$ is defined as:

$$C(u) = \sqrt{\frac{2}{N}}\alpha(u)\sum_{n=0}^{N-1} f(x)\cos\left[\frac{(2x+1)u\pi}{2N}\right] \tag{3.7}$$

for u=0,1,2....,N-1

where, $\alpha(u) = 0.707$ for $u = 0$ and

$= 1$ otherwise.

The 2-D DCT pair is defined as [27]

$$C(u,v) = \sqrt{\frac{2}{N}}\alpha(u)\alpha(v)\sum_{y=0}^{N-1}\sum_{x=0}^{N-1} f(x,y)\cos\left[\frac{(2x+1)u\pi}{2N}\right]\cos\left[\frac{(2y+1)v\pi}{2N}\right] \tag{3.8}$$

for u, v=0, 1, 2,...,N-1, $\alpha(0) = 0.707$; $\alpha(u) = 1$; for u=1. 2, 3,...,N-1.

$$f(x,y) = \sqrt{\frac{2}{N}}\alpha(u)\alpha(v)\sum_{v=0}^{N-1}\sum_{u=0}^{N-1} C(u,v)\cos\left[\frac{(2u+1)x\pi}{2N}\right]\cos\left[\frac{(2v+1)y\pi}{2N}\right] \tag{3.9}$$

For x, y=0, 1, 2,...,N-1,

where, $\alpha(0) = 0.707$; $\alpha(u) = 1$; for u=1, 2, 3,...,N-1.

$\alpha(v) = 1$; for v=1, 2, 3,...,N-1.

DCT–Based Compression Block Diagram is shown in Figure 3.2. In the DCT coding, first image pixels are grouped into 8x8 blocks and then Equation 3.8 is applied to each block. Because adjacent image pixels are highly correlated, the DCT

processing step lays the foundation for achieving data compression by concentrating most of the signal in the lower spatial frequencies. The DCT is performed on an NxN square matrix of pixel values. and yields an NxN square matrix of frequency coefficients. In this new matrix. the coefficients in row and columns 0. known as DC components carry more notable information about the image than the other coefficients. The DC coefficient at position (0.0) in output matrix represents an average of the overall magnitude of the input matrix and as the elements are farther from this coefficient, they tend to become lower in magnitude. It means that by performing the DCT on the input data, the representation of the image has been concentrated in the upper left coefficients of the calculated matrix, with the coefficients containing less useful information in the lower right of the same matrix. These less important components can be ignored without seriously damaging the quality of the picture.

As a result of calculation of 64 DCT coefficients for each block, the coefficients are arranged in order by Zig-Zag order rule (see Figure 3.4). The coefficients which are not used in compression are set to zero. The decompressed image is obtained by using inverse Discrete Cosine Transform given in Equation 3.9. In principle, the DCT introduces no loss to the source image samples; it merely transforms them to a domain in which they can be more efficiently encoded.

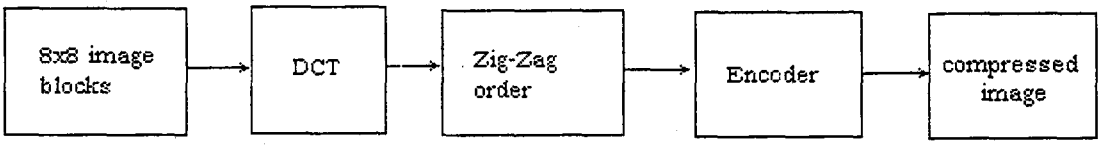


Figure 3.2 DCT –Based Compression Block Diagram

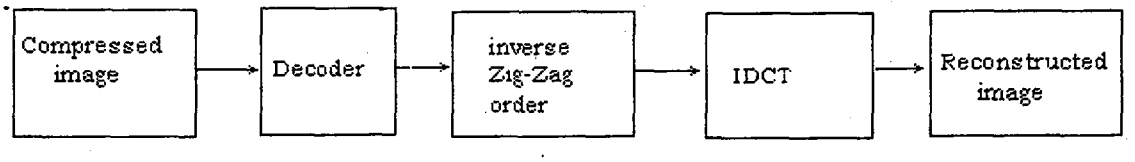


Figure 3.3 DCT-Based Decompression Block Diagram

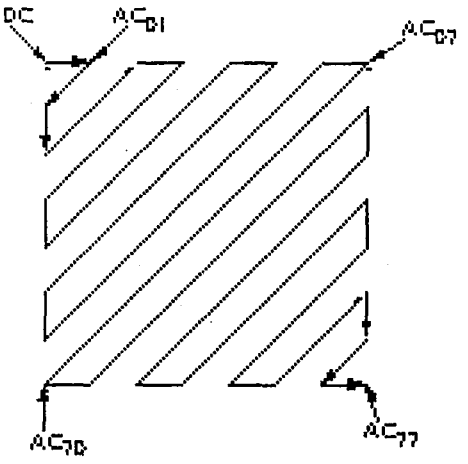


Figure 3.4 Zig-Zag sequence

3.8 JPEG : DCT-Based Image Coding Standard

In 1992, JPEG established the first international standard for still image compression where the encoders and decoders are DCT-based. The JPEG standard specifies three modes namely sequential, progressive, and hierarchical for lossy encoding, and one mode of lossless encoding. The baseline JPEG coder [39], [30], which is the sequential encoding in its simplest form, will be briefly discussed here. Figure 3.5 (a) and 3.5 (b) show the key processing steps in such an encoder and decoder for grayscale images. Color image compression can be approximately regarded as compression of multiple grayscale images, which are either compressed entirely one at a time, or are compressed by alternately interleaving 8x8 sample blocks from each in turn. In this thesis, we focus on grayscale images only.

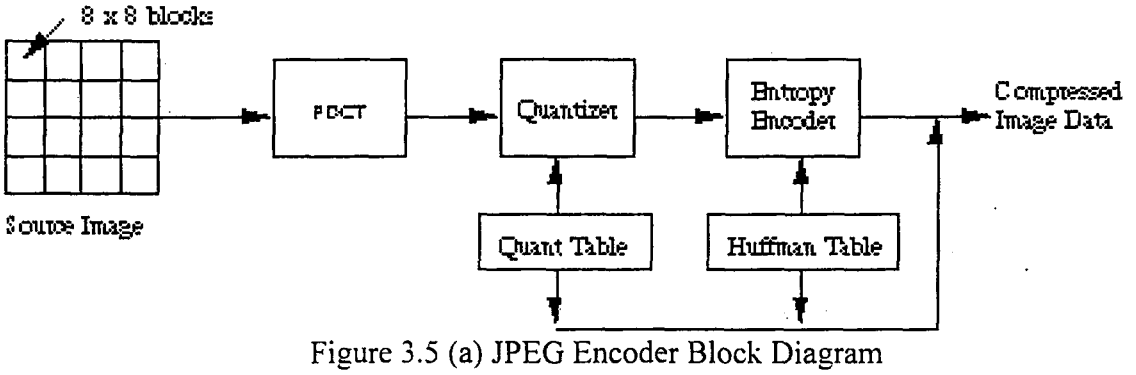


Figure 3.5 (a) JPEG Encoder Block Diagram

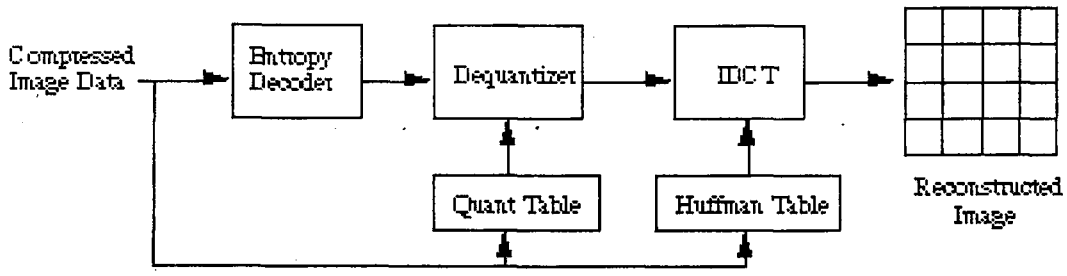


Figure 3.5 (b) JPEG Decoder Block Diagram

After output from the FDCT, each of the 64 DCT coefficients are uniformly quantized in conjunction with a carefully designed 64-element Quantization Table (QT) [39]. At the decoder, the quantized values are multiplied by the corresponding QT elements to recover the original unquantized values. After quantization, all of the quantized coefficients are ordered into the "zig-zag" sequence as shown in Figure 3.4. This ordering helps to facilitate entropy encoding by placing low-frequency non-zero coefficients before high-frequency coefficients. The DC coefficient, which contains a significant fraction of the total image energy, is differentially encoded.

Entropy Coding (EC) achieves additional compression losslessly by encoding the quantized DCT coefficients more compactly based on their statistical characteristics. The JPEG proposal specifies both Huffman coding and arithmetic coding. The baseline sequential codec uses Huffman coding, but codecs with both methods are specified for all modes of operation. Arithmetic coding, though more complex, normally achieves 5-10% better compression than Huffman coding.

3.9 VECTOR QUANTIZATION

In vector quantization, the image data are transformed by a nonlinear operator which maps image blocks into codeword indices. Each codeword is used as the reconstructed image block for the corresponding index upon decoding.

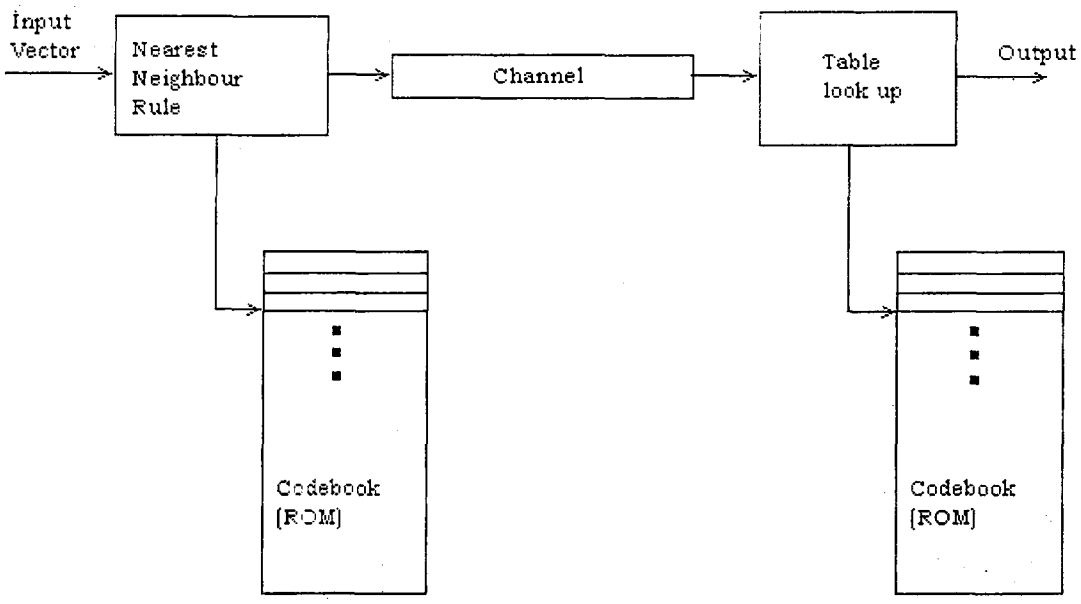


Figure 3.6 Block Diagram of simple VQ

The process of quantization maps a signal $x(n)$ into a series of K discrete messages. For the k th. message, there exists a pair of thresholds t_k and t_{k+1}, and an output value q_k such that $t_k < q_k \leq t_{k+1}$. For a given set of quantization values, the optimal thresholds are equidistant from the values. The concept of quantizing data can be extended from scalar or one-dimensional data to vector data of arbitrary dimension. Instead of output levels, vector quantization (VQ) employs a set of representation vectors (for one dimensional case) or matrices (for the two dimensional case). The set is referred to as the "codebook" and the entries as "codewords". The thresholds are replaced by decision surfaces defined by a distance metric. Typically, Euclidean distance from the codeword is used. The advantage of vector quantization over scalar quantization is that the high degree of correlation between neighboring pixels can be exploited.

In coding phase, the image is subdivided into blocks, typically of a fixed size of nxn pixels. For each block the nearest codebook entry under the distance metric is found and the ordinal number of the entry is transmitted. On reconstruction, the same codebook is used and a simple lookup operation is performed to produce the

reconstructed image. The standard approach to calculate the codebook is the by the way of the Linde, Buzo, Gray (LBG) algorithm [22], [23]. Initially K codebook entries are set to random values. On each iteration, each block in the input space is classified, based on its nearest codeword. Each codeword is then replaced by the mean of its resulting class. The iterations continue until a minimum acceptable error is achieved.

3.9.1 LBG Algorithm

Basically a K-dimensional vector quantizer with M levels, can be viewed as a mapping operation represented by Q which assigns to each input vector, $X = \{x_0, x_1, ..., x_{K-1}\}$, a reproduction vector $\overline{X} = Q(X)$, drawn form a finite reproduction codebook $\overline{A} = \{y_i; i = 1,2,...,M\}$. The quantizer Q is completely described by the codebook $\overline{A}$ together with the partition, $S = \{S_i; i = 1,2,...,M\}$ of the input vector space into the sets $S_i = \{X; Q(X) = y_i\}$ of input vectors mapping into the i th reproduction vector. The vector quantizer will be said to be optimal if it minimizes the expected distortion, that is, Q^* is global optimal if for all other quantizers Q mapping M reproduction vectors we have: $D(Q^*) \le D(Q)$. Linde, Buzo and Gray proposed the following algorithm known as LBG vector quantization to achieve the optimal quantization.

1) Initialization : Given $M =$ number of quantization levels (or number of reproduction vectors), distortion threshold $\varepsilon \ge 0$, an initial M – level reproduction codebook A_0 and a training sequence $\{X_j; j = 0,...,n-1\}$, set m =0 and $D_{-1} = \infty$.
2) Given the reproduction alphabet $A_m = \{W_i; i = 1,2,...,M\}$ find the minimum distortion partition of the training sequence: $X_j \in S_i$ if $d(X_j, W_i) \le d(X_j, W_k)$ for all k.

Compute the average distortion:

$$D_m = D(\{A_m, P(A_m)\}) = \frac{1}{n}\sum_{=0}^{n-1}\{\min d(X_j, W)\} \tag{3.8}$$

3) If $(D_{m-1} - D_m)/D_m \le \varepsilon$, halt with A_m as the final reproduction codebook, otherwise continue.

4) Find the optimal reproduction codebook

$A_{m-1} = X'(P(A_m)) = \{X'(S_i); i = 1,2,...,M\}$ for $P(A_m)$. For the square-error criterion, $X'(S_i)$ is the Euclidean centroid given by;

$$X'(S_i) = \frac{1}{\|S_i\|}\sum_{x_j \in S_i} X_j \tag{3.9}$$

where $\|S_i\|$ denotes the number of training vectors in the cell S_i. If $\|S_i\| = 0$, set $X'(S_i) = W_i$, the old codeword, define $A_{i+1} = X'(P(A_m))$, replace m by $m+1$. and go to step (2) .

This algorithm minimizes the mean squared error over the training set. While the LBG algorithm converges to a local minimum, it is not guaranteed to reach the global minimum. In addition, the algorithm is very sensitive to the initial codebook. Furthermore, the algorithm is slow since it requires an exhaustive search through the entire codebook on each iteration.

With this brief review of conventional image compression techniques at hand, we are ready to consider the role of neural networks as an image compression tool.

3.10 NEURAL NETWORKS AND IMAGE DATA COMPRESSION

"A picture is worth a thousand words". This axiom expresses the essential difference between our ability to perceive linguistic information and visual information. For the same message, a visual representation tends to be perceived as being more efficient than the spoken or written word. This is hardly surprising when viewed from an evolutionary perspective. Language is but an instantaneous development in the course of evolutionary history and is manifested in only a single species. In contrast, vision, in one form or another has existed for hundreds of

millions of years and is shared by a countless number of organisms. But why should there be in such a difference between the two representations in such an advanced being as ourselves?

The processing of language is inherently serial. Words and their meanings are recorded or perceived one at a time in a causal manner. Visual information, on the other hand, is processed by massively parallel interconnected networks of processing units. In the mammalian visual system, this parallelism is evident from the retina right through to the higher-order structures in the visual cortex and beyond. The efficiency of such parallel architectures is reflected by the efficiency with which we process images over language.

There are many tasks which the brain performs well, but which we eluded efficient implementation a serial architecture. It would seem logical that to emulate such tasks at which the brain is very efficient in an artificial system, the architectural characteristics of the artificial system should reflect the corresponding natural system. For image processing applications, this suggests the most efficient network [9].

When selecting a compression technique, it is helpful to consider the type of data that is of interest. Images are inherently imperfect data. That is, minor loses of data (distortion or noise) in images are tolerable to the human eye. For example, it has been established that we can not distinguish brightness levels less than 2 % of 8-bit resolution [21]. Furthermore, our perception plays a major role in our visualization of real world. When we observe images and store them in our memory, we are most likely performing some kind of compression. If we did not do some kind of compression, the amount of data we would have to store in order to recall ordinary events would overflow our memory banks in a very short time.

In many applications, the neural network model may have a number of advantages over the serial model. Because of its parallel architecture, neural networks may break down some of the computational bottlenecks which limit the

performance of serial machines. Since neural networks are trained using example data, they can be made to adapt changes in the input data by allowing the training to continue during the processing of new data. Another advantage of training is that since data are presented individually, no overhead is required to store the entire training set. This is particularly important when very large data sets, of which images an example. The high degree of connectivity can allow neural networks to self-organize, which is an advantage when the structure of data is not known beforehand.

3.10.1 The Use of Multilayer Feedforward ANN's as Image Data Compressor

If an artificial neural feedforward network with a single hidden layer and an equal number of input and output nodes, undergoes supervised training with identical input and output data, then a representation of that data will be formed at the outputs of the hidden layer nodes. This property can be used as a basis of a data compression system [3], [9], [10], [18], [32], [36], [38].

The feedforward artificial neural network is shown in Figure 3.7. The compression is created by having fewer nodes in the hidden layer than in the input and output layers, and so the representation formed on the hidden output layers will have a reduced dimension compared to that at the input and the output. After the network is trained, the reduced dimension representation of the input data has to be transmitted to the receiver together with the weights of the trained output layer. These weights can then be used for reconstruction of the original data set at the receiver.

In effect the hidden layer is separated into two : a compressor formed of the input and the hidden layers and a decompressor formed of hidden and output layers as shown in Figure 3.8. The output of the compressor has exactly the same function as the hidden layer of the trained network, but the input layer of the decompressor is made of input nodes that has no computing function: they merely provide a fan out capability. The transmission links have an effective constant weight of one so the data sent from the compressor is identical to that input to the decompressor.

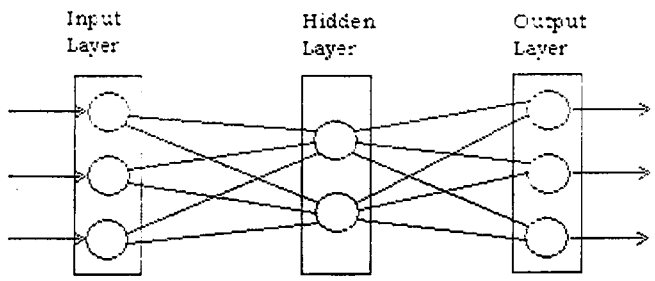


Figure 3.7 Feedforward artificial neural network with a single hidden layer.

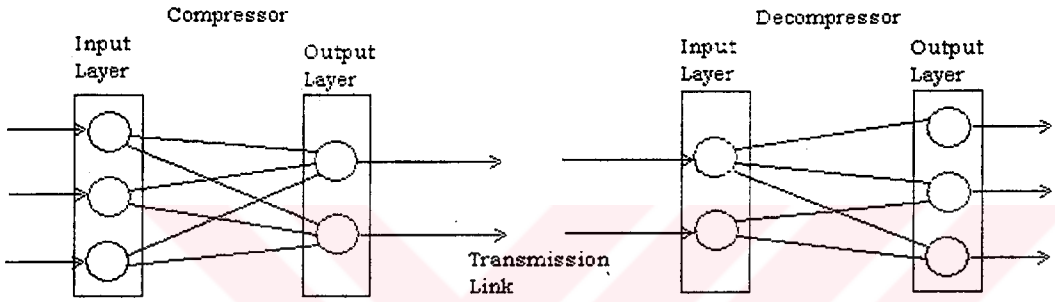


Figure 3.8 Feedforward artificial neural network used as compressed data link.

Since compression and decompression is performed by matrix multiplication of input data and network weights we can classify this method of image compression into transform based techniques.

CHAPTER FOUR
SIMULATIONS AND RESULTS

In this chapter, first DCT based compression is applied to different images and the simulation results are presented then the artificial neural network based approach is tested on some standard images. The performance of neural network based compression is further improved with preprocessing image data by quad-tree based image segmentation method.

4.1 Performance Results for DCT-Based Compression

DCT-based compression method. which is explained in section 3.7. is applied to different kinds of images. The compression ratio is changed by changing the number of DCT coefficients that are used for the compressed image. Table 4.1 shows the compression ratio versus PSNR value for compression of image "Couple". As expected PSNR value is increased by increasing the number of DCT coefficients used in reconstructed image. In Figure 4.1 and Figure 4.2, the reconstructed images "Zelda" are shown for different compression ratios.

Table 4.1 PSNR values for different compression ratios, DCT-based method tested on image "Couple".

Compression ratio	2/64	4/64	6/64	8/64	10/64
PSNR	23.20	25.36	26.85	27.85	28.86

T.C. YÜKSEKÖĞRETİM KURULU
DOKÜMANTASYON MERKEZİ

Original image.

Cr=64/6, PSNR=34.17

Figure 4.1 DCT-Based compression applied to image "Zelda" for different compression ratios

Cr=64/5, PSNR=32.78

Cr=64/2, PSNR=29.23

Figure 4.2 DCT-Based compression applied to image "Zelda" for different compression ratios.

4.2 SIMULATIONS WITH NEURAL NETWORKS

The modified feedforward artificial neural network based image compression algorithm given in section 3.10.1 and used for simulations is summarized as follows.

4.2.1 Training

1) Create a neural network with this architecture.
   64 Input Neurons.
   2-16 Hidden Neurons.
   64 Output Neurons.
   Set *noise flag* and *reinitialization flags* to zero.
   Specify maximum number of epochs *(me)*
2) Break the training image into 8x8 blocks.
3) Construct a training set by randomly selecting a specified number of blocks from the image.
4) Convert the blocks in the training set into 64x1 vectors.
5) Scale each vector in the training set from an integer with a maximum value of 255 to a real number with a maximum value of 1.0.
6) Use the scaled vectors as input to the neural network.

i) Train the neural network using gradient descent minimization technique until the mean squared error between target and input vectors (in this case the target and the input are the same) is less than a specified threshold. If the error goal is reached then stop the training.

ii) If the error goal is not reached after presenting epochs *me (maximum number of epochs)* times, and *noise flag* is zero, add some noise to weight values. Set *noise flag* to one and go to 6-i. If the noise flag is one, then go to 6-iii. If the error goal is reached stop training.

iii) If the *reinitialization flag* is zero, reinitialize the weights. Set *reinitialization flag* to one, then go to 6-i.

iv) If both *noise flag* and *reinitialization flag* are one stop training.

The learning time is related with the size of the training set; when all of the blocks in an image are used in training set, the training time is inherently increased. In this case the network tries to internally represent all the details of training set and the desired error goal may not be reached within a reasonable time interval. By using a training set consisting of randomly selected vectors from the training image the desired error goal is reached with less computational effort.

A neural network having 9 hidden layer nodes is trained using gradient descent minimization technique by 100 samples, that are randomly selected from the image "Zelda". The Figure 4.3 shows the variation of the sum squared error versus the number of training epochs. As can be seen from the illustration the sum squared error does not decrease after 2000 epochs. Therefore the maximum number of epochs *(me)*, used in all training is set to 2000, the training is stopped after 2000 epochs even if the error goal is not reached.

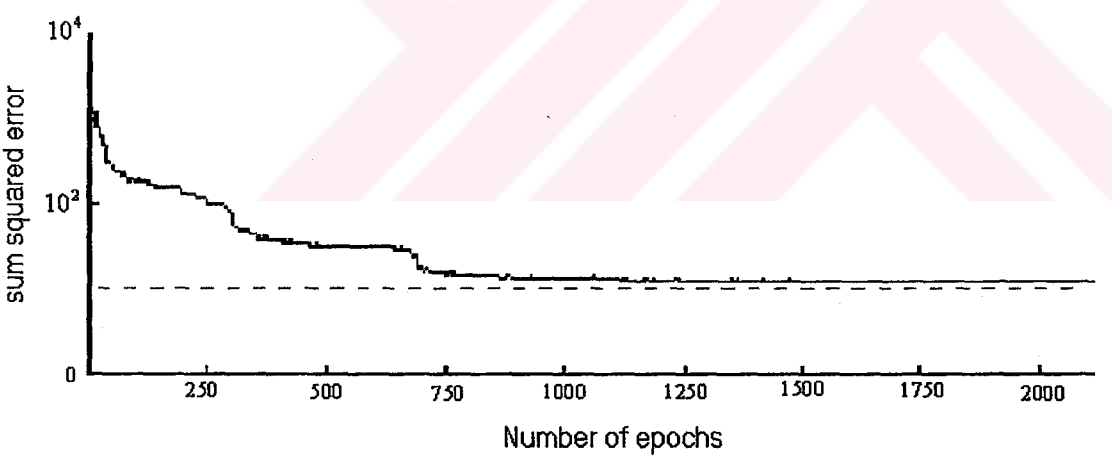


Figure 4.3 The sum squared error vs. number of training cycles for a neural network having 9 hidden layer nodes with 100 training vectors.

Sometimes the learning is affected by initial weight values. If the error goal is not reached the weights are reinitialized and the training is repeated. After presenting *me* epochs, if the error goal is still not reached it means that the network is most probably in a local minimum point. In this case some noise is added to weight values and the training is repeated. Not to be held at a local minimum, a momentum factor

of 0.9 is used. The sum-squared error goal for training the neural network with 100 training vectors is set to 8 in the training phase.

In the worst case, the network is trained 3 times by the maximum number of epochs: one for the first training, one for addition of noise to weight values and one for reinitialization.

4.2.2 Compression

1) Break the image into 8x8 blocks
2) Convert these blocks into 64x1 vectors.
3) Scale each vector from an integer with a maximum value of 255 to a real number with a maximum value of 1.0.
4) Use the scaled vectors as input to the neural network.
5) Take the output vector of the hidden Layer.
6) Quantize this data to 8 bits.
7) Save the quantized data.
8) Repeat for each block of the image.

4.2.3 Decompression

1) Read the saved quantized data.
2) Set the output of the hidden layer equal to the read data.
3) Find the response of the output layer.
4) Scale the real output to an integer value between 0 and 255.
5) Put scaled output into image.
6) Repeat for each block of the image.

The network is trained by using gradient descent minimization technique with an image of 512x512 pixels. The learning rate is set to 0.1, and the momentum is set to 0.9. The logsig is used as thresholding function in a neuron.

By changing the number of hidden layer nodes the compression ratio is changed. The compression ratio and PSNR values are calculated for each number of hidden layer nodes. From the results of these tests it is seen that the compression efficiency varies with the number of hidden layer nodes (see Table 4.2). As the number of hidden layer nodes is increased the quality of decompressed image is increased, but the compression efficiency is decreased.

Table 4.2 Number of hidden layer nodes/ DCT coefficients used in decompressed image versus PSNR value for image "Zelda". PSNR values are in dB. The network is trained with the same image for neural network based method.

Hidden layer nodes / DCT coefficients	7	8	9	10	11
PSNR for neural network based method	20.21	26.40	27.19	27.28	28.33
PSNR for DCT based method	34.88	35.44	35.99	36.33	36.43

Inspecting Table 4.2 it is seen that compression performance is better for DCT based compression method. However unlike the traditional image compression algorithms, neural network based compression and decompression consists of a simple matrix multiplication and therefore the speed of the compression system increases. For example with using Matlab routines on a personal computer (Pentium-2, 350 MHz, 64 MB RAM) the calculation of DCT coefficients for a 512x512 image (4096 8x8 blocks) takes 10.32 seconds. Using the same computer running another program written in Matlab, for simulating a neural network having 64 input, 8 hidden, 64 output nodes, calculation of the hidden layer outputs of the neural network for the same image takes 1.44 seconds. Significant improvements can be obtained in speed over a serial implementation by realizing a neural network with an inherently parallel architecture.

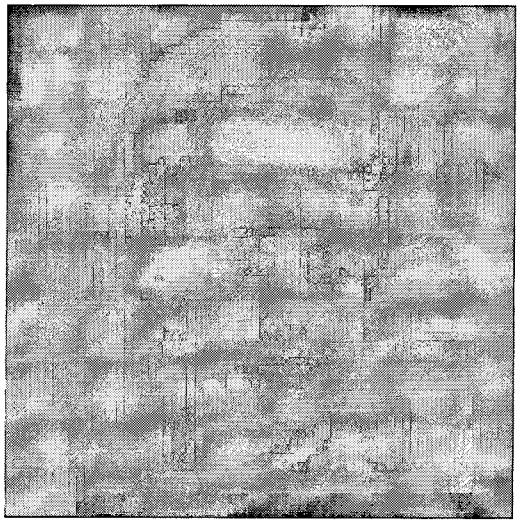
Exploiting the generalization capability of the neural networks which allows the same network to be used for untrained data, we can use the same neural network for compression of statistically similar untrained images. The untrained images are also compressed and decompressed by this algorithm. Table 4.3 and Figure 4.4 show the

simulation results for neural network based compression applied to trained and untrained images. More useful results are obtained for the images that are similar to the training image. For example the PSNR value for image "Lena" is higher than that of images "Peppers" and "Boat". The reduction in quality of decompressed image can be improved by additional training using old weight values as initial weights for untrained images.

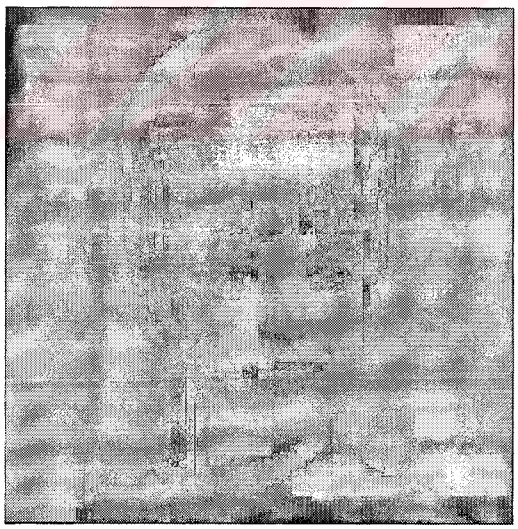
Table 4.3 PSNR values for untrained images, a network with 8 hidden layer nodes is trained with image "Zelda".

IMAGE	Zelda	Lena	Peppers	Boat
PSNR (dB)	26.40	22.10	21.65	21.17

Generalization is one of the properties that make neural networks favorable over most other conventional information processing techniques for many problems. The property that has the greatest impact on the generalization ability is the network architecture and, with that, the number of hidden units. Basically, the more hidden layer units a network has, the larger the number of functions it can compute. On the other hand, when there are only few hidden layer units, the network can not store the complete pattern set and has to extract features in common to many patterns, which will naturally help to generalize the new patterns.



8 hidden layer nodes, PSNR=25.60



11 hidden layer nodes, PSNR=28.33

Figure 4.4 Reconstructed images, network is trained by image 'Zelda', Number of hidden layer nodes is changed.

Since this technique require training to calculate the parameters of the encoder, care must be taken in selecting the training images. They must be representative of the class images for which the network is to be used. For example, if natural images are to be processed, the training set should include a variety of naturally occurring senses. On the other hand, if artificial images such as synthetic aperture radar (SAR) or magnetic resonance imaging (MRI) are used, then the training set must be representative of that class of images.

Among different commercial applications that use image compression, most are designed for use in low noise environments. For these applications, the DCT-based compression algorithm is an attractive choice. However, an increasing number of applications are required for use in high noise environments. In these applications, DCT-based and similar image compression algorithms, which rely on entropy coding, are not suitable [5]. Artificial neural networks are robust for high noise environments. Of course a price must be paid for this robustness. In this case, the price is a reduction in decompressed image quality for the same compression efficiency.

4.2.4 Preprocessing with Region-Based Segmentation

In order to improve the compression performance of neural network based compression algorithm, region-based segmentation can be used for preprocessing of all input data [31]. First the image to be compressed is segmented into a set of regions by using quad-tree based segmentation. And neural network based compression is applied to segmented image.

A region is accepted as homogenous if the difference between the maximum and the minimum elements of that region is lower than a threshold value. Threshold value used for simulations in this thesis is 0.4. The quad-tree segmentation algorithm which is explained in section 1.4.3 is used for segmentation stage. Also in order to reduce the complexity of segmentation section a region is not split anymore if the dimension of processed region is 16x16. And the region is segmented if it has a dimension greater than 64x64 without inspecting the homogeneity criteria. By this

way three different sets of regions having dimensions 16x16, 32x32, and 64x64 are obtained.

Each of these regions are separated into 8x8 blocks and these blocks are converted into 64x1 data vectors to obtain three different vector sets. Three different training sets are obtained by selecting 100 random vectors from each of these vector sets. These training sets are used to train a different neural networks each will be used to compress corresponding region. The training approach given in section 4.2.1 is used with these training sets. The weight sets for these neural networks are used to compress the corresponding region of the segmented image.

The compression algorithm used for the hybrid technique that consists of a preprocessing stage of quad-tree based segmentation and artificial neural networks is summarized as follows.

1) Scale pixel value range to 0-1.
2) Apply quad-tree segmentation to scaled image.
3) Convert regions into vectors of 64x1.
4) Use these vectors to obtain training sets. The elements of the training set are randomly selected from these vector sets.
5) Use these training sets to train neural networks.
6) Use the calculated weight sets to compress the corresponding region in the image.

The tests are repeated for different images. Table 4.4 shows PSNR values for different images, compressed and decompressed using neural networks having 8 hidden layers. Three networks with 8 hidden layer nodes is trained with image "Zelda". Images are preprocessed by quad-tree segmentation, and trained neural networks are used for compression. Test results for the compression without preprocessing are also shown in the same table.

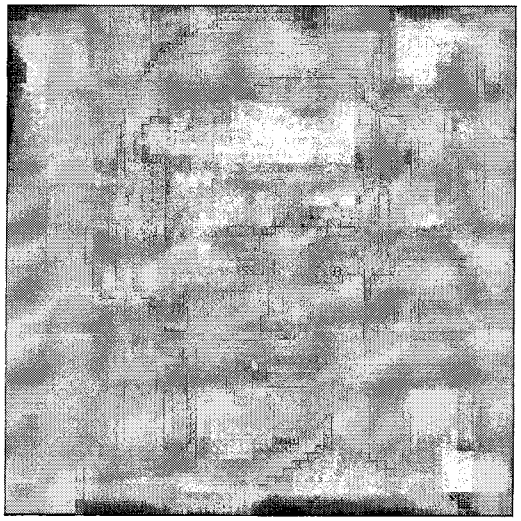
Table 4.4 PSNR values for compression systems using preprocessing stage the system without preprocessing stage. PSNR values are in dB.

IMAGE	Zelda	Lena	Peppers	Boat
PSNR for image that is Preprocessed with quad-tree segmentation	27.30	22.68	21.85	21.35
PSNR for the image that is not preprocessed	26.40	22.10	21.65	21.17

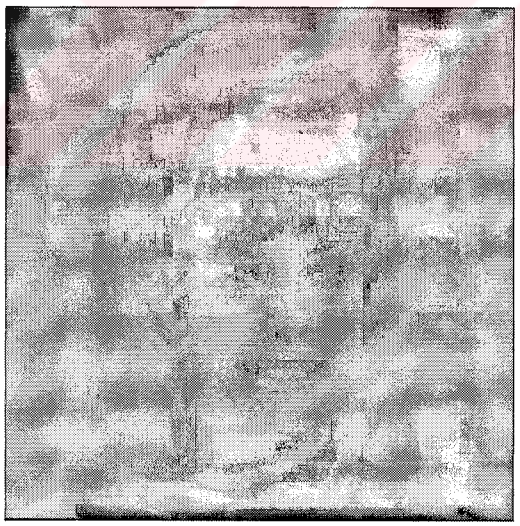
As inspected from Table 4.4, the compression performance is improved by using the preprocessing stage; when using only one neural network for compression of an image the performance is a little poor. Of course the results for untrained images can be improved by additional training.

Figure 4.5 shows the original and reconstructed images for a compression system consisting of 3 neural networks with 9 hidden layer nodes. The neural networks are trained with image "Zelda". The image is preprocessed by quad-tree based segmentation and neural networks are used for compression and decompression. For the compression system using the preprocessing stage, the PSNR value is 27.97. The same image is compressed and decompressed with a compression system consisting of only a single neural network having the same number of hidden layers. The PSNR value of 27.19 is achieved for the compression system that does not use the preprocessing stage.

Advantage of using segmentation as the preprocessing stage and using different neural networks is that; each neural network is specialized on a different region and this improves the compression performance of the system. Three sets of weights are used for decompression of image data instead of using only one weight set. The weight sets obtained from training one neural network can be used as the initial weights for the next neural network for training. This reduces the training time compared with training each neural network independently having random initial weight values.



Reconstructed image for a neural network having 9 hidden layer nodes, PSNR=27.19



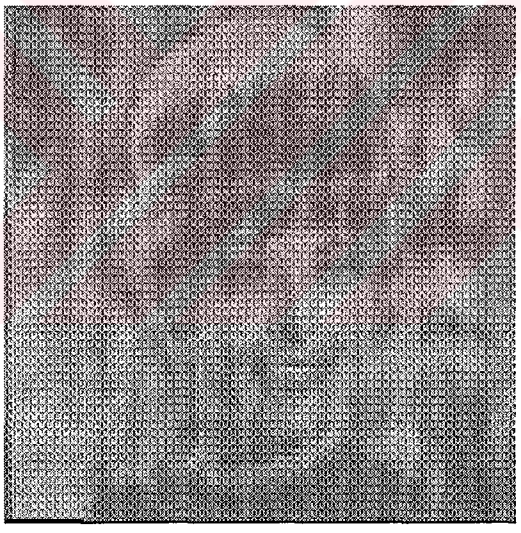
Reconstructed image for hybrid system, PSNR=27.97

Figure 4.5 Reconstructed images "Zelda" for compression system without preprocessing and for the hybrid compression system, neural networks having 9 hidden layer nodes and trained with the same image are used.

4.2.5 Fault tolerance test

In order to inspect the fault tolerance property of artificial neural networks in digital image compression first a neural network having 9 hidden layer nodes is used to compress the image "Zelda". Then all the weights of a hidden layer node of the neural network are set to zero and the network is used to compress the same image. In Figure 4.6the reconstructed images are shown for the original network and damaged network. It is seen that even if all the weights of a node of the network are lost the network continues to operate. In this case only the performance of the system is degraded.

Reconstructed image for the compression system with a neural network having 9 hidden layer nodes PSNR=27.19



Reconstructed image for the compression system with a neural network having 9 hidden layer nodes, one hidden layer output is not used for reconstruction. PSNR=13.27

Figure 4.6 The effect of removing one hidden layer node output for reconstruction of the image "Zelda".

CHAPTER FIVE
CONCLUSIONS

The aim of this work is to use artificial neural networks for digital image compression. The test results has shown that the use of feedforward artificial neural networks is a valid and useful method for image data compression.

A feedforward artificial neural network is used to compress digital image data. DCT based compression technique is also simulated. The neural network based method is poor in compression performance compared with DCT based compression technique, but it is faster to implement. The decompressed image quality is increased with a hybrid compression system that consists of a preprocessing stage of quad-tree based segmentation.

The main disadvantage of using feedforward artificial neural networks for image compression is that, neural networks require a certain number of training cycles before any appropriate output can be obtained. This results in a negative effect upon processing speed.

Training time is directly related with the size of the training set. By using a training set consisting of randomly selected vectors from the training image, the training time is reduced.

One of the major features of artificial neural networks is their ability to generalize, that is, to successfully classify patterns that have not been previously presented. Multilayer perceptrons generalize by detecting features of input pattern that have been learnt to be significant, and so coded into the internal units. Thus the same neural network can be used for compression of untrained images, without requirement of retraining. A neural network trained on a test image is used for compression of different images, and it is seen to be useful for compression of statistically similar images. For images that are not statistically similar further training is required.

In terms of compression performance, neural network technology standing alone can not, at the present stage, provide satisfactory solutions to practical image coding problems compared with traditional techniques, but on the other hand today's parallel structure artificial neural networks are robust for high noise environments. Among different commercial applications that use image compression, most are designed for use in low noise environments. For these applications, the DCT-based compression algorithm is an attractive choice. However, an increasing number of applications are required for use in high noise environments. In these applications, DCT-based and similar image compression algorithms are not suitable and alternatively artificial neural networks can be used for digital image compression. Artificial neural networks are intrinsically fault-tolerant, since they are distributed parallel processing elements, with each node contributing to the final output response. If a node or its weights are lost or damaged, recall is impaired in quality,

but the distributed nature of the information means that damage has to be extensive before a networks response degrades badly. This is an important consideration for tasks such as spacecraft image compression systems.

1. Future Work

More optimized segmentation algorithms can be used to improve the compression performance for preprocessing with region based segmentation. The segmentation threshold can be selected depending on the image to be compressed. For further enhancement in decompressed image quality, for data blocks which are reasonably smooth a small number of hidden layer nodes may be used, while maintaining a high quality decompressed image. This is not, however true for data blocks with lots of detail. For these blocks, a large number of hidden layer nodes will be required in order to maintain a reasonable image quality.

Theoretically, all the existing state-of-the-art image compression algorithms can be implemented by extended neural network like structures. From this point of view, image compression development is essentially the design of learning algorithms for neural networks. In this way, better algorithms in terms of both compression performance and high speed processing can be produced in the near future.

REFERENCES

[1] Adobe. TIFF, (1992, June.). Tag based Image File Format, revision 6.0. http://www.adobe.com/Support/TechNotes.html/.

[2] Ahmed, N., Natarajan, T., & Rao K. R. (1974, January). Discrete Cosine Transform, IEEE Trans. Computers, C-23, 90-93.

[3] Basso, A. (1992, 4-7 September). A massively Parallel Implementation of autoassociative neural networks for image compression, In Proc. Int. Conf. On Artificial Neural Networks-ICANN92, Brighton (UK).

[4] Beale, R. & Jackson, T. (1990). Neural Computing an Introduction, Adam Hilger, Bristol, England.

[5] Benbenisti, Y., & Kornreich, D., Mitchell, H.B. & P.A., Schaefer, (1999, September). Fixed Bit-Rate Image Compression Using a Parallel-Structure Multilayer Neural Network, IEEE Transactions on Neural Networks, 10, No. 5., 1166-1172.

[6] Chrysafis, C., (2000, March). Wavelet Image Compression rate distortion optimizations and complexity reductions, Ph.D. thesis, University of Southern California.

[7] Cottrell G., W, Munro P,& Zipser D, (1987). Learning Internal Representations from Gray-Scale Images: An Example of Extensional Programming, Cognitive Science Soc.Annual Conference, 9th Procs.

[8] Cover, T. M., & Thomas, J. A., (1991). Elements of information theory. Wiley Series in Communications.

[9] Dony, R., D., & Haykin, S., (1995, February). Neural Network Approaches to Image Compression, Proceedings of the IEEE, 83, 288-303.

[10] Gelenbe E., & Sungur M., (1994). Random Network Learning and Image Compression.IEE World Congress on Computational Intelligence., 6 , 3996-3999.

[11] Gonzalez, Rafael C. & Richard E. (1992). Digital Image Processing. Addison-Wesley Publishing Company, New York.

[12] Hauf, C. R. & Houchin, J. C., (1996, November). The FlashPix(TM) image file format. Proceedings of the Fourth Color Imaging Conference: Color Science, Systems and Applications, 234-238.

[13] A. Hoekstra. (1998). Generalization in Feed Forward Neural classifiers. Ph.D. thesis, Technical University of Delft, Delft, Netherlands.

[14] Huffman, D. A. (1952). A method for the construction of minimum redundancy codes. Proc IRE, 40, 1098-1101.

[15] Information of IEEE Neural Network Council, http://engine.ieee.org/nnc

[16] ITU, (1993). Standardization of Group 3 Facsimile Apparatus for Document Transmission, ITU-T Recommendation T.4. ITU.

[17] ITU, (1992). Information Technology Digital Compression and Coding of Continuous-Tone Still Images Requirements and Guidelines. ITU Recommendation T.81. ITU.

[18] Jiang, J., (1995, September). Neural Network Technology for Image Compression, International Broadcasting Convention, 14-18, Conference Publication No: 413, IEE.

[19] Kılıç, İ., (1997, February). Digital Image Compression Techniques, Ms.Sc. thesis, Dokuz Eylül University, İzmir.

[20] Langdon, G. (1984). An Introduction to Arithmetic Coding. IBM J. Res. Develop., 135-149.

[21] Lehn, W., (1991). Digital Image Processing, Lecture notes. Winnipeg, M.B. Department of Electrical and computer Engineering, University of Manitolia.

[22] Linde, Y., Buzo, A., & Gray, R. M., (1980, January). An introduction for vector quantized design. IEEE Trans. Commun., C-28, 84-95.

[23] Masselos. K.. Stouraitis. T. & Goutis. C.E, (1998). Novel Codebook Generation Algorithms for Vector Quantization Image Compression, Proceedings of the IEEE International Conference on. Acoustics, Speech and Signal Processing. 5 . 2661 –2664.

[24] Nelson, M., McCord. I.. & llingworth, W.T. (1990). A practical Guide to Neural Nets. Addison-Wesley.

[25] McCulloch, W.S., & Pitts, W. (1943). A Logical Calculus of Ideas immanent in nervous activity, Bulletin of Mathematical Biophysics, 5, 115-133.

[26] Minsky, M. L., & Papert, S. A.(1969). Perceptrons, Cambridge: MIT Press.

[27] Nelson, M. (1995, November). The Data Compression Book, 2nd ed., M&T books.

[28] Oren, E., B., (1988). The Fast Fourier Transform and Its Applications, Englewood Cliffs. Prentice-Hall, Inc., 448.

[29] PC Week/Russian Eddition, (1995, 24 April). 1.

[30] Pennebaker, W.B., & Mitchell, J.L., (1993). JPEG - Still Image Data Compression Standards, Van Nostrand Reinhold.

[31] Pinho, A. J., (1998). Improvement of Region-based Image Coding by Neural Networks, The 1998 IEEE World Congress on Computational Intelligence 2, 1338-1343.

[32] Reeder, Brian, M., Gough, M. Paul, (1996). Application of artificial neural networks for spacecraft instrument data compression, Microprocessors and Microsystems, 20 ,285-295.

[33] Rissanen, J. & Langdon, G. G. (1979, March). Arithmetic Coding. IBM J. Res. Develop., 149-162.

[34] McClelland, J. L., & Rumelhart, D.E., (1986). Parallel distributed Processing. MIT Bradford Press.

[35] Schindler, M. (1998, October), Practical Huffman Coding. http://www.compressconsult.com/huffman

[36] Sirat, J.A., Viala, J.R., & Remus, C., (1989). Image Compression with Competing Multilayer Perceptrons, IEE Int. Conference on Neural Networks, 404-409.

[37] Vetterli, M. & Kovacevic, J. (1995). Wavelets and Subband Coding, Englewood Cliffs, Prentice Hall, http://cm.bell-labs.com/who/jelena/Book/home.html.

[38] Wahhab, O. A., & Fahmy, M.,(1997, June 9-12). Image Compression using Multi-layer Neural Networks, IEE International Symposium on Circuits and Systems, 1273-1276, Hong Kong.

[39] Wallace, G. K. (1991, April). The JPEG Still Picture Compression Standard, Comm. ACM, 34-4, 30-44.

[40] Werbos, P., (1974). Beyond regression: new tools for prediction and analysis in the behavioural sciences. Ph.D. thesis, Harvard University.

[41] Widrow, B., & Hoff, M. E., (1960). Adaptive Switching Circuits. IRE WESCON Convention record, 96-104. New York: IRE.

[42] Witten, R. M., Neal, I. H., & Cleary, J. G. (1987, June). Arithmetic Coding for Data Compression, Communications of the ACM, 30-6, 520-540.