

**NEAR EAST UNIVERSITY**

**Faculty of Engineering**

**Department of Computer Engineering**

**IMAGE CLASSIFICATION USING ARTIFICIAL  
NEURAL NETWORK**

**Graduation Project**

**COM – 400**

**Student: FIRAS SULAIMAN BHAR**

**Supervisor: Associated. Prof. DR. RAHİB ABİYEV**

**Nicosia - 2006**

## ACKNOWLEDGEMENTS

*First* of all I am thanking full to the most gracious “ALLAH” the almighty, who enable me to complete this project

*Secondly*, I would like to award my supervisor Associated. Prof. DR. RAHİB ABİYEYEV for being so operative averring supervises me in this work, and for his overwhelming and limitless help he had done to me.

*Thirdly*, I will never over look the encourage I had resaved from all my family, specially my parents, my “father” the best father in this world, and the best sweetest woman in the word “mother”, for there supporting me and caring for me, I am thankful for them.

*Fourthly* this will be the most important moment in my life that I will settlement my project to my brothers” Rawad, Rami, Abd El Hakeem and to my sister Marwa” and surly to the best uncle “DR. Amen Bhar”, because of there encourage me and supporting me in all my educations and specially in the last three and half years of study in Near East University, I am so thankful for them, for all my relatives and for every one caring for me.

*Finally* I like to thank the best friends Youssef Abu Khuruj, Mohamad Al Sharaf, Mahamoud Ghuneim, Youssef Ghuneim, Mohammad Majzoob, Khaled Ghuneim, Ahmad Hameid, Fadi Alia and Shadi El Haj Hussein.

My best friends I had met them in Cyprus Amir Sarayreh, Abdul Halim Abu kuwaik, Esen Sultan Çil, Mohammad Aldarabie, Saied Almoheisin, Mahmoud Antar, Mohammad Sarhan, Rami Hasan Amar, Hicham Tirawi, Serein Hassan and to all my friends wherever they are.

I will not forget my doctors and teachers in Near East University. From all my heart I am saying thanks to all and to every one stood beside me to complete my education.

# ABSTRACT

Since the beginning, humankind has sought to use elements in the surrounding environment to make life easier and the tasks at hand more efficient. In keeping with this tradition, people have toyed with and explored the concept of using machines to solve problems since ancient times, Only in this 20th century have significant advances occurred, making the possibility of an actual manifestation of artificial intelligence more and more a reality.

This project explores the theoretical and particular underpinning of Neural Networks and its applications, the reader of this project will come away with an appreciation for the basic concepts of Neural Network., and with an idea about Image classification using Artificial Neural Networks field and the use of its applications.

This projects includes three kinds of network are using for image classification, also this project is supplied with MATLAB code which help in implementing and training the neural network for image classification and the reader of this project will learn about the wavelet decomposition which help in compression or reducing the image size without losing the image data.

# Table of Contents

<b>ACKNOWLEDGEMENTS</b>	i
<b>ABSTRACT</b>	ii
<b>TABLE OF CONTENTS</b>	iii
<b>INTROUDUCTION</b>	1
<b>CHAPTER 1: ARTIFICIAL NEURAL NETWORKS</b>	
1.1 Overview	5
1.2 Introduction to Artificial Neural Network	5
1.2.1 Artificial Neural Networks	6
1.2.2 Analogy of the Brain	6
1.2.3 Artificial Neurons and how they work	7
1.3 Components of Artificial Neural Network.	9
1.3.1 Neurons	9
1.3.2 Layers	9
1.3.3 Connections (weights)	10
1.3.4 Transfer Function	10
1.4 Artificial Neural Network: Operation Mode and Training Mode	11
1.4.1 Operation Mode	11
1.4.2 Training Mode	13
1.4.2.1 Supervised Training.	13
1.4.2.2 Unsupervised Training (Adaptive Training)	14
1.5 Artificial Neural Networks versus Traditional Computing	16
1.6 The Artificial Neural Network Applications	16
1.6.1 Banking and Financial	17
1.6.2 Language Processing	17
1.6.3 Character Recognition	17
1.6.4 Image (data) Compression	18

1.6.5 Pattern Recognition	18
1.7 Neural networks structure	19
1.7.1 FeedForward-Back Propagation Neural Network	19
1.8 Summery	22

## **CHAPTER 2: CLASSIFICATION METHOD**

2.1 Overview	23
2.2 Statistical Methods	23
2.2.1 Maximum likelihood Classification	23
2.2.2 Minimum distance Classification	24
2.3 Classification using Artificial Neural Network	24
2.3.1 Competitive Neural Network	24
2.3.1.1 Basic Operation of Competitive Layer	25
2.3.1.2 Source of error in the competitive layer network	26
2.3.1.3 Bias Learning Rule (learncon)	26
2.3.2 Learning Vector Quantization(LVQ)	27
2.3.2.1 Architecture of Learning Vector Quantization Network	27
2.3.2.2 Training of LVQ Networks	29
2.3.2.3 Drawbacks of LVQ Networks	30
2.4 Summary	30

## **CHAPTER 3: INTRODUCTION TO WAVELET ANALYSIS**

3.1 Overview	32
3.2 The need of wavelet transformation	32
3.3 The Drawback of Fourier Transform	33
3.4 The solution of the limitation of the Fourier Transform	33
3.5 The appearance of Wavelet analysis	34
3.6 Wavelet Computing	35
3.6.1 The Continuous Wavelet Transform and the wavelet series	36



3.6.2	The Discrete Wavelet Transform	37
3.6.3	DWT and Filter Banks	38
3.7	Wavelet Families	40
3.8	Summary	41

#### **CHAPTER 4: DESIGN OBJECTIVE AND PREPROCESSING**

4.1	Overview	42
4.2	Image characteristics	43
4.3	The MTLAB Wavelet Toolbox usage	47
4.3.1	The decomposition process	48
4.3.2	The reconstruction Process	48
4.4	Images preprocessing; the use of WAVELET transformer	49
4.5	Summary	55

#### **CHAPTER 5: FEEDFORWARD DESIGN**

5.1	Overview	56
5.2	The FeedForward Construction	56
5.3	FeedForward ANN Design using the original size	59
5.4	The use of Wavelet transformation	64
5.5	The end results of FeedForward design	66
5.6	Summary	67

#### **CHAPTER 6: CLASSIFICATION USING COMPETITIVE LAYER ANN**

6.1	Overview	68
6.2	Construction of Competitive Layer ANN	68
6.3	Network design for the original size images	69
6.4	Network design for reduced images size	71
6.4.1	Network design using third wavelet decomposition	72

6.4.2	Network design using fourth level wavelet decomposition	77
6.4.3	Network design using fifth level wavelet decomposition	82
6.5	Summary	87

## **Chapter 7: CLASSIFICATION USING LEARNING VECTOR QUANTIZATION**

7.1	Overview	88
7.2	Construction of LVQ ANN	88
7.3	Network design for the original size images	89
7.4	LVQ Network design for reduced images size (one wavelet level)	91
7.5	LVQ Network design for reduced images size (two levels wavelet)	94
7.6	LVQ Network design for reduced images size (three levels wavelet)	94
7.7	Network design for reduced images size (four levels wavelet)	99
7.8	Network design for reduced images size (five levels wavelet)	101
7.9	Summary	102

<b>CONCLUSION</b>	<b>103</b>
<b>LIST OF FIGURES</b>	<b>105</b>
<b>LIST OF TABLES</b>	<b>106</b>
<b>APPENDIX</b>	<b>I</b>
<b>REFERENCES</b>	<b>R</b>

# Introduction

Image classification plays an important part in the fields of Remote sensing, Image analysis and Pattern recognition. Digital image classification is the process of sorting all the pixels in an image into a finite number of individual classes. The conventional statistical approaches for image classification use only the gray values.

Digital image consists of discrete picture elements called pixels which are associated with a digital number represented as DN that depicts the average radiance of relatively small area within a scene. The range of DN values is normally 0 to 255. Digital image processing is a collection of techniques for the manipulation of digital images by computers. Classification generally comprises four steps:

- 1- Pre-processing: Atmospheric correction, noise suppression, and finding the band ratio, principal component analysis, etc.
- 2- Training: Selection of the particular feature which best describes the pattern.
- 3- Decision: Choice of suitable method for comparing the image patterns with the target patterns.
- 4- Assessing the accuracy of the classification.

## Project description

In this project we are discussing the image classification using artificial neural network, since ANN according to Haykin is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. ANNs can provide suitable solutions for problems, which are generally characterized by non-linearities, high dimensionality noisy, complex, imprecise, and imperfect or error prone sensor data, and lack of a clearly stated mathematical solution or algorithm. A key benefit of neural networks is that a model of the system or subject can be built just from the data.



This research consists of seven chapters, the goal of this research is to describe some neural network and how they are able to classify images, the description of each chapter as follow.

First chapter is Artificial Neural Networks:

This chapter presents an introduction to artificial neural network, and it contains a definition of artificial neural network, analogy of the brain, how artificial neurons work, it also discusses the components of artificial neural network, which contains neuron, layers, connections, and transfer function with description of each transfer function supplied in appendix, this chapter will also present the operation mode and training mod of artificial neural network, comparison between artificial neural network and traditional computing, applications of artificial neural network, and finally neural network structure.

Chapter 2: Classification Methods:

This chapter introduces the classification method which is divided into two parts first part was statistical method of classification which is divided into maximum likelihood classification and minimum distance classification. The second part was about classification using an artificial neural network which was also divided into competitive neural network and learning vector quantization network. Also we will see in the appendix a description of the dist function which had been used in the competitive layer architecture and in the learning vector quantization network architecture.

Chapter3: Introduction to Wavelet Analysis:

This chapter discusses the wavelet analysis, its needed, the drawback of Fourier transform and its solution with STFT then the problem with STFT, also it discussed the appearance of wavelet analysis and its better resolution properties and its high compression capabilities, then it discussed the wavelet computing which divided into three parts: the continuous wavelet transform and the wavelet series, discrete wavelet transform (DWT)

and DWT and filter banks. And finally it shows some of basis functions that can be used as the mother wavelet for wavelet transformation which related to wavelet family's part.

#### Chapter4: Design objective and Preprocessing:

This chapter discussed some MATLAB codes and it show how they work on the selected images with some examples, it used the wavelet transformation which helped in the data processing, but we involved with the two dimension discrete branch of the wavelet transformation and it was applied on the decomposition and reconstruction process. Also it discussed the cause of using the two dimension wavelet transformation through an example using the MATHLAB code.

#### Chapter5: FeedForward Design:

This chapter discussed on of the most popular and effective network which is the Feedforward architecture, its construction, its training and the use of wavelet transformation within the Feedforward in order to reduce the size of the image which will reduce the number of neurons in the input layer, although the result of training was failed even by trying the fifth and sixth level of decomposition.

#### Chapter6: Competitive Layer ANN Design:

This chapter discussed the competitive layer ANN design which is an unsupervised training ANN, the architecture of this network contain only one layer which is the competitive layer. This chapter shows the network design for the original size image using the MATLAB code which was failed , the this chapter discussed the network design for reduced image size which failed to classify the images in the first and second order wavelet decomposition, but it was successful for the third , forth and fifth order wavelet decomposition , but as the experiment shows that network design using fifth level wavelet decomposition was better than the network design using the third and the forth order wavelet decomposition.

## Chapter7: Classification using Learning Vector Quantization:

This chapter discussed the classification using learning vector quantization because its architecture has the architecture of competitive layer plus the ability to have supervised training; this chapter show the construction of the LVQ ANN using MATLAB code. The LVQ ANN was applied on the original image but the training failed so the wavelet decomposition was considered, and it was started from the first wavelet decomposition which got failed and the same result for the second wavelet decomposition order, so it considered the third, forth, and fifth order of wavelet which was successfully trained and the fifth wavelet decomposition order proved that it's the best order in the classification.

### **Aim of project**

The aims of this project are:

- 1- To give general information about ANN, Classification method, and Wavelet Analysis.
- 2- To introduce some networks and how they are able to classify images.
- 3- Using the MATLAB code for implementing the networks and to train each network
- 4- To show the differences between the used networks and to show that Learning vector quantization is better than other used network.



# **Chapter One**

## **Artificial Neural Networks**

### **1.1 Overview**

This chapter presents an introduction to artificial neural network, and it contains a definition of artificial neural network, analogy of the brain, how artificial neurons work, it also discusses the components of artificial neural network, which contains neuron, layers, connections, and transfer function with description of each transfer function supplied in appendix, this chapter will also present the operation mode and training mod of artificial neural network, comparison between artificial neural network and traditional computing, applications of artificial neural network, and finally neural network structure.

### **1.2 Introduction to Artificial Neural Network**

Artificial Neural Networks are being touted as the wave of the future in computing. They are indeed self learning mechanisms which don't require the traditional skills of a programmer. But unfortunately, misconceptions have arisen. Writers have hyped that these neuron-inspired processors can do almost anything. These exaggerations have created disappointments for some potential users who have tried, and failed, to solve their problems with neural networks. These application builders have often come to the conclusion that neural nets are complicated and confusing. Unfortunately, that confusion has come from the industry itself. Avalanches of articles have appeared touting a large assortment of different neural networks, all with unique claims and specific examples. Currently, only a few of these neuron-based structures, paradigms actually, are being used commercially. One particular structure, the feedforward, back-propagation network, is by far and away the most popular. Most of the other neural network structures



represent models for "thinking" that are still being evolved in the laboratories. Yet, all of these networks are simply tools and as such the only real demand they make is that they require the network architect to learn how to use them.

### ***1.2.1 Artificial Neural Networks***

The basic concept of the Artificial Neural Network is that they are electronic models based on the neural structure of the brain. This familiarization with the brain structure give the Artificial Neural Network the ability to do more complex operation that computer can not do.

Advanced researches in biological structure of the brain, give us an initial understanding of the Neural thinking mechanism. It appears that the brain store information as patterns, some of these patterns are very complicated so it gives us the ability to recognize individual faces from many different angles.

This process of storing information as patterns, utilizing those patterns, and then solving problems, by establishing a new field in computing, does not utilize traditional programming but involves the creation of massively parallel networks and then training these networks to solve specific problems.

### ***1.2.2 Analogy of the Brain***

Although the exact exercises procedures of the human brain are still a mystery yet, some aspects of this amazing processor are known. In particular, the most basic element of the human brain is a specific type of cell which, unlike the rest of the body, doesn't appear to regenerate. Because this type of cell is the only part of the body that isn't slowly replaced, it is assumed that these cells are what provide us with our abilities to remember, think, and apply previous experiences to our every action. These cells, all 100 billion of them, are known as Neurons. Each of these neurons can connect with up to 200,000 other neurons.

The power of the human mind comes from the huge number of the neurons and the multiple connections between them, it also comes from learning.

Together these neurons and their connections form a process which is not binary, not stable, and not synchronous. In short, it is nothing like the currently available electronic computers, or even artificial neural networks.

The Artificial Neural Networks or (ANN) tries to replicate only the most basic elements of this complicated, versatile, and powerful organism.

### 1.2.3 Artificial Neurons and how they work

The basic processing element of the Neural Network is the Neuron, so it's recommended to know how the biological neuron works.

A biological neuron receive inputs from other sources (maybe other neurons), combines them in some way, performs a nonlinear operation on the result, and then outputs the final result. Figure (1) shows the main components of the biological neuron.

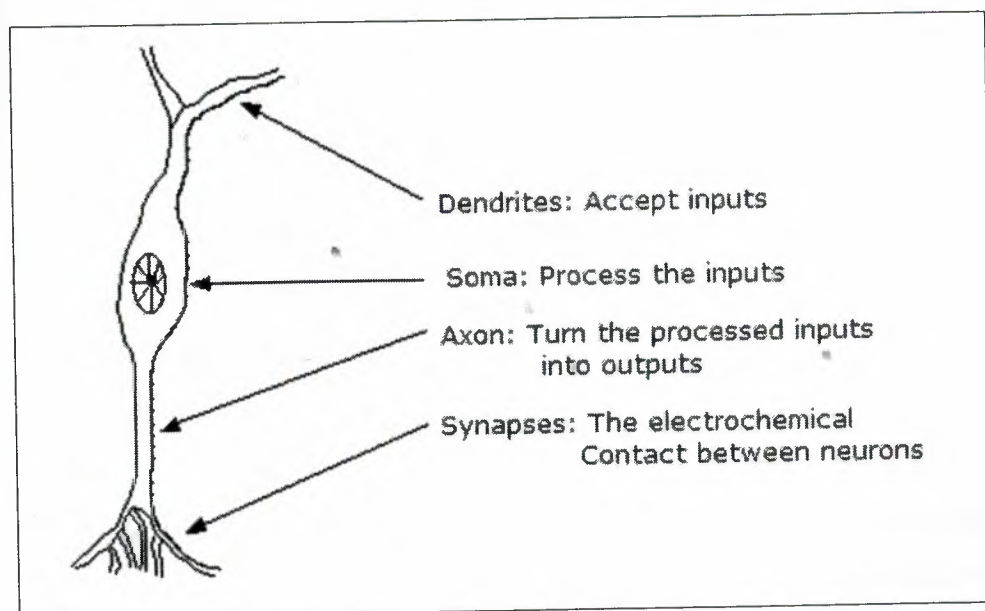


Figure (1): The main 4 parts of the Nerve Cell (neuron)

These components are known by their biological names - dendrites, soma, axon, and synapses. Dendrites are hair-like extensions of the soma which act like input channels. These input channels receive their input through the synapses of other neurons. The soma then processes these incoming signals over time. The soma then turns that processed value into an output which is sent out to other neurons through the axon and the synapses.

It's recommended to mention that the biological neurons are structurally more complex than the simplistic explanation above, and they are significantly more complex than the existing artificial neurons that are built into today's artificial neural networks.

The goal of the Artificial Neural Network is not to build an exact model of the human brain, but to understand the natural capabilities behind the human brain structure.

To do this, the basic unit of neural networks, the artificial neuron, simulates the four basic functions of natural neurons. Figure (2) shows the representation of the artificial neuron.

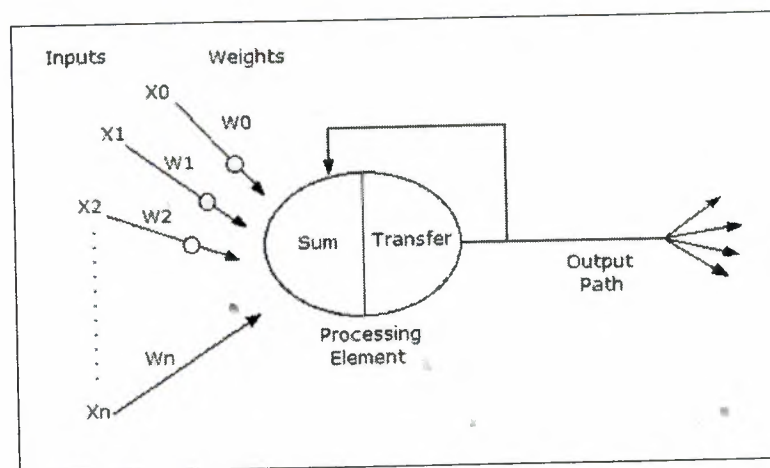


Figure (2): The Artificial Neuron

The synapses and dendrites of the artificial neuron are the inputs to the processing element (soma). Each of the inputs ( $x_n$ ) has an associated connection weight ( $w_n$ ) which simulates the strength of a particular synaptic connection. The processing

element multiplies each input by its connection weight and usually sums these products ( $I = \sum X_n W_n$ ), which is then passed to the transfer function  $f(I)$  to generate a result which is transmitted via the output path. The transfer function dictates the firing of the neuron. This could be based on a certain threshold level, a linear function, or a sigmoid function where the threshold for output varies. Neurons can be classed as excitatory or inhibitory depending on the effect their output has on the output of a target neuron

### 1.3 Components of Artificial Neural Network.

#### 1.3.1 Neurons

A simple artificial neuron is shown in figure (3). In this architecture the input ( $I$ ) is transmitted through a connection that multiplies its value by the weight ( $w$ ), to form the product ( $wI$ ). Then the bias is added to the product ( $wI$ ) which will apply to the transfer function ( $f$ ), to get the output. The bias is much like a weight, except that it has a constant input of (1).

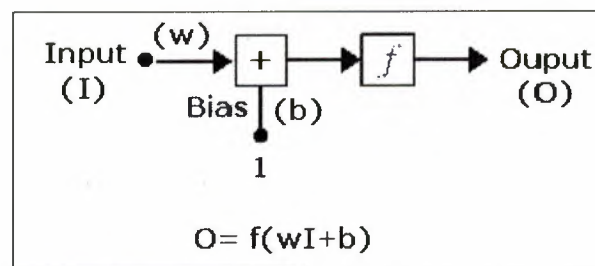


Figure (3): Simple neuron with bias

#### 1.3.2 Layers

The neurons are grouped in layers, which construct the neural network, The layer may vary in size from one neuron to large numbers, and the sequence of layers are connected through connections (weights).



The output of each neuron in one layer is connected to the input of each neuron in the next layer, and the strength of this connection depends on the value of the weight ( $w$ ). A general architecture of consecutive is shown in figure (4).

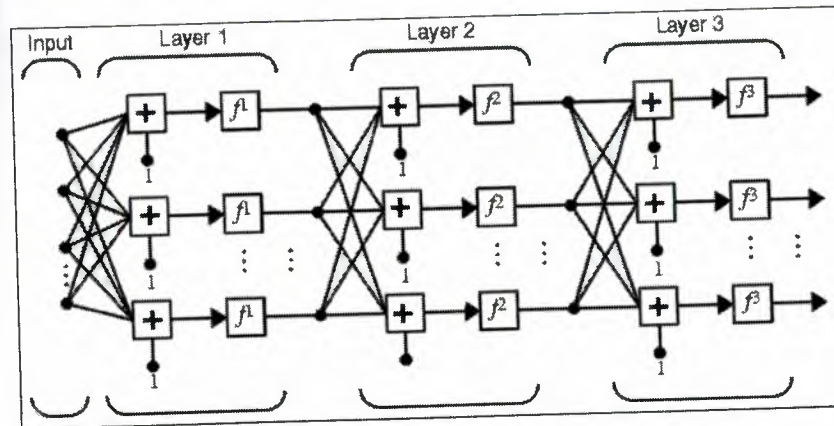


Figure (4): General architecture of multi layer neural network

### 1.3.3 Connections (weights)

The strength of the connections between the neuron and the next one in the next layer is defined by the weight. The weights of such layer are grouped in one matrix called the weights matrix.

### 1.3.4 Transfer Function

The multiplication result of the input by the weight added to the bias is then applied to the transfer function of the neuron. Each neuron can be defined separately with a single transfer function, while usually we define each layer with a transfer function, such that all of the neurons in the same layer have the same transfer function.

There are various transfer functions can be used some of them are listed in table (1);

Command	Description
Compet	Competitive transfer function.
Hardlim	Hard limit transfer function.
Hardlims	Symmetric hard limit transfer function
Logsig	Log sigmoid transfer function.
Poslin	Positive linear transfer function
Purelin	Linear transfer function.
Radbas	Radial basis transfer function.
Satlin	Saturating linear transfer function.
Satlins	Symmetric saturating linear transfer function
Softmax	Softmax transfer function.
Tansig	Hyperbolic tangent sigmoid transfer function.
Tribas	Triangular basis transfer function.

Table (1): Neurons Transfer functions.

## 1.4 Artificial Neural Network : Operation Mode and Training Mode

### 1.4.1 Operation Mode

The question now is how we can use these basic elements (neurons) to give the performance expected from it? From our knowledge in biological neural network the grouping of these basic elements is recommended.

This grouping occurs in the human mind in such way the information can be processed in a dynamic, interactive, and self organizing way. The limited power of the Artificial Neural Network with respect to the biological one come from the type of construction they use, the biological networks are constructed in a three dimensional way, and the integrated circuits used to implement the

artificial network are two dimensional devices. This physical reality restrains the types, and scope, of artificial neural networks that can be implemented in silicon.

The grouping in Artificial Network occurs by creating Layers which are then connected to each other. How each layer is connected to the other define the action and the ability of the network. The most common structure contains an Input Layer (non-processing neurons), Hidden Layer (processing neurons), and Output Layer (processing neurons), where processing means that the neuron can be ON or OFF while comparing with the threshold value. Figure (5) illustrates layers of artificial neural network which are then connected to each other.

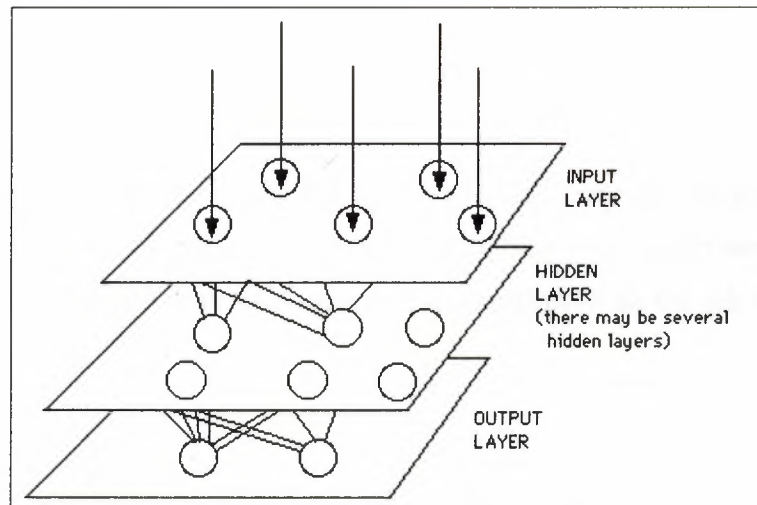


Figure (5): Artificial neural network layers

The input layer contain a specific number of neurons to interface the real world to receive its inputs, the received data may be either from input files or directly from electronic sensors, the opposite function is done by the neurons in the output layer which provide the real world with the network's output. Many hidden layers can be inserted between two layers. These hidden layers contain many of the neurons in various interconnected structures. The inputs and outputs of each of these hidden neurons simply go to other neurons.

Although there are useful networks which contain only one layer, or even one element, most applications require networks that contain at least the three normal types of layers - input, hidden, and output.

In the most networks each neuron receives the signals from all the neurons in the previous layer (or from sensors or files in the case of the input layer); after a neuron performs its function it passes its output to all the neurons in the next layer.

#### ***1.4.2 Training Mode***

As the biological network gain its information and the ability of solving problems from learning, the power of the Artificial Neural Network comes from training.

When the training process is started the network chose the weights of the connection between neurons randomly, and as the training goes on the weights are adjusted to provide the network with optimum connection to do the job expected from it.

There are two approaches to training; supervised and unsupervised. Supervised training involves a mechanism of providing the network with the desired output either by manually "grading" the network's performance or by providing the desired outputs with the inputs. Unsupervised training is where the network has to make sense of the inputs without outside help.

##### ***1.4.2.1 Supervised Training***

In supervised training, both the inputs and the outputs are provided. The network then processes the inputs and compares its resulting outputs against the desired outputs. Errors are then propagated back through the system, causing the



system to adjust the weights which control the network. This process occurs over and over as the weights are continually tweaked. The set of data which enables the training is called the "training set." During the training of a network the same set of data is processed many times as the connection weights are ever refined. Figure (6) illustrates the idea of the supervised training.

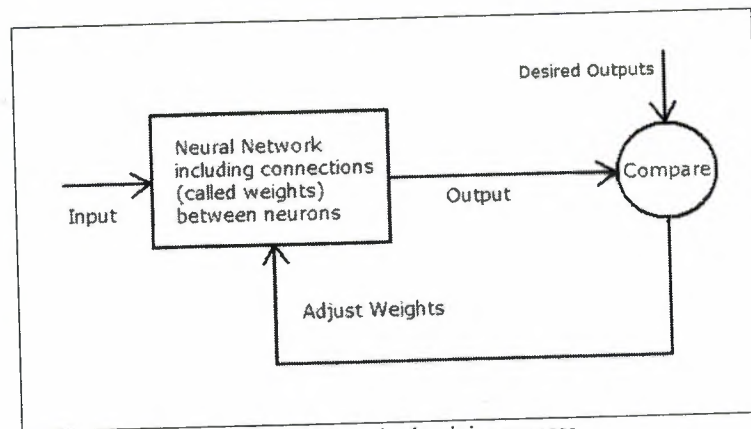


Figure (6): Supervised training process.

Some networks never learn. This could be because the input data does not contain the specific information from which the desired output is derived or the application is not supplied by the ANN architecture.

If a network simply can't solve the problem, the designer then has to review the input and outputs, the number of layers, the number of elements per layer, the connections between the layers, the summation, transfer, and training functions, and even the initial weights themselves. Those changes required to create a successful network constitute a process wherein the "art" of neural networking occurs.

#### ***1.4.2.2 Unsupervised Training (Adaptive Training)***

The other type of training is called unsupervised training. In unsupervised training, the network is provided with inputs without desired outputs. The system

itself must then decide what features it will use to group the input data. This is often referred to as self-organization or adaptation.

At the present time, unsupervised learning is not well understood. This adaptation to the environment is the promise which would enable science fiction types of robots to continually learn on their own as they encounter new situations and new environments. Life is filled with situations where exact training sets do not exist. Some of these situations involve military action where new combat techniques and new weapons might be encountered. Because of this unexpected aspect to life and the human desire to be prepared, there continues to be research into, and hope for, this field. Yet, at the present time, the vast bulk of neural network work is in systems with supervised learning. Supervised learning is achieving results.

One of the leading researchers into unsupervised learning is Tuevo Kohonen, an electrical engineer at the Helsinki University of Technology. He has developed a self-organizing network, sometimes called an auto-associator that learns without the benefit of knowing the right answer. It is an unusual looking network in that it contains one single layer with many connections. The weights for those connections have to be initialized and the inputs have to be normalized. The neurons are set up to compete in a winner-take-all fashion.

Kohonen continues his research into networks that are structured differently than standard, feedforward, back-propagation approaches. Kohonen's work deals with the grouping of neurons into fields. Neurons within a field are "topologically ordered." Topology is a branch of mathematics that studies how to map from one space to another without changing the geometric configuration. The three-dimensional groupings often found in mammalian brains are an example of topological ordering.

Kohonen has pointed out that the lack of topology in neural network models make today's neural networks just simple abstractions of the real neural networks within the brain. As this research continues, more powerful self learning networks may become possible. But currently, this field remains one that is still in the laboratory.

## 1.5 Artificial Neural Networks versus Traditional Computing

The characteristics that can be compared between the Artificial Neural Networks and the traditional computers are shown in table (2).

CHARACTERISTICS	TRADITIONAL COMPUTING (including Expert Systems)	ARTIFICIAL NEURAL NETWORKS
Processing style Functions	<ul style="list-style-type: none"> <li>-Sequential</li> <li>-Logically (left brained)</li> <li>-Via Rules Concepts</li> <li>-Calculations</li> </ul>	<ul style="list-style-type: none"> <li>-Parallel</li> <li>-Gestalt (right brained)</li> <li>-Via Images</li> <li>-Pictures</li> <li>-Controls</li> </ul>
Learning Method Applications	<ul style="list-style-type: none"> <li>-By rules (didactically)</li> <li>-Accounting</li> <li>-Word processing</li> <li>-Math inventory</li> <li>-Digital communications</li> </ul>	<ul style="list-style-type: none"> <li>-By example (Socratic ally)</li> <li>-Sensor processing</li> <li>-Speech recognition</li> <li>-Pattern recognition</li> <li>-Text recognition</li> </ul>

Table (2): ANN vs. traditional computing

## 1.6 The Artificial Neural Network Applications

When a concept leaves the academic environment and is thrown into the harsher world of users who simply want to get a job done, then the artificial neural network is to do the job. Although many networks now being designed are quite accurate, but still leave bad taste for those users who expect the computer to



give them absolute solutions. These networks might be 85% to 90% accurate. Unfortunately, only few applications tolerate that level of error.

There are various possibilities where neural networks might offer solutions, possibilities such as language processing, character recognition, image compression, pattern recognition, and pattern classification.

### ***1.6.1 Banking and Financial***

Loan approval is one of those few applications. The financial institutions gain more profits by making less bad loan decisions. And if we are talking about 90% accurate systems which can be achieved by artificial neural network, this will be an excellent improvement for the traditional used systems as proven by some banks and credit card companies.

### ***1.6.2 Language Processing***

Language processing has a wide variety of applications. These applications include text-to-speech conversion, auditory input for machines, automatic language translation, secure voice keyed locks.

Many companies, universities and research centers working these days on artificial neural network could help in responding to voice commands. The economic rewards of such approach will be just a gold mine. And if this approach was built on a chip million of these chips could be sold.

### ***1.6.3 Character Recognition***

Character recognition is another area in which neural networks are providing solutions. Some of these solutions are beyond simply academic curiosities. Many companies markets a neural network based product that can



recognize hand printed characters through a scanner. This product is 98% to 99% accurate for numbers, a little less for alphabetical characters. Several vendors are saying they are close to commercial products that can scan pages.

#### **1.6.4 *Image (data) Compression***

A number of studies have been done proving that neural networks can do real-time compression and decompression of data. These networks are auto associative in that they can reduce eight bits of data to three and then reverse that process upon restructuring to eight bits again. However, they are not lossless. Because of this losing of bits they do not favorably compete with more traditional methods.

#### **1.6.5 *Pattern Recognition***

Recently, a number of pattern recognition applications have been done. One application on how a physician had trained a neural network on data collected in emergency rooms from people who felt that they were experiencing a heart attack to provide a probability of a real heart attack versus a false alarm.

Another application involves the grading of rare coins. Digitized images from a digital camera are fed into a neural network. These images include several angles of the front and back. These images are then compared against known patterns which represent the various grades for a coin. This system has enabled a quick evaluation for about \$15 as opposed to the standard three-person evaluation which costs \$200.

A neural network is now being used in the scanning of smears. This network is trying to do a better job at reading the smears than can the average lab technician. A missed diagnosis is a too common problem throughout this industry. In many cases, a professional must perceive patterns from noise, such as

*identifying a fracture from an X-ray or cancer from a X-ray "shadow."* Neural networks promise, particularly when faster hardware becomes available, help in many areas of the medical profession where data is hard to read.

Recognition neural networks are being used in a field known as quality control. A number of automated quality applications are now in use. These applications are designed to find that one in a hundred or one in a thousand part that is defective. Human inspectors become fatigued or distracted. Systems now evaluate solder joints, welds, cuttings, and glue applications. One car manufacturer is now even prototyping a system which evaluates the color of paints. This system digitizes pictures of new batches of paint to determine if they are the right shades.

## **1.7 Neural networks structure**

### **1.7.1 *Feedforward-Back Propagation Neural Network.***

This is the most popular network that used more than all other types of networks. It is used in many different types of applications, that because this back-propagation architecture is the most effective, and easy to learn model for complex, multi layered network.

The typical back-propagation network has an input layer, an output layer, and at least one hidden layer. There is no theoretical limit on the number of hidden layers but typically there is just one or two. Some work has been done which indicates that a minimum of four layers (three hidden layers plus an output layer) are required to solve problems of any complexity. Each layer is fully connected to the succeeding layer, as shown in Figure (7).

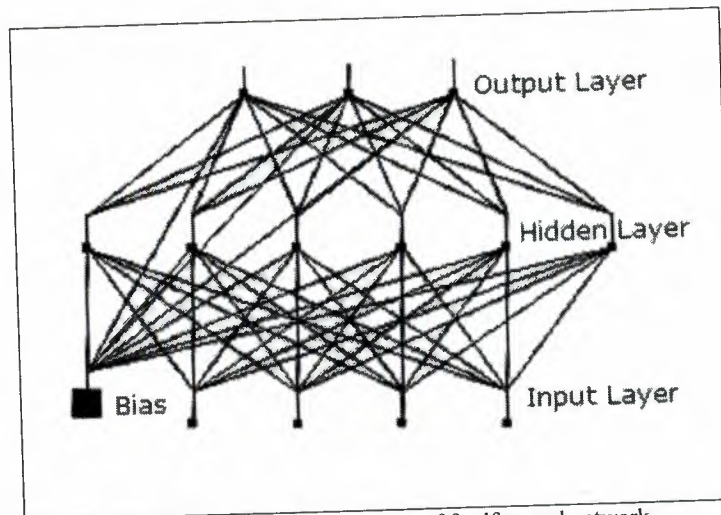


Figure (7): The architecture of feedforward network.

The input and output layers indicate the flow of information during recall. Recall is the process of putting input data into a trained network and receiving the answer. Back-propagation is not used during recall, but only when the network is learning a training set.

Since our research is concerned with the Neural Networks that are capable of classifying patterns, the discussion of such Neural Architecture will be discussed later.

The process of choosing a suitable architecture, for a specific application depends on many parameters, but for simplicity table (3) may help the designers.



Network Type	Networks	Use for Network
Prediction	<ul style="list-style-type: none"> <li>-Back-propagation</li> <li>-Delta Bar Delta</li> <li>-Extended Delta Bar Delta</li> <li>-Directed Random Search</li> <li>-Higher Order Neural Networks</li> <li>-Self-organizing map into Back-propagation</li> </ul>	Use input values to predict some output (e.g. pick the best stocks in the market, predict weather, identify people with cancer risks etc.)
Classification	<ul style="list-style-type: none"> <li>-Learning Vector Quantization</li> <li>-Counter-propagation</li> <li>-Probabilistic Neural Networks</li> <li>-Competitive Layer</li> </ul>	Use input values to determine the classification (e.g. is the input the letter A, is the blob of video data a plane and what kind of plane is it)
Data Association	<ul style="list-style-type: none"> <li>-Hopfield</li> <li>-Boltzmann Machine</li> <li>-Hamming Network</li> <li>-Bidirectional associative Memory</li> <li>-Spation-temporal Pattern Recognition</li> </ul>	Like Classification but it also recognizes data that contains errors (e.g. not only identify the characters that were scanned but identify when the scanner isn't working properly)
Data Conceptualization	<ul style="list-style-type: none"> <li>-Adaptive Resonance Network</li> <li>-Self Organizing Map</li> </ul>	Analyze the inputs so that grouping relationships can be inferred (e.g. extract from a database the names of those most likely to buy a particular product)
Data Filtering	<ul style="list-style-type: none"> <li>-Recirculation</li> </ul>	Smooth an input signal (e.g. take the noise out of a telephone signal)

Table (3): Network Selector Table



## 1.8 summary

This chapter shows some information about artificial neural network which based on the neural structure of the brain, the operation mode and training mode of artificial neural network, comparison between artificial neural network versus traditional computing, its also provide the process of choosing the suitable architecture of artificial neural network through the network selector table which may help the designer and it supports some fields of artificial neural networks and where it can be used in the real world.

## **Chapter TWO**

### **Classification Methods**

#### **2.1 Overview**

Artificial neural networks are undergoing the change that occurs when a concept leaves the academic environment and is thrown into the world of users who simply want to get a job done. Many of the networks now being designed are statistically quite accurate. These networks might be 85% to 90% accurate. Unfortunately, few applications tolerate that level of error. And the selected application of this research is one of these applications.

Since the traditional methods of classification have the same level of accuracy and even worse for some cases, the use of artificial neural networks is applicable here.

In this chapter we will introduce methods of classification like statistical method classification and classification using artificial neural networks

#### **2.2 Statistical Methods:**

When mention the classification processes the statistical approaches are the basic methods in use, and these statistical methods are defined as follow;

##### **2.2.1 *Maximum likelihood Classification***

Maximum likelihood Classification is a statistical decision criterion to assist in the classification; images are assigned to the class of highest probability.

The maximum likelihood classifier uses mean, standard deviations and covariance matrices in classification.

The maximum likelihood classifier is considered to be slow due to extra computations. The accuracy of this method depends on the selection of the input data if it had a Gaussian distribution and this is not always safe.

### **2.2.2 *Minimum distance Classification***

Minimum distance classifies images on a database file using a set of classes. In this criterion only the mean vector in each image data is used. Other data, such as standard deviations and covariance matrices, are ignored (though the maximum likelihood classifier uses this).

The statistical methods needs a lot of computations which results in time consuming, and makes these methods inapplicable for large scale classification. So that another approach had to make this process faster; hence the artificial neural network approaches appeared

## **2.3 Classification using Artificial Neural Network**

One of the most important applications of the Neural Network is classification. A network that can classify could be used in the medical industry to process both lab results and doctor-recorded patient symptoms to determine the most likely disease.

### **2.3.1 *Competitive Neural Network***

The Competitive Neural Network uses the unsupervised training algorithm, where the network is not supplied with the target output and this

property results a low level of accuracy and it makes the process of classification uncontrollable.

### 2.3.1.1 Basic Operation of Competitive Layer

The architecture of a competitive network is shown in figure (8).

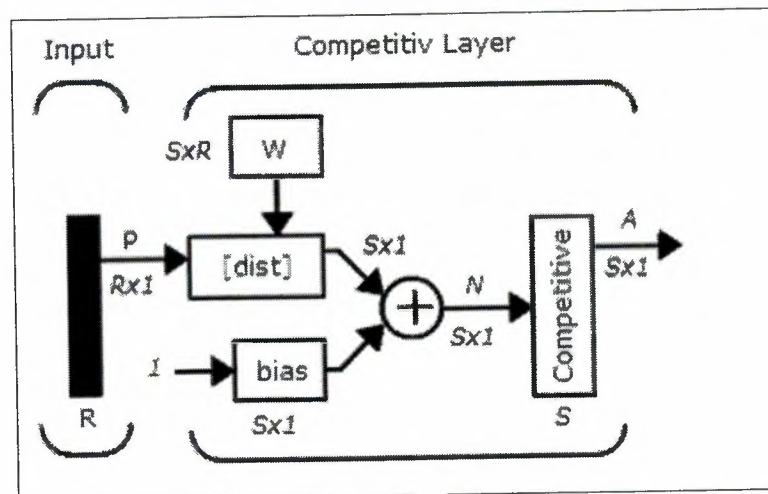


Figure (8): Architecture of competitive network.

The [dist] (distance) box in this figure accepts the input vector ( $P$ ) and the input weight matrix ( $W$ ), and produces a vector having ( $S$ ) elements. The elements in vector ( $S$ ) are the negative of the distances between the input vector and the vectors formed from the rows of the input weight matrix ( $W$ ).

The net input ( $N$ ) of a competitive layer is computed by finding the negative distance between input vector ( $P$ ) and the weight vectors and adding the biases (bias). If all biases are zero, the maximum net input a neuron can have is (0). This occurs only when the input vector  $p$  equals that neuron's weight vector.

The competitive transfer function accepts a net input vector for a layer and returns neuron outputs of (0) for all neurons except for the winner, the neuron associated with the most positive element of net input ( $N$ ). The winner's output is (1).



If all biases are (0), then the neuron whose weight vector is closest to the input vector has the least negative net input and, therefore, wins the competition to output a (1).

### **2.3.1.2 Source of error in the competitive layer network**

Some neurons may not always get *allocated*. In other words, some neuron weight vectors may start out far from any input vectors and never win the competition, no matter how long the training is continued. The result is that their weights do not get to learn and they never win. These unfortunate neurons, referred to as dead neurons, never perform a useful function.

### **2.3.1.3 Bias Learning Rule (*Learncon*)**

To stop the above problem in competitive layer network bias learning rule is used, where biases are used to give neurons that only win the competition rarely (if ever) an advantage over neurons that win often. A positive bias, added to the negative distance, makes a distant neuron more likely to win.

To do this job a running average of neuron outputs is kept. It is equivalent to the percentages of times each output is 1. This average is used to update the biases with the learning function 'learncon' so that the biases of frequently active neurons will get smaller, and biases of infrequently active neurons will get larger. The result is that biases of neurons that haven't responded very frequently will increase versus biases of neurons that have responded frequently.

As the biases of infrequently active neurons increase, the input space to which that neuron responds increases. As that input space increases, the infrequently active neuron responds and moves toward more input vectors. Eventually the neuron will respond to an equal number of vectors as other neurons.

- **Some advantages of the Bias Learning Rule:**

- If a neuron never wins a competition because its weights are far from any of the input vectors, its bias will eventually get large enough so that it will be able to win. When this happens, it will move toward some group of input vectors. Once the neuron's weights have moved into a group of input vectors and the neuron is winning consistently, its bias will decrease to 0. Thus, the problem of dead neurons is resolved.
- The biases force each neuron to classify roughly the same percentage of input vectors. Thus, if a region of the input space is associated with a larger number of input vectors than another region, the more densely filled region will attract more neurons and be classified into smaller subsection.

### **2.3.2 Learning Vector Quantization (LVQ)**

This network topology was originally suggested by Tuevo Kohonen in the mid of 80's, after his original work in self-organizing maps. Both LVQ network and self-organizing maps are based on the Kohonen layer, which is capable of sorting items into appropriate categories of similar objects. Specifically, LVQ is an artificial neural network model used both for classification and image segmentation problems.

#### **2.3.2.1 Architecture of Learning Vector Quantization Network**

The architecture of a competitive network is shown in figure (9).

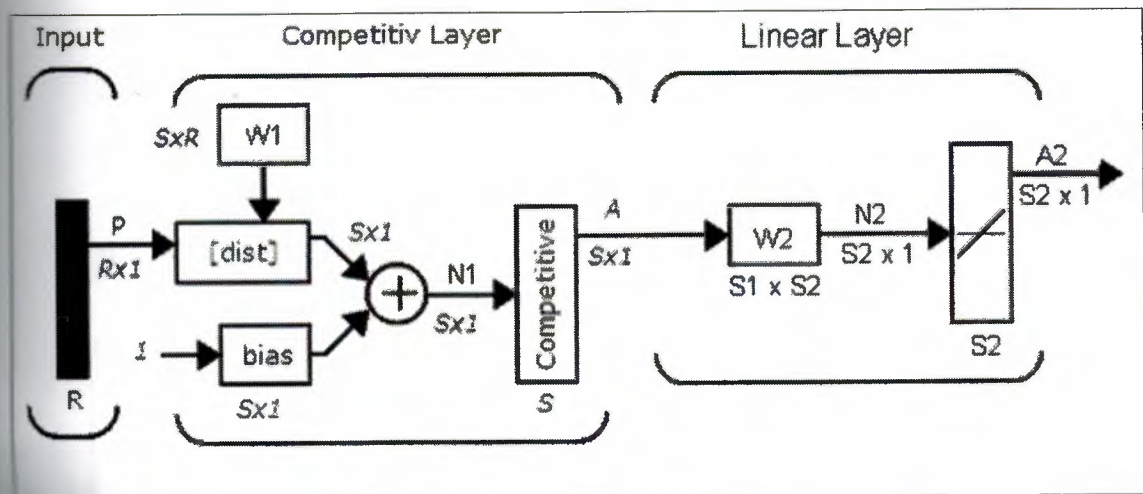


Figure (9): Architecture of LVQ network.

Learning vector quantization (LVQ) network has a first competitive layer and a second linear layer. LVQ is a method for training competitive layers in a supervised manner. A competitive layer automatically learns to classify input vectors. However, the classes that the competitive layer finds are dependent only on the distance between input vectors. If two input vectors are very similar, the competitive layer probably will put them in the same class. Where there is no mechanism in a strictly competitive layer design to say whether or not any two input vectors are in the same class or different classes.

The linear layer transforms the competitive layer's classes into target classifications defined by the user. We refer to the classes learned by the competitive layer as *subclasses* and the classes of the linear layer as *target classes*. Both the competitive and linear layers have one neuron per (sub or target) class. Thus, the competitive layer can learn up to  $S1$  subclasses. These, in turn, are combined by the linear layer to form  $S2$  target classes. ( $S1$  is always larger than  $S2$ .)

Topologically, the network contains an input layer, a single Kohonen layer and an output layer. An example network is shown in Figure (10). The output layer has as many processing elements as there are distinct categories, or classes. The Kohonen layer has a number of processing elements grouped for each of



these classes. The number of processing elements per class depends upon the complexity of the input-output relationship. Usually, each class will have the same number of elements throughout the layer. It is the Kohonen layer that learns and performs relational classifications with the aid of a training set. This network uses supervised learning rules.

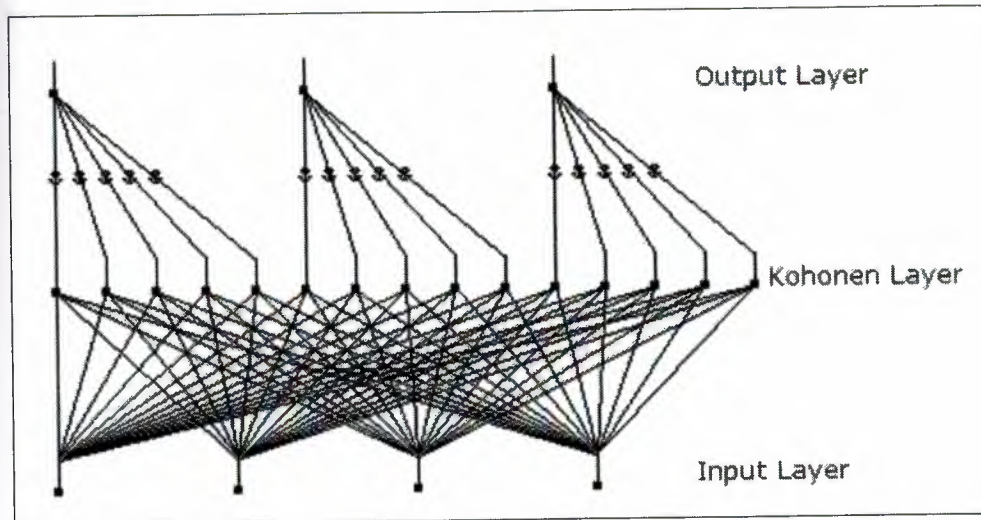


Figure (10): An Example Learning Vector Quantization Network.

### 2.3.2.2 Training of LVQ Networks

In the training mode, this supervised network uses the Kohonen layer such that the distance of a training vector to each processing element is computed and the nearest processing element is declared the winner. There is only one winner for the whole layer. The winner will enable only one output processing element to fire, announcing the class or category the input vector belonged to. If the winning element is in the expected class of the training vector, it is reinforced toward the training vector. If the winning element is not in the class of the training vector, the connection weights entering the processing element are moved away from the training vector. This latter operation is referred to as repulsion. During this training process, individual processing elements assigned to a particular class migrate to the region associated with their specific class.



During the recall mode, the distance of an input vector to each processing element is computed and again the nearest element is declared the winner. That in turn generates one output, signifying a particular class found by the network.

### **2.3.2.3 Drawbacks of LVQ Networks**

There are some shortcomings with the Learning Vector Quantization architecture. Obviously, for complex classification problems with similar objects or input vectors, the network requires a large Kohonen layer with many processing elements per class. This can be overcome with selectively better choices for, or higher-order representation of, the input parameters.

The simple form of the Learning Vector Quantization network suffers from the defect that some processing elements tend to win too often while others, in effect, do nothing. This particularly happens when the processing elements begin far from the training vectors. Here, some elements are drawn in close very quickly and the others remain permanently far away. To alleviate this problem, a conscience mechanism is added so that a processing element which wins too often develops a "guilty conscience" and is penalized. The actual conscience mechanism is a distance bias which is added to each processing element. This distance bias is proportional to the difference between the win frequency of an element and the average processing element win frequency. As the network progresses along its learning curve, this bias proportionality factors needs to be decreased.

## **2.4 Summary**

This chapter introduces the classification method which is divided into two parts first part was statistical method of classification which is divided into maximum likelihood classification and minimum distance classification. The second part was about classification using an artificial neural network which was

also divided into competitive neural network and learning vector quantization network. Also we will see in the appendix a description of the dist function which had been used in the competitive layer architecture and in the learning vector quantization network architecture.

## **Chapter 3**

### **Introduction to Wavelet Analysis**

#### **3.1 Overview**

In general, signals in their raw form are time-amplitude representations. These time-domain signals are often needed to be transformed into other domains like frequency domain, time-frequency domain, etc., for analysis and processing. Transformation of signals helps in identifying distinct information which might otherwise be hidden in the original signal. Depending on the application, the transformation technique is chosen, and each technique has its advantages and disadvantages.

#### **3.2 The need of wavelet transformation**

The Digital Signal Processing (DSP) applications depend, in most cases, on the frequency content of the signal; by using the Fourier Transform we can obtain the frequency spectrum of any signal. But the Fourier Transform is only suitable for stationary signal, signals whose frequency content does not change with time. The Fourier Transform, while it tells how much of each frequency exists in the signal, it does not tell at which time these frequency components occur.

Signals such as image and speech have different characteristics at different time or space, i.e., they are non-stationary. Most of the biological signals too, such as, Electrocardiogram, Electromyography, etc., are non-stationary. To analyze these signals, both frequency and time information are needed simultaneously, i.e., a time-frequency representation of the signal is needed.

### 3.3 The Drawback of Fourier Transform

Fourier analysis has a serious drawback. In transforming to the frequency domain, time information is lost. When looking at a Fourier transform of a signal, it is impossible to tell when a particular event took place. If the signal properties do not change much over time -- that is, if it is what is called a stationary signal -- this drawback isn't very important. However, most interesting signals contain numerous non-stationary or transitory characteristics: drift, trends, abrupt changes, and beginnings and ends of events. These characteristics are often the most important part of the signal, and Fourier analysis is not suited to detecting them.

### 3.4 The solution of the limitation of the Fourier Transform

In an effort to correct this deficiency, Dennis Gabor (1946) adapted the Fourier transform to analyze only a small section of the signal at a time -- a technique called windowing the signal --. Gabor's adaptation, called the Short-Time Fourier Transform (STFT), maps a signal into a two-dimensional function of time and frequency. The STFT represents a sort of compromise between the time and frequency-based views of a signal. It provides some information about both when and at what frequencies a signal event occurs. However, you can only obtain this information with limited precision, and that precision is determined by the size of the window.

- ***The drawback of the (STFT):***

While the STFT compromise between time and frequency information can be useful, the drawback is that once you choose a particular size for the time window, that window is the same for all frequencies. Many signals require a more



flexible approach, where we can vary the window size to determine more accurately either time or frequency.

### **3.5 The appearance of Wavelet analysis**

The Wavelet Transform solves the drawback of the STFT to a certain extent, since the Wavelet Transform uses short windows at high frequencies and long windows at low frequencies. This results in multi resolution analysis by which the signal is analyzed with different resolutions at different frequencies, i.e., both frequency resolution and time resolution vary in the time-frequency plane.

In Wavelet Transform, as frequency increases, the time resolution increases; likewise, as frequency decreases, the frequency resolution increases. Thus, a certain high frequency component can be located more accurately in time than a low frequency component and a low frequency component can be located more accurately in frequency compared to a high frequency component.

The Wavelet Transform was developed independently in applied mathematics and signal processing. It is gradually substituting other transforms in some signal processing applications. For example, previously, the STFT was extensively used in speech signal processing, and Discrete Cosine Transform (DCT) was used for image compression. But now, the Wavelet Transform is substituting these, due to its better resolution properties and high compression capabilities.

The properties of Wavelet Transform allow it to be successfully applied to non-stationary signals for analysis and processing, e.g., speech and image processing, data compression, communications, etc.

A wave is an oscillating function of time or space and is periodic. In contrast, wavelets are localized waves as shown in figure (11). They have their energy concentrated in time or space and are suited to analysis of transient signals. While Fourier Transform and STFT use waves to analyze signals, the Wavelet Transform uses wavelets of finite energy.

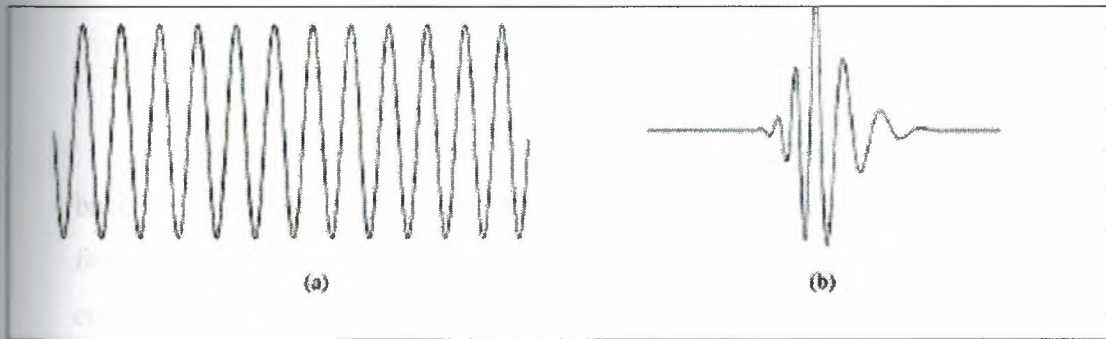


Figure (11): Demonstration of (a) a Wave and (b) a Wavelet.

### 3.6 Wavelet Computing

The wavelet analysis is done similar to the STFT analysis. The signal to be analyzed is multiplied with a wavelet function just as it is multiplied with a window function in STFT, and then the transform is computed for each segment generated.

However, unlike STFT, in Wavelet Transform, the width of the wavelet function changes with each spectral component. The Wavelet Transform, at high frequencies, gives good time resolution and poor frequency resolution, while at low frequencies; the Wavelet Transform gives good frequency resolution and poor time resolution.

### 3.6.1 The Continuous Wavelet Transform and the wavelet series

The Continuous Wavelet Transform (CWT) is provided by the following equation:

$$X_{WT}(\tau, s) = \frac{1}{\sqrt{|s|}} \int x(t) \cdot \psi^* \left( \frac{t - \tau}{s} \right) dt$$

Where  $x(t)$  is the signal to be analyzed.  $\psi(t)$  is the mother wavelet or the basis function. All the wavelet functions used in the transformation are derived from the mother wavelet through translation (shifting) and scaling (dilation or compression).

The mother wavelet used to generate all the basis functions is designed based on some desired characteristics associated with that function. The translation parameter ( $\tau$ ) relates to the location of the wavelet function as it is shifted through the signal. Thus, it corresponds to the time information in the Wavelet Transform. The scale parameter ( $s$ ) is defined as  $|1/\text{frequency}|$  and corresponds to frequency information. Scaling either dilates (expands) or compresses a signal. Large scales (low frequencies) dilate the signal and provide detailed information hidden in the signal, while small scales (high frequencies) compress the signal and provide global information about the signal. Notice that the Wavelet Transform merely performs the convolution operation of the signal and the basis function. The above analysis becomes very useful as in most practical applications, high frequencies (low scales) do not last for a long duration, but instead, appear as short bursts, while low frequencies (high scales) usually last for entire duration of the signal.

The Wavelet Series is obtained by discretizing CWT. This aids in computation of CWT using computers and is obtained by sampling the time-scale plane. The sampling rate can be changed accordingly with scale change without



violating the Nyquist criterion. Nyquist criterion states that, the minimum sampling rate that allows reconstruction of the original signal is  $(2\omega)$  radians, where  $(\omega)$  is the highest frequency in the signal. Therefore, as the scale goes higher (lower frequencies), the sampling rate can be decreased thus reducing the number of computations.

### **3.6.2 The Discrete Wavelet Transform**

The Wavelet Series is just a sampled version of CWT and its computation may consume significant amount of time and resources, depending on the resolution required.

The Discrete Wavelet Transform (DWT), which is based on sub-band coding, is found to yield a fast computation of Wavelet Transform. It is easy to implement and reduces the computation time and resources required.

The foundations of DWT go back to 1976 when techniques to decompose discrete time signals were devised. Similar work was done in speech signal coding which was named as sub-band coding. In 1983, a technique similar to sub-band coding was developed which was named pyramidal coding. Later many improvements were made to these coding schemes which resulted in efficient multi-resolution analysis schemes.

In CWT, the signals are analyzed using a set of basis functions which relate to each other by simple scaling and translation. In the case of DWT, a time-scale representation of the digital signal is obtained using digital filtering techniques. The signal to be analyzed is passed through filters with different cutoff frequencies at different scales.



### 3.6.3 DWT and Filter Banks

Filters are one of the most widely used signal processing functions. Wavelets can be realized by iteration of filters with rescaling. The resolution of the signal, which is a measure of the amount of detail information in the signal, is determined by the filtering operations, and the scale is determined by up-sampling and down-sampling (sub-sampling) operations.

The DWT is computed by successive low-pass and high-pass filtering of the discrete time-domain signal as shown in figure (12). This is called the Mallat algorithm or Mallat-tree decomposition. Its significance is in the manner it connects the continuous time multi-resolution to discrete-time filters. In the figure, the signal is denoted by the sequence  $x[n]$ , where  $(n)$  is an integer. The low pass filter is denoted by  $G_0$  while the high pass filter is denoted by  $H_0$ . At each level, the high-pass filter produces detail information;  $d[n]$ , while the low pass filter associated with scaling function produces coarse approximations,  $a[n]$ .

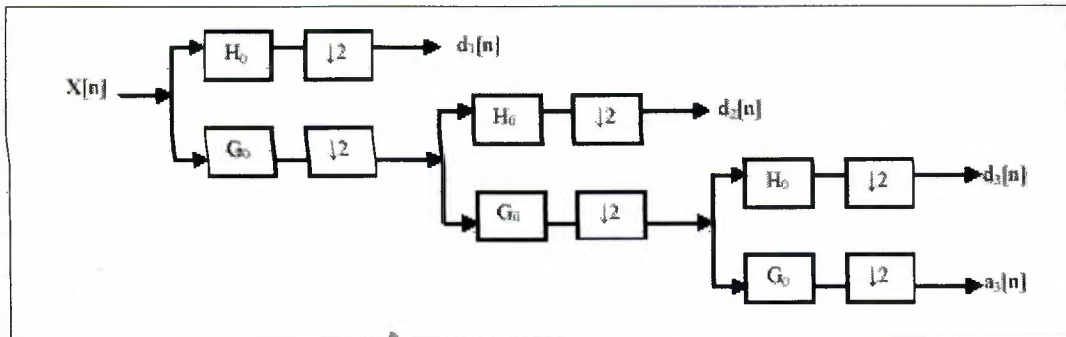


Figure (12): Three-level wavelet decomposition tree.

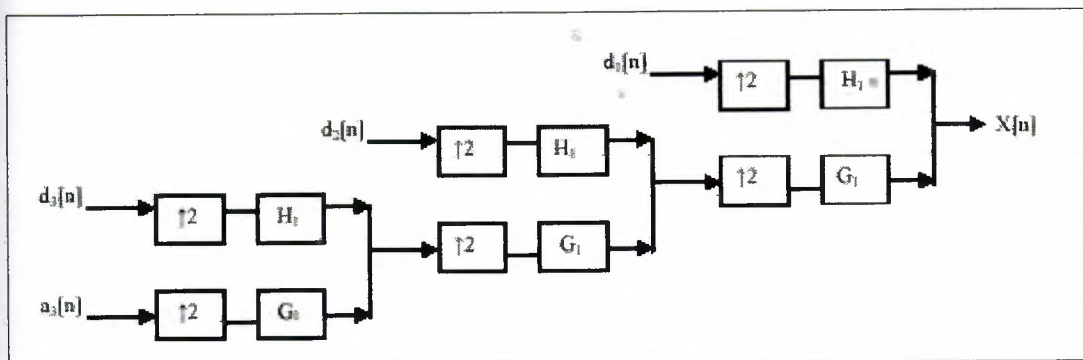
At each decomposition level, the half band filters produce signals spanning only half of the frequency band. This doubles the frequency resolution as the uncertainty in frequency is reduced by half. In accordance with Nyquist's rule if the original signal has a highest frequency of  $(\omega)$ , which requires a sampling frequency of  $(2\omega)$  radians, then it now has a highest frequency of  $(\omega/2)$

radians. It can now be sampled at a frequency of  $(\omega)$  radians thus discarding half the samples with no loss of information.

This decimation by 2 halves the time resolution as the entire signal is now represented by only half the number of samples. Thus, while the half band low pass filtering removes half of the frequencies and thus halves the resolution, the decimation by 2 doubles the scale.

With this approach, the time resolution becomes arbitrarily good at high frequencies, while the frequency resolution becomes arbitrarily good at low frequencies. The filtering and decimation process is continued until the desired level is reached. The maximum number of levels depends on the length of the signal. The DWT of the original signal is then obtained by concatenating all the coefficients,  $a[n]$  and  $d[n]$ , starting from the last level of decomposition.

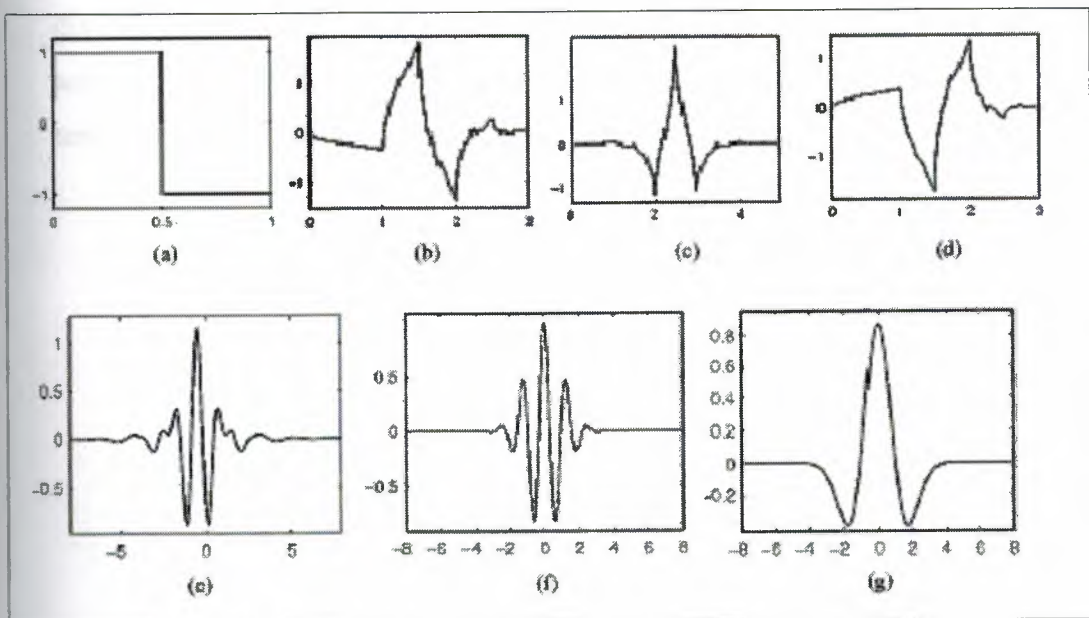
Figure (13) shows the reconstruction of the original signal from the wavelet coefficients. Basically, the reconstruction is the reverse process of decomposition. The approximation and detail coefficients at every level are up-sampled by two, passed through the low-pass and high-pass synthesis filters and then added. This process is continued through the same number of levels as in the decomposition process to obtain the original signal. The Mallat algorithm works equally well if the analysis filters,  $G_0$  and  $H_0$ , are exchanged with the synthesis filters,  $G_1$  and  $H_1$ .



Figure(13): Three-level wavelet reconstruction tree.

### 3.7 Wavelet Families

There are a number of basis functions that can be used as the mother wavelet for Wavelet Transformation. Since the mother wavelet produces all wavelet functions used in the transformation through translation and scaling, it determines the characteristics of the resulting Wavelet Transform. Therefore, the details of the particular application should be taken into account and the appropriate mother wavelet should be chosen in order to use the Wavelet Transform effectively.



Figure(14): Wavelet families (a) Haar (b) Daubechies4 (c) Coiflet1 (d) Symlet2  
(e) Meyer (f) Morlet (g) Mexican Hat.

Figure (14) illustrates some of the commonly used wavelet functions. Haar wavelet is one of the oldest and simplest wavelet. Therefore, any discussion of wavelets starts with the Haar wavelet. Daubechies wavelets are the most popular wavelets. They represent the foundations of wavelet signal processing and are used in numerous applications. These are also called Maxflat wavelets as their frequency responses have maximum flatness at frequencies 0 and  $\pi$ . This is a very desirable property in some applications.



The utilizing of DWT using the MATLAB will be discussed later when we use it, in performance improvement of the research results.

### **3.8 summary**

This chapter discusses the wavelet analysis, its needed, the drawback of Fourier transform and its solution with STFT then the problem with STFT, also it discussed the appearance of wavelet analysis and its better resolution properties and its high compression capabilities, then it discussed the wavelet computing which divided into three parts: the continuous wavelet transform and the wavelet series, discrete wavelet transform (DWT) and DWT and filter banks. And finally it show some of basis functions that can be used as the mother wavelet for wavelet transformation which related to wavelet family's part.



## Chapter 4

### Design objective and Preprocessing

#### 4.1 Overview

In order to show the efficiency of ANN in pattern classification field, an example had to be applied through this research. So that we used a set of images, to classify them into three different categories. Those images were three different types of leave those are differ in shape. As shown in the following figures (15), (16), (17).

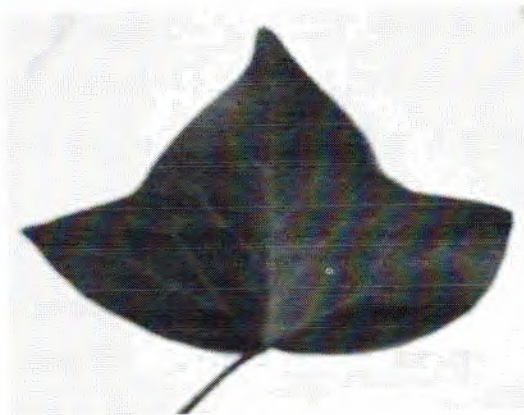
The data set (training set) we have consists of 60 images of each class, we will use 59 image of each class for training and the simulation (the test) will involve the whole pictures



figure(15): the first class example



figure(16): the second class example



figure(17): the third class example

## 4.2 Image characteristics

The images we had were all from the same size (896 x 592) and they all were colored (RGB images), so we had to apply some preprocessing on those images in order to have an adequate input to the ANN, since the input layer of the network to be designed has not an open range of inputs, we had to convert the given images into a form that is applicable for this purpose.

So we had to convert the image into numerical coefficients, which is the most applicable input of an ANN, and this can be done easily by the MATLAB code using the following code statements;

```
imread Reads image from graphics file.
image_RGB = imread('FILENAME.FMT');
```

image_RGB	:	is the destination array (matrix).
'FILENAME.FMT '	:	is a string that contains the graphic file name including its extension.

This command reads the image in FILENAME into an array A. If the file contains a grayscale intensity image, then A is a two-dimensional array. If the file contains a true color (RGB) image, then A is a three-dimensional (M-by-N-by-3) array.

The third dimension of the array is the color index, and making it unity will convert the array from colored picture array (three dimensions array) into grayscale picture array (two dimensions array), where the latter is easier to deal with and since the color is not one of the characteristics that the image can be distinguished by because it is a common characteristics of three classes. And this can be done using the following command;

```
image_gray = image_RGB(:,:,1);
```

This command sets the color index of the array to unity for all pixels, thus the array will be converted into two dimensions array which is simply a grayscale picture array.

In order to illustrate the use of the previous commands, the use of another command that restore the array to an image will be useful in this case, and this command is IMSHOW.

```
imshow(A);
```

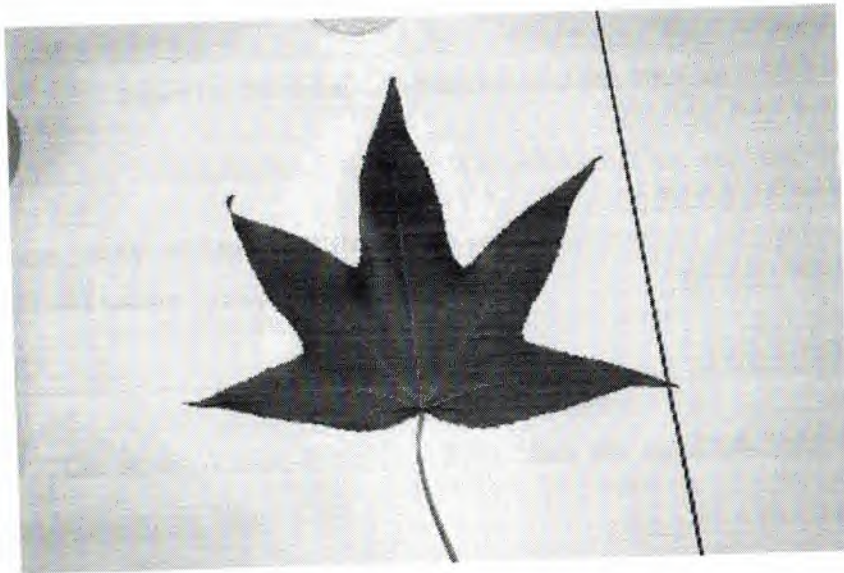
Displays an image with true color mode (RGB) if A was a three dimensions array, and in grayscale mode if A was a two dimensions array. The



following simple example shows the effect of omitting the color index from the three dimension array.

**Example:**

The previous process was applied on one of the given pictures. The image shown in figure (18), is saved as 'LA\_01.jpg' in the workspace of the MATLAB; and it is an RGB (896 x 592) pixels image.



Figure(18): The original image displayed with 50% scale.

```
image_RGB = imread('LA_01.jpg');  
imshow(image_RGB)
```

The array (image\_RGB) size is (592 x 896 x 3), and then the resultant image is true color image as shown in figure (19);





Figure(19): The reconstructed image displayed with 50% scale.

```
image_gray = image_RGB(:,:,1);  
imshow(image_gray)
```

The array A size is (592 x 896), then the resultant image is grayscale image as shown in figure (20);



Figure(20): The reconstructed grayscale image displayed with 50% scale.

The previous example shows that we can decompose the image into a single two dimensions matrix, and any two dimension matrix could be converted into a single column or single row vector, and both are applicable as an input for an ANN.

Converting the (IMREAD) result matrix into a vector will yields a vector of (530,432) elements, which is a very huge number. And such number is impossible to be fed to an ANN as an input, in other words we can initiate an ANN with this number of input neurons, but the training operation will be impossible as will be shown later.

Reducing number of input neurons can be made utilizing many methods, but the question is which one is better and which is applicable in our case.

The first, we can design several networks and each one can work on a segment of the whole matrix, (i.e. divide the matrix into 4 segments), this approach may be applied but it will be weak since our object (the leaf) may not be located at the same location of the image in all images, and this may raise the error rate in the classification process.

Reducing the number of numerical coefficient to a level that is adequate for an ANN use, may be a second solution. But in this case we have to worry about the way that will be used to reduce the number of elements, because it has to be selective and sensitive. Also it has to be efficient. So we need a procedure to reduce the number on the coefficients while keeping the images distinguishable, and here we were directed to the wavelet transformer.

### **4.3 The MTLAB Wavelet Toolbox usage**

The Wavelet toolbox is rich with functions and commands those are so useful in data processing, as was mentioned before the wavelet transformation has

two branches continuous and discrete and in image processing we are involved with the discrete one more precisely we use the two dimension discrete wavelet.

### 4.3.1 The decomposition process

The following MATLAB statement decompose a two dimensions array (matrix) into four different arrays called (The approximate matrix, the horizontal details matrix, the diagonal details matrix and the vertical details matrix)

```
[A1 A2 A3 A4]=dwt2(MATRIX,'db4');
```

A1	:	The approximation image matrix
A2	:	Horizontal details
A3	:	Diagonal details
A4	:	Vertical details
dwt2	:	Computes one level wavelet coefficients.
MATRIX	:	Matrix to be decompose (the gray scale matrix in this case)
'db4'	:	The considered wavelet family.

### 4.3.2 The reconstruction Process

An approximate reconstruction of the matrix (the image) can be held using the approximate matrix as well as we can reconstruct the other details using their matrices using. This can be done simply by using the IMSHOW command, and their still another way to reconstruct the exact image using the four matrices and the inverse discrete wavelet transformation.



```
imshow(A1, [min(min(A1)) max(max(A1))]);
```

A1	:	The approximation image matrix
min(min(A1))	:	Computes the minimum in each row of the matrix to set the ranges.
max(max(A1))	:	Computes the maximum in each row of the matrix to set the ranges.

#### 4.4 Images preprocessing; the use of WAVELET transformer

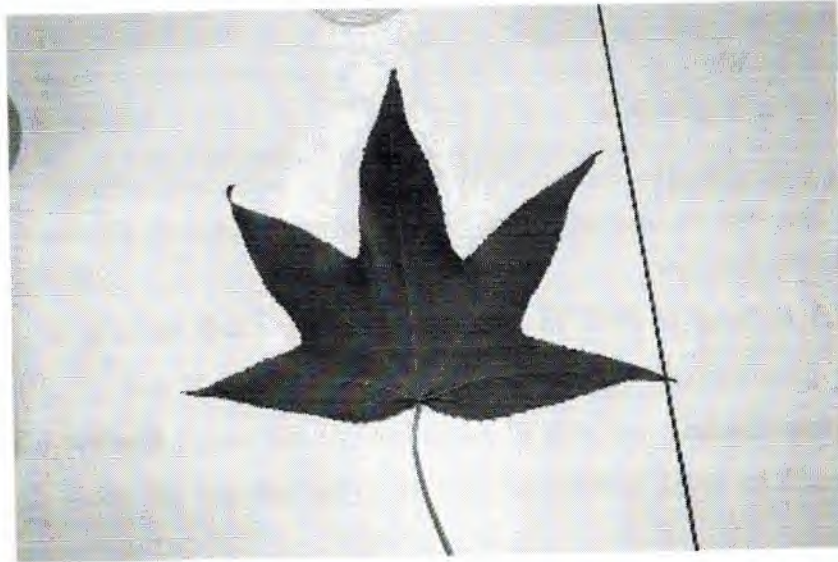
Starting from the truth that the image consists of many frequencies, and knowing that the general features and general shape of the image represent the low frequency components of the image, while the fine details plus the deepest details are represented by the high frequency components. And since we consider the general shape of the leaf, we are interested in the lowest frequency components, and keep in mind that the wavelet transform is nothing but location-frequency mapping. All of this leads us to the wavelet solution.

To understand the use of two dimension wavelet transformation, a real example can be useful to show how it is helpful in our case; lets start with a picture for a leaf of the first class; starting with one level decomposition;

The following code decompose the contents of the matrix which we got using IMREAD command;



```
image_RGB = imread('LA_01.jpg');
image_gray = image_RGB(:,:,1);
[A1 A2 A3 A4]=dwt2(image_gray,'db4');
```



Figure(21): The original image 'LA\_01.jpg' displayed with 50% scale.

The matrix (A1) contains the low frequency coefficients of the original image; and its size is (299 x 451) which is (134,849) elements. We can reconstruct the image that represented by the approximate matrix (A1) to compare it to the original image shown in figure (21). And the statement that reconstructs it; is shown below, and the result is shown in figure (22)

```
imshow(A1,[min(min(A1)) max(max(A1))]);
```

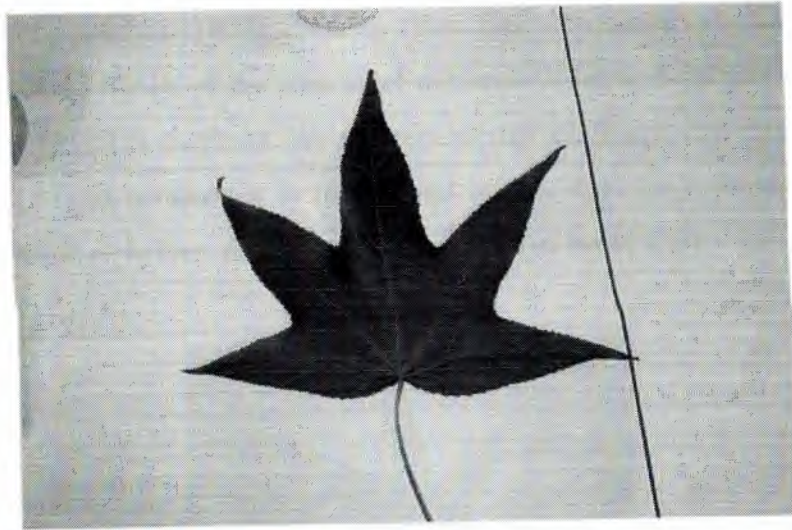


Figure (22): The first level wavelet decomposition result displayed with 100% scale.

It is clear that we had a reduction of 25% at least in the number of coefficients, and if we compared the figure of the reconstructed image to the original we will figure out the reduction in size too. Unfortunately 25% is not enough since the number still too big. So another level of decomposition is recommended.

Now we will decompose the image that reconstructed from matrix (A1), or simply we will apply an extra one level wavelet decomposition on matrix (A1), as shown in the next code segment;

```
[B1 B2 B3 B4]=dwt2(A1,'db4');
```

B1	:	The approximation image matrix of A1.
B2	:	Horizontal details of A1.
B3	:	Diagonal details of A1.
B4	:	Vertical details of A1.

The matrix (B1) contains the low frequency coefficients of the reconstructed image of the first level decomposition; and its size is (153 x 229) which is (35,037) elements. We can reconstruct the image that represented by the matrix (B1) to compare it to its original image. And the reconstruction process can be made as before. And the result is shown in figure (23).



Figure (23): The second level wavelet decomposition result displayed with 100% scale.

The number of matrix B1 shows about another 25% reduction in the image coefficients, and its size shows that too. But (35,037) elements still large to train a neural network with it as will be shown later. And another level of decomposition have to be done again.

The following results show another three levels of decomposition; by the same code used before the third, fourth and fifth levels can be computed and the results were as shown below. These results show that the size of image is reduced five times and still can be classified by human eye, so it may be classified by ANN too (that will be proved in the design).






WAVELET Level	Image reconstructed from the approximate matrix	Matrix size	Number of coefficients
Third		80 x 118	9440
Fourth		43 x 62	2666
Fifth		25 x 34	850

Table (4): The wavelet reduction procedure results

Table (5) shows the effective of wavelet transformation on the size of the resultant matrix and on the reduction of its elements number.

Decomposition Level	Previous No. of Elements	Next No. of Elements	% of Reduction	% of Over all Reduction
First	530,432	134,849	74.52%	74.25 %
Second	134,849	35,037	74.18%	93.40%
Third	35,037	9440	73.06%	98.22%
Fourth	9440	2666	71.76%	99.50%
Fifth	2666	850	68.12%	99.84%

Table (5): The reduction of matrix sizes due to multi level wavelet transformation



The whole process of the five levels decomposition can be summarized by the following two figures (24) and (25);

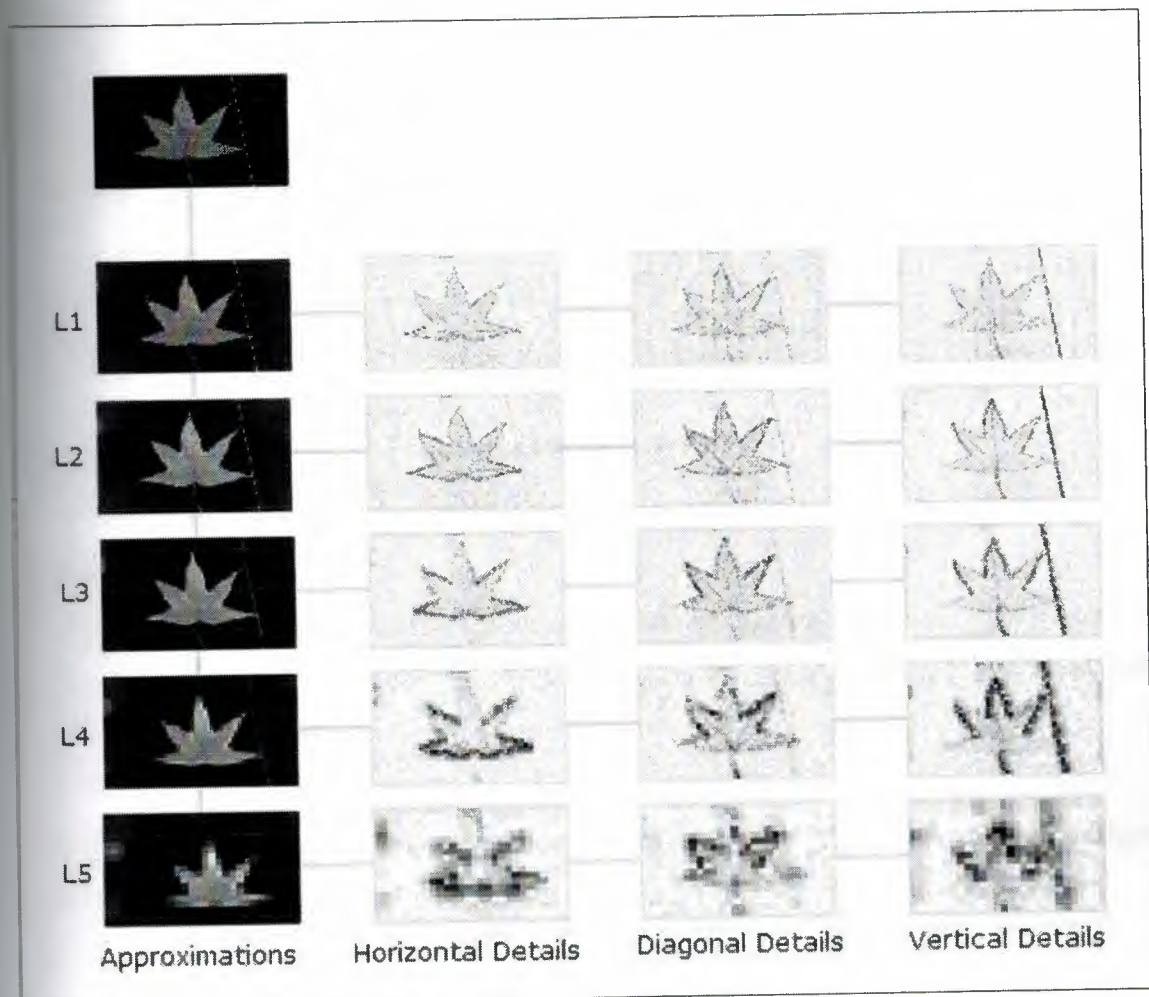


Figure (24): Tree arrangement of five levels wavelet decomposition.

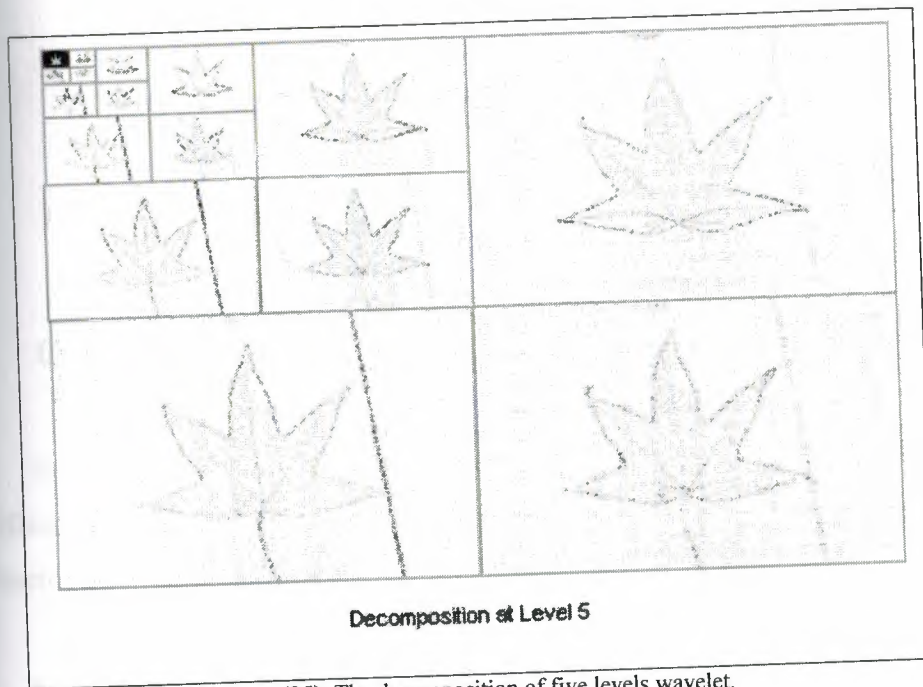


Figure (25): The decomposition of five levels wavelet.

The previous two figures show an arrangement of five levels wavelet decomposition, it is clear that the approximation image of the last level still distinguishable.

Now since we had many input levels to the ANN, the possible designs may be made now, and this is the next topic.

## 4.5 summary

This chapter discussed some MATLAB codes and it show how they work on the selected images with some examples, it used the wavelet transformation which helped in the data processing, but we involved with the two dimension discrete branch of the wavelet transformation and it was applied on the decomposition and reconstruction process. Also it discussed the cause of using the two dimension wavelet transformation through an example using the MATHLAB code.

## **Chapter 5**

### **FeedForward Design**

#### **5.1 Overview**

In order to develop the optimum Neural Network that is capable to do the classification with the minimum acceptable range of error, we try different Network architectures to reach that goal.

One of the most popular, and effective network is the Feedforward architecture which is used in many different types of applications. This architecture has many different topologies and training methods, which make it strong in the nonlinear field of application.

To compare the feedforward performance with the Neural Network which are specially designed for the classification purpose such as Competitive Layer, and Learning vector Quantization (which will discussed later). The next section will discuss the design of the Feedforward network for the classification purpose.

#### **5.2 The FeedForward Construction**

As mentioned in previous sections in this research; the basic structure in the Feedforward Neural Network must have at least one input layer, one hidden layer, and one output layer. The connection between these layers will define the ability and strength of the network.



The design of the Network must satisfy the following conditions:

- Accept the input image that to be classified, in the input layer as numerical coefficients represent the image specifications.
- Do the required calculation in the hidden layer to recognize the image and to know in which class it belongs to.
- Represent a number or index in the output layer which refer to the image class.

The following MATLAB code will illustrate how we can design such network to do the application:

```
net = network;
```

This command constructs a blank network, without any properties, the number of input layers, hidden layers, output layers, and the number of neurons in each layer is still undefined.

```
net.numInputs = 1;
```

This command will define an input layer in our blank network (net), yet the number of the neurons contained in this layer is not defined.

```
net.numLayers = 2;
```

This command defines the layers in the network (hidden, output), one hidden layer, and one output layer.



```
net.inputs{A}.size = n;
```

{A} is the index of the input layer, in our case there is only one input layer so A=1; also this will define the number of neurons in the input layer (n).

Figure (26) illustrates how the network looks like at this point.

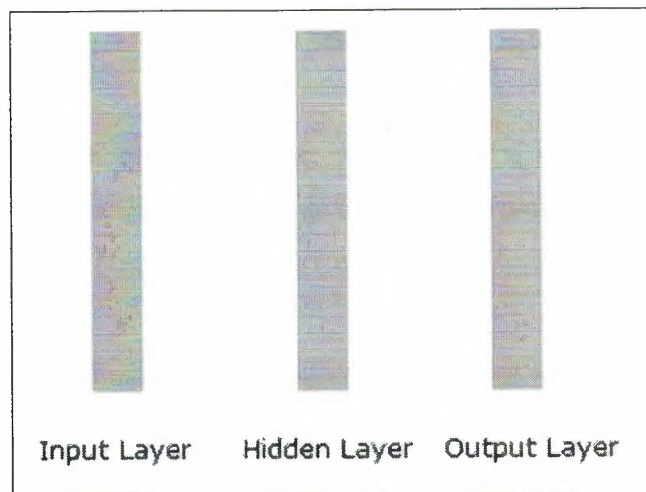


Figure (26): Simple representation of FeedForward ANN.

The following code will construct a FeedForward ANN, this code determined as shortcut in constructing the network, for this purpose;

```
net = newff(PR, [S1 S2...SN1], {TF1 TF2...TFN1}, BTF);
```

PR	:	Rx2 matrix of min and max values for R input elements.
Si	:	Size of ith layer, for N1 layers.
TFi	:	Transfer function of ith layer, default = 'tansig'

BTF : Back propagation network training function, default = 'trainlm'.
---

The number of the neurons in the input layer will specify the number of the numerical coefficients taken from the picture. Although we could define a large number of neurons for the input layer, but that doesn't mean that the design is done, because the training process limits the design process. So a preprocessing (Wavelet Decomposition) on the images may be useful in this case (as will be shown).

### 5.3 FeedForward ANN Design using the original size

Using the original image, which have a size of (896 x 592), without wavelet preprocessing, will lead us to build a neural network with (530432) input neurons, we will build this network to prove the limitation.

The FeedForward ANN utilizes supervised training, so the training set and the desired output (Targets) have to be set. The following code sets these matrices;

```
image_RGB=imread('filename');  
%where 'filename' is the name of the image file.  
  
image_gray=image_RGB(:,:,1);  
%converts the RGB to Grayscale.  
  
image_vector=image_gray(:);  
%converts the matrix to vector.
```

```
training_set(:,1)=image_vector;  
% converts matrix to column vector.
```

Evaluating the previous code for all images (using for loop function)\* will result a matrix that has a number of rows equal to the number of elements of the grayscale image matrix, and number of columns equal to that of the total number of images.

The resultant matrix has a large size (530432 x 177) that the MATLAB doesn't support this large size and an error (OUT OF MEMORY) was reported, so we reduced the number of images in the training set to 60 images 20 of each class, and this was the maximum acceptable number by the MATLAB.

After the evaluating of this MATLAB code, the data matrix (training set) will contain 60 columns, each column represent one image, the first 20 column belongs to the first class, the second 20 columns belongs to the second class, and the third class have the last 20 columns in the matrix.

The design requires a number or index to be represented in the output layer to indicate the class of the input image (desired output), and since we have three different classes, the output layer will contain 3 neurons, each neuron associated with one class, the output of the neuron will be (1) if the input image belong to the class which the neuron is related to, other wise the output will be zero. This will be denoted as the targets matrix.

The target matrix must contain 60 columns, which will represent the output combination for each input image, each column contain 3 rows, the first element represent the first class, which mean that if the input image belongs to the

---

\* The complete code is shown in appendix ()

first class, this element must be one, other wise it will be zero (in the first 20 columns).

The reference for constructing this matrix is the data matrix (training set), these two matrices must be related with each other, meaning that the first 20 column in the target matrix must represent the output of the network while the input is class one image, that's because the first 20 columns of the data matrix represent the first class, for the second class, the second element (row number two) in the next 20 columns of the target matrix must be one, other wise this element must be zero, the third element must be one only in the last 20 columns of the target matrix.

The following MATLAB code will construct the target matrix:

**Class one outputs:**

```
targets(1,1:20)=1;  
targets(1,21:60)=0;
```

**Class two outputs:**

```
targets(2,1:20)=0;  
targets(2,21:40)=1;  
targets(2,41:60)=0;
```

**Class three outputs:**

```
targets(3,1:40)=0;  
targets(3,41:60)=1;
```

Now the network is ready to be constructed, since we have the training set and the desired output, then the training will be held. And the input neurons



ranges must be defined, because the initial weights of the connection between the input layer and the hidden layer depend on these values.

```
ranges=(minmax(training_set));
```

minmax will take the maximum value and the minimum value of the (training\_set) matrix rows, and put these values in the (ranges) matrix.

All the requirements to construct the Neural Network are ready; we just have to put it all in one statement with the proper transfer function between the layers.

In the first trail we will make the number of neurons in the hidden layer equal to the half of the number in the input layer (265216 hidden neurons), and all the transfer function of the hidden layer will be 'tansig' where the transfer function of the output layer will be 'pure linear'.

```
net=newff(ranges,[265216,3],{'tansig','purelin'},'trainlm');
```

This command will create the neural network with all the specification mentioned above in the network description.

To train the network we must define some training parameters in order to get the highest performance.

```
net.trainparam.epochs=1000;
```

Defines the maximum number of times the complete data set may be used for training,

```
net.trainparam.show=100;
```

Defines the time between status reports of the training function.

```
net.trainparam.goal=1e-2;
```

Defines the maximum error accepted in the training process.

Now the network is ready to be trained with the input (training set) and output matrices.

The following MATLAB command will start the training of the network:

```
net=train(net,trainin_set,targets);
```

This command will train the network with the data matrix as its input, and the target matrix as its output.

- **The training result**

As expected the Network could not handle this large number of neurons in the input layer either the hidden layer. So the network failed in completing the training under these conditions, the error reported from the MATLAB command window was (OUT OF MEMORY).

## 5.4 The use of Wavelet transformation

It is obvious that the number of inputs must be reduced, by reducing the size of the images, one of the recommended solutions, as mentioned in previous sections, was the Wavelet Analysis. Performing the wavelet analysis on the images will reduce its size, by separating the high frequency component from the low frequency component and put each group of frequencies in different matrix.

The question that appears now is what size the image should be reduced to, without losing the most important details? Performing the wavelet processing on the images will reduce the size to approximately one fourth, and the result image (approximate image matrix) will still have the main shape and the main details. And this is valid until the fifth level of decomposition as mentioned before.

In this phase we will perform the wavelet analysis on the images once (one level preprocessing), the result matrix after the performing will have the size of (299 x 451) which mean that we reduce the number of the neurons in the input layer to (134850) neuron.

Since the number of elements of each image will be reduced, we can go back to the larger training set which contains the 177 images, and the code that sets the matrix of this training set is attached in appendix().

The construction of this training set did not face the same problem as the previous training set; the MATLAB did not report an (OUT OF MEMORY) error.

A construction of targets matrix (desired output), can be done using the same previous code except it will be for 177 elements, and this code is attached in Appendix ().

The number of neurons in the input layer has to be changed, to meet the new conditions; the following code will change the number of input neurons to 134850:

```
net.inputs{1}.size = 135148;
```

To change the network to the new specification:

```
ranges=(minmax(data));  
net=newff(ranges,[67800,3],{'purelin','purelin'},'trainlm');
```

Now the network is ready to be trained with the input (training set) and output matrices.

```
net=train(net,training_set,targets);
```

▪ **The result of the training:**

The network failed in completing the training, the error reported from the MATLAB command window was (OUT OF MEMORY).



## 5.5 The end results of FeedForward design

Again the number of elements was reduced, to the second, third, fourth, fifth and sixth. But the same error was appearing each time. Hence we concluded that this architecture is not suitable for this application.

We found that the reduction in the number of neurons in the input layer will not affect the network performance.

We were directed to change the number of the hidden layers to avoid the abrupt change in the number of neurons between two consecutive layers, we utilized many hidden layers, and we decrease the number of neurons gradually from the input layer to the output layer. Also we tried to change the transfer functions and used various transfer functions.

Also these changes in the architecture failed to make a possible training, and the (OUT Of MEMORY) error came out again. So we started to look for a new architecture which is the competitive layer. Tables 6 and 7 contain the results of the training with different combination of the network parameters and wavelet preprocessing:

Wavelet level	Size of the image matrix	Number of the input Neurons	Number of Hidden Neurons	Training Result.
Original Image	896 x 592	530432	265216	Failed
First	299 x 451	134,850	67425	Failed
Second	153 x 229	35,037	17519	Failed
Third	80 x 118	9440	4720	Failed
Fourth	43 x 62	2666	1333	Failed
Fifth	27 x 36	850	425	Failed
Sixth	18 x 22	396	198	Failed

Table (6): Summary of the feedforward results for one hidden neuron

Wavelet level	Number of input neurons	Number of hidden layers	Training Result.
Original Image	530432	2	Failed
		3	
First	134037	2	Failed
		3	
Second	35,037	2	Failed
		3	
Third	9440	2	Failed
		3	
Fourth	2666	2	Failed
		3	
Fifth	850	2	Failed
		3	
Sixth	396	2	Failed
		3	

Table (7): Summary of the feedforward results for multiple hidden neuron

## 5.6 Summary

This chapter discussed on of the most popular and effective network which is the Feedforward architecture, its construction, its training and the use of wavelet transformation within the Feedforward in order to reduce the size of the image which will reduce the number of neurons in the input layer, although the result of training was failed even by trying the fifth and sixth level of decomposition.

## **Chapter 6**

### **Competitive Layer ANN Design**

#### **6.1 Overview**

Since the feedforward architecture did not give the result expected from ANN, so another approach had to be used that why we started looking for a new approach.

As mentioned before (section 1.4); when the network does not solve the problem that it designed for, then the designer has to review the input and outputs, the number of layers, the number of elements per layer, and the connections between layers, and since we did all that in designing the feedforward Network and the result still the failing of the network, we were forced to go to try different architecture which is the Competitive Layer.

The Competitive Layer is unsupervised trained ANN that is widely used in classification application, so that we used it.

#### **6.2 Construction of Competitive Layer ANN**

The architecture of this network contains only one layer which is the competitive layer, the number of neurons in this layer equals the number of the different classes within the training data.

This simple network is supported by the MATLAB through Neural Network toolbox, there is simple MATLAB statements to construct this kind of networks, the following code summaries that;



```
net = newc(PR, S, KLR, CLR);
```

PR : R x 2 matrix of min and max values for R input elements. or simply the ranges of the training set  
S : Number of hidden neurons. Or number of classes  
KLR : S2 element vector of typical class percentages or the probability of occurrence.

The training process of this kind of networks can be held like that of the Feedforward networks, using the same code, except that the training process is a function of the number of iterations without making a goal control, the only way to control the performance is by repeating the training many times with different epochs number, as will be shown later.

### 6.3 Network design for the original size images

The original size image is very large as we show in the previous chapter, so we started from training set that contains only the 60 images (20 of each class; the following code summaries the process of making that training set.

```

image_RGB=imread('filename');
%where 'filename' is the name of the image file.

image_gray=image_RGB(:,:,1);
%converts the RGB to Grayscale.

image_vector=image_gray(:);
%converts the matrix to vector.

training_set(:,1)=image_vector;
% converts matrix to column vector.

```

The previous code used within a for loop function\*, will result the 'training\_set' matrix which contains coefficients of 60 images (60 columns) and 530432 element by column (the image coefficients).

The simplicity in this network is not only in the process but it extends to the design and training too. So we have to set only the training set and the ranges of the input neurons. And then the network will be ready to be constructed.

```

ranges=(minmax(data));

```

minmax will take the maximum value and the minimum value of the data matrix rows, and put these values in the (ranges) matrix

The construction of the Competitive Layer network then simply done by the following MATLAB statement.

---

\* The complete code supplied in appendix ().

```
net=newc(Ranges,3);
```

Since the number of classes in our design is three, so the number of neurons in the competitive layer will always be (3) regardless the number of input elements.

At this stage the network is ready to be trained and the training is simply can be held as in the FeedForward networks; except that we don't set a goal.

```
net.trainparam.epochs=1000;
```

```
net = train(net,data);
```

Since the competitive layer is unsupervised, the training command doesn't contain the targets or the desired output.

The training process failed to start in this case, and this is due to the large number of inputs, hence we started to reduce the number of inputs using the wavelet decomposition.

#### **6.4 Network design for reduced images size**

The 530432 elements of the original size image were reduced to 135148 elements using the first order wavelet decomposition, and to 35037 elements using the second order wavelet decomposition, both networks were failed to be trained too.



## 6.4.1 Network design using third wavelet decomposition

Since the 35037 elements was too large to be held by the training process we decompose the images one extra level using the following code to build the training set matrix. This time the elements are reduced to 9440 elements.

```
image_RGB=imread('filename');  
%where 'filename' is the name of the image file.  
  
image_gray=image_RGB(:,:,1);  
%converts the RGB to Grayscale.  
  
image_vector=image_gray(:);  
%converts the matrix to vector.  
  
[A1 A2 A3 A4]=dwt2(image_gray,'db4');  
%computes the first level wavelet decomposition.  
  
[B1 B2 B3 B4]=dwt2(A1,'db4');  
%computes the second level wavelet decomposition  
  
[C1 C2 C3 C4]=dwt2(B1,'db4');  
%computes the third level wavelet decomposition  
  
training_set(:,1)=image_vector;  
% converts matrix to column vector.
```

The previous code was applied to (for loop) function to get the whole matrix of training data 'training\_set'.

---

\* The whole code is supplied in appendix ().

```
net.trainparam.epochs=10000;
```

```
net = train(net,data);
```

Since the competitive layer is unsupervised, the training command doesn't contain the targets or the desired output.

The training process could be held this time, and the training was completed successfully. But the simulation results didn't reach the accuracy level that was expected from the ANN.

Table (8) shows the results of training the network for 10000 iterations and this table shows how many images from the training set could be classified correctly.

	Number of correct classified images out of 59	% Error
Class A	5	91.53 %
Class B	6	89.83 %
Class C	3	94.91 %
Average Error		92.09 %

Table (8): Results of Competitive ANN (9440 elements)  
for 10000 epochs

The percentage error is too large and this is due to the large number of inputs compared to the number of epochs so we raised the number of epochs in order to reduce the percentage error. We used many values for the epochs and we simulated the network each time we raised the epochs.

- **15000 epochs**

Table (9) shows that a 5000 extra epochs decreased the average error a very small value about 3%, then extra epochs were added until we got acceptable result.

	Number of correct classified images out of 59	% Error
Class A	6	89.83 %
Class B	6	89.83 %
Class C	4	93.22 %
Average Error		90.96 %

Table (9): Results of Competitive ANN (9440 elements)  
for 15000 epochs

- **30000 epochs**

Table (10) shows the result of 30000 epochs which improved the average error and decreased it about 8%.

	Number of correct classified images out of 59	% Error
Class A	11	81.36 %
Class B	13	77.97 %
Class C	9	84.75 %
Average Error		82.36 %

Table (10): Results of Competitive ANN (9440 elements)  
for 30000 epochs



- **50000 epochs**

Table (11) shows the result of 50000 epochs which improved the average error and decreased it about 30%.

	Number of correct classified images out of 59	% Error
Class A	25	57.63 %
Class B	30	49.15 %
Class C	28	52.54 %
Average Error		53.11 %

Table (11): Results of Competitive ANN (9440 elements)  
for 50000 epochs

- **100000 epochs**

Since we have a good improvement by making more iteration we added the epochs to 100000 epochs and table (12) shows its improvement.

	Number of correct classified images out of 59	% Error
Class A	40	32.20 %
Class B	41	30.51 %
Class C	44	25.42 %
Average Error		29.38 %

Table (12): Results of Competitive ANN (9440 elements)  
for 100000 epochs

The statistics graph shown in figure (27) and this shows the relation between the epochs of training and the correctly classified images.



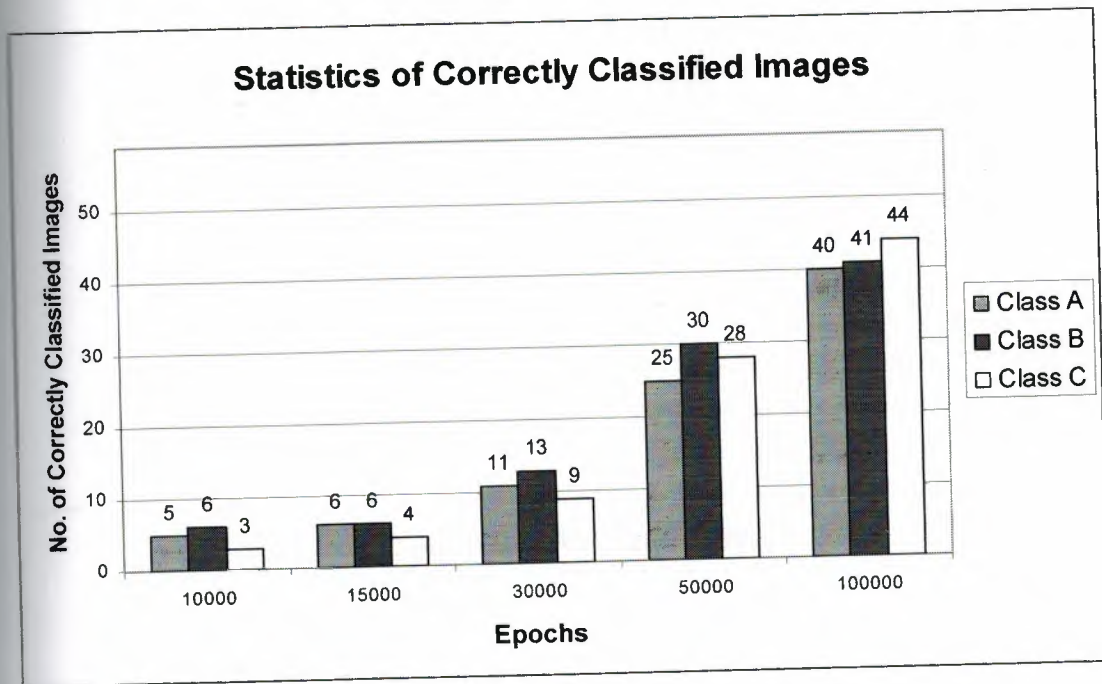


Figure (27): Statistics graph shows the amount of correctly classified images for Competitive ANN (9440 elements).

The accuracy was rose to 70.62% which is still unacceptable from ANN since when we use ANN we expect 85-95% accuracy. Also if we kept adding extra epochs the training process will not be efficient since the 100000 epochs training consumes 18 hours processing on 3.0GHz PENTIUM 4 processor with 1024MB RAM. So go back to reduce the number of the inputs by applying another wavelet level of decomposition.

Table (13) shows the average error related to the number of training epochs, and the graph shown in figure (28) shows the reverse proportional relation between the average error and the number of iterations.

No. of Iterations (Epochs)	Average Error
10000	92.09
15000	90.96
30000	82.36
50000	53.11
100000	29.38

Table (13): Epochs vs. Average Error for  
Competitive ANN (9440 elements)

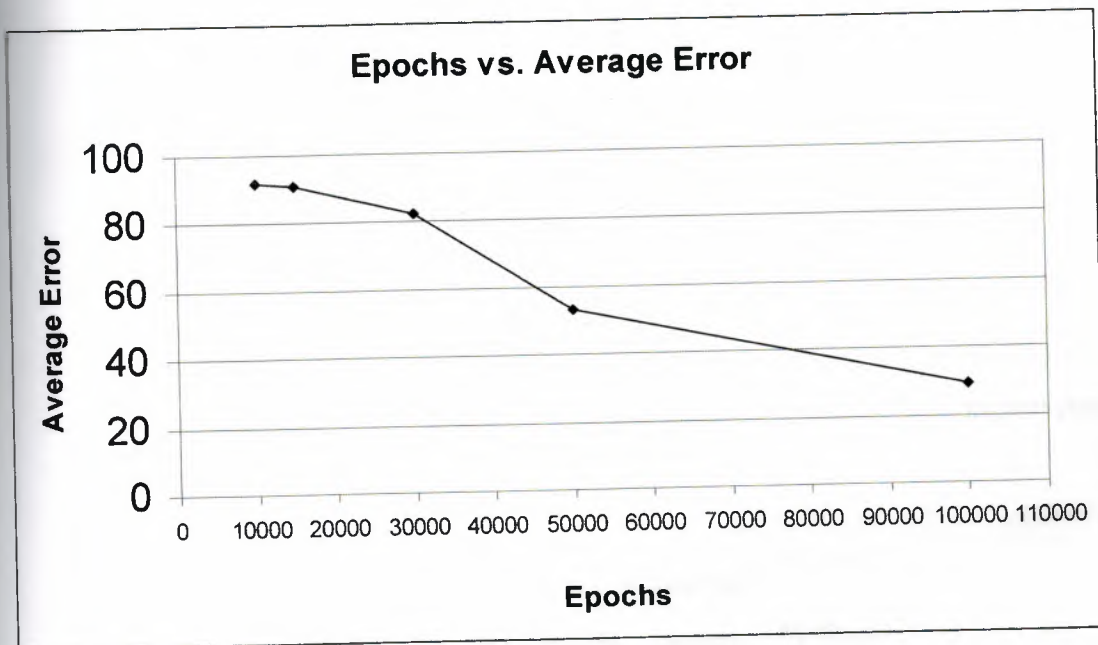


Figure (28): Epochs Vs. Average Error for Competitive ANN (9440 elements)

#### 6.4.2 Network design using fourth level wavelet decomposition

The fourth level results 2666 elements for the input, and this number can be utilized easier by the training process. The code that extract the training set and the code that initiates the network structure and the training process are summarized below and the complete code is still available in appendix ().

Also for this number of inputs we made many trials on iterations we started from lower values since the number of inputs is reduced in this case. The

start was with 5000 epochs which gave approximately the results of 10000 epochs in the previous design. As shown in table (14).

	Number of correct classified images out of 59	% Error
<b>Class A</b>	5	91.53 %
<b>Class B</b>	6	89.83 %
<b>Class C</b>	5	91.53 %
<b>Average Error</b>		90.96 %

Table (14): Results of Competitive ANN (2666 elements)  
for 5000 epochs

- **10000 epochs**

A good improvement in the accuracy by making more iterations we added the epochs to 10000 epochs and table (15) shows this improvement which is 10%.

	Number of correct classified images out of 59	% Error
<b>Class A</b>	13	77.97 %
<b>Class B</b>	11	81.36 %
<b>Class C</b>	10	83.05 %
<b>Average Error</b>		80.79 %

Table (15): Results of Competitive ANN (2666 elements)  
for 10000 epochs

- **15000 epochs**

The accuracy improved again with 15000 epochs and this is shown in table (16).



	Number of correct classified images out of 59	% Error
Class A	20	66.10 %
Class B	18	69.49 %
Class C	22	62.71 %
Average Error		66.10 %

Table (16): Results of Competitive ANN (2666 elements)  
for 15000 epochs

- **30000 epochs**

The accuracy improved to 50 % with 30000 epochs and this is shown in table (17).

	Number of correct classified images out of 59	% Error
Class A	32	45.76 %
Class B	30	49.15 %
Class C	29	50.85 %
Average Error		48.59 %

Table (17): Results of Competitive ANN (2666 elements)  
for 30000 epochs

- **50000 epochs**

The accuracy improved to 85.3 % with 50000 epochs and this is shown in table (18). These results have accepted accuracy since our acceptable range is (85-95%). But to improve these results we have two choices the first one is to increase the epochs or decrease the number of inputs which will decrease the enough epochs.



Hence we will start simulating another size of networks that with the fifth level wavelet decomposition. And this network may be able to be trained with lower number of epochs.

	Number of correct classified images out of 59	% Error
Class A	48	18.64 %
Class B	52	11.86 %
Class C	51	13.56 %
Average Error		14.69 %

Table (18): Results of Competitive ANN (2666 elements)  
for 50000 epochs

The statistics graph shown in figure (29) and this shows the relation between the epochs of training and the correctly classified images.

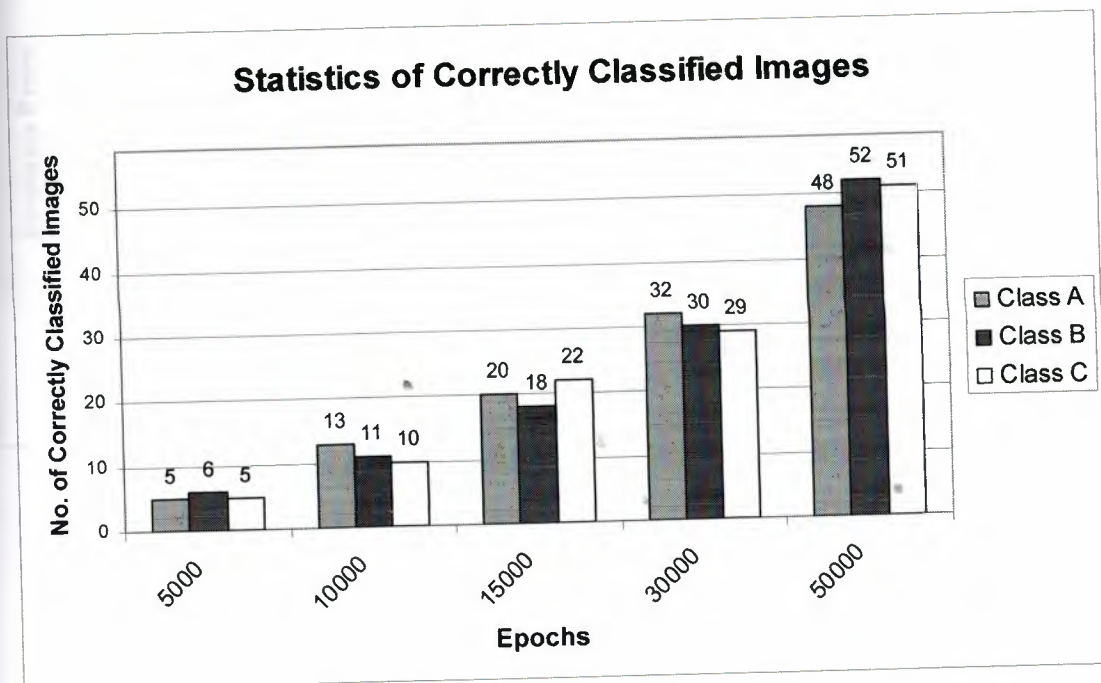


Figure (29): Statistics graph shows the amount of correctly classified images for Competitive ANN (2666 elements).

Table (19) shows the average error related to the number of training epochs, and the graph shown in figure (30) shows the reverse proportional relation between the average error and the number of iterations.

No. of Iterations (Epochs)	Average Error
5000	90.96
10000	80.79
15000	66.10
30000	48.59
50000	16.38

Table (19): Epochs vs. Average Error  
for Competitive ANN (2666 elements)

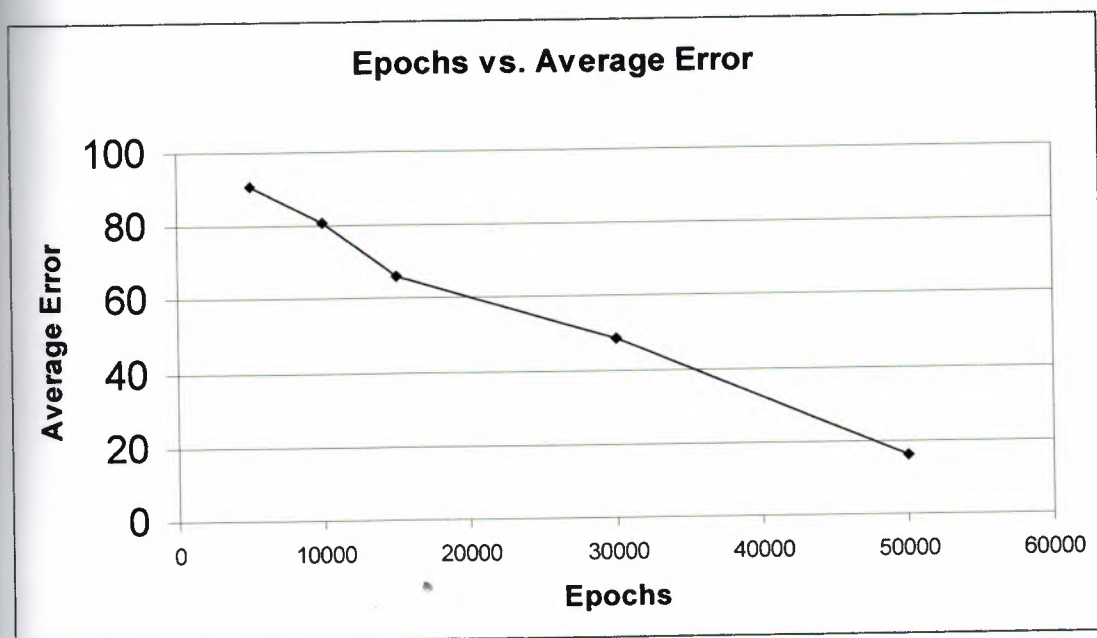


Figure (30): Epochs Vs. Average Error for Competitive ANN (2666 elements)

#### 6.4.3 Network design using fifth level wavelet decomposition

The 850 elements which were results from the fifth wavelet decomposition will be fed to the new size network. This will be easier to train. The complete MATLAB code is attached in the appendix.

The training of the network now may be start from smaller number of epochs (iterations). So we started from 1000 epochs. This gave approximately the same accuracy of the 10000 epochs of the 9440 input network, and the accuracy of the 5000 epochs of the 2666 inputs network.

The results of the 1000 epochs training of the 850 inputs network is shown in table (20) which shows the improvement in the design.

	Number of correct classified images out of 59	% Error
Class A	7	88.13 %
Class B	6	89.83 %
Class C	7	88.13 %
Average Error		88.70 %

Table (20): Results of Competitive ANN (850 elements)  
for 1000 epochs

- **3000 epochs**

We used 3000 epochs in order to get more accuracy and this is clear in table (21) which shows 11 % improvement by adding only 2000 epochs in contrast with the previous designs which took more than 10000 epochs to get this improvement.

	Number of correct classified images out of 59	% Error
Class A	14	76.27 %
Class B	14	76.27 %
Class C	12	79.66 %
Average Error		77.40 %

Table (21): Results of Competitive ANN (850 elements)  
for 3000 epochs

- **5000 epochs**

The accuracy improved by 24% again with 5000 epochs and this is better than the previous designs shown in table (22).

	Number of correct classified images out of 59	% Error
Class A	26	55.93 %
Class B	29	50.85 %
Class C	27	54.24 %
Average Error		53.67 %

Table (22): Results of Competitive ANN (850 elements)  
for 5000 epochs

- **10000 epochs**

The average error reduced again, which means improvement in the accuracy as shown in table (23).



	Number of correct classified images out of 59	% Error
Class A	39	33.90 %
Class B	41	30.51 %
Class C	43	27.12 %
Average Error		30.51 %

Table (23): Results of Competitive ANN (850 elements)  
for 10000 epochs

- **20000 epochs**

The accuracy rose to 85.88% as shown in table (24) which is acceptable, but still can be improved by more epochs. So we used the 40000 epochs.

	Number of correct classified images out of 59	% Error
Class A	51	13.56 %
Class B	51	13.56 %
Class C	50	15.25 %
Average Error		14.12 %

Table (24): Results of Competitive ANN (850 elements)  
for 20000 epochs

- **40000 epochs**

A 40000 epoch rose the accuracy to 93.22% as shown in table (25) which is the best accuracy we could get from the Competitive Layer ANN. Since we had approximately the same value for 100000 epochs with the same network

	Number of correct classified images out of 59	% Error
Class A	55	6.78 %
Class B	54	8.47 %
Class C	56	5.08 %
Average Error		6.78 %

Table (25): Results of Competitive ANN (850 elements)  
for 40000 epochs

The statistics graph shown in figure (31) shows the amount the correctly classified images at different values of epochs.

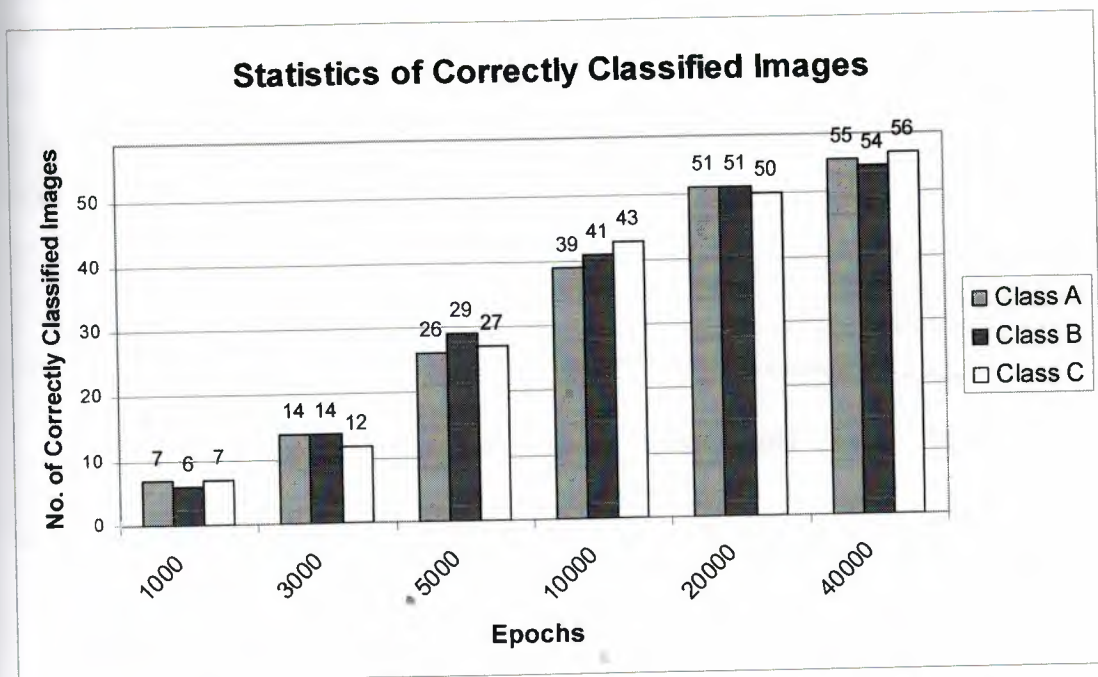


Figure (31): Statistics graph shows the amount of correctly classified images  
for Competitive ANN (850 elements).

Table (26) shows the average error related to the number of training epochs, and the graph shown in figure (32) shows the reverse proportional relation between the average error and the number of iterations.

No. of Iterations (Epochs)	Average Error
1000	88.70
3000	77.40
5000	53.67
10000	30.51
20000	14.12
40000	6.78

Table (26): Epochs vs. Average Error for  
Competitive ANN (850 element)

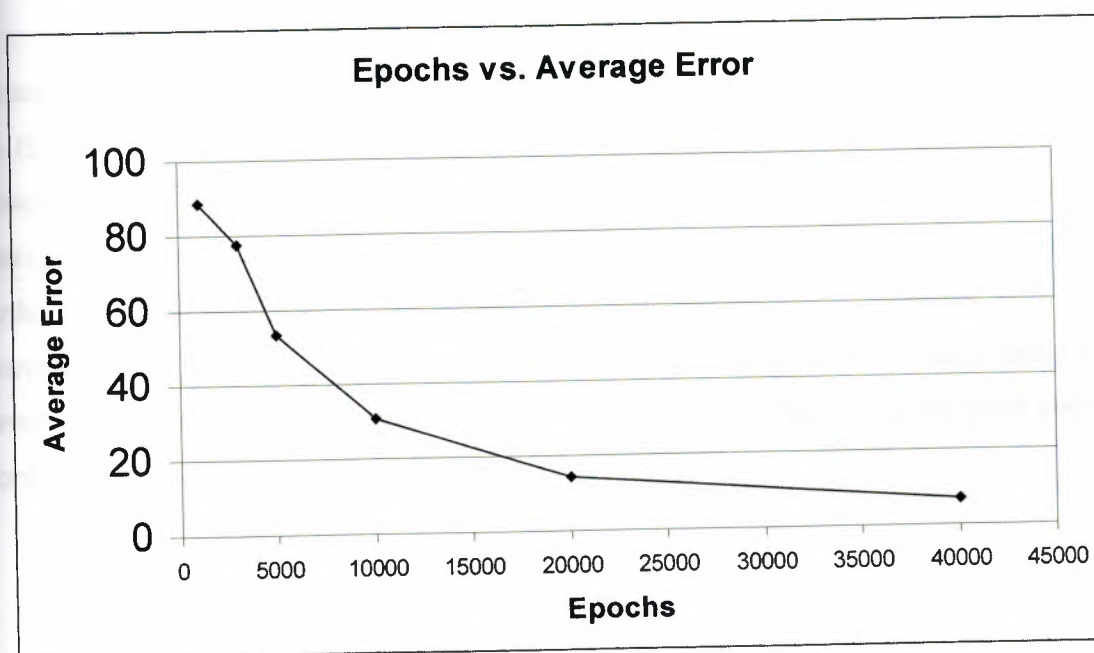


Figure (32): Epochs Vs. Average Error for Competitive ANN (850 elements).

#### ▪ *The End result of Competitive Layer Networks*

It is obvious that the last design, the design of the (850 inputs), is the best over all design, since it has the best performance over all the previous designs. So we consider this one as the best design because it is easier to train, build and train.

Because this design is not reserved only for this application, but we can train it for any set of images so the training process which is the most time consuming process can be held in shorter time than the other designs.

This design may be the best over the previous but there is another architecture that may present a better design. And that is the Learning Vector Quantization which will be discussed in the next chapter.

## 6.5 Summary

This chapter discussed the competitive layer ANN design which is an unsupervised training ANN, the architecture of this network contain only one layer which is the competitive layer. This chapter shows the network design for the original size image using the MATLAB code which was failed , the this chapter discussed the network design for reduced image size which failed to classify the images in the first and second order wavelet decomposition, but it was successful for the third , forth and fifth order wavelet decomposition , but as the experiment shows that network design using fifth level wavelet decomposition was better than the network design using the third and the forth order wavelet decomposition.



## Chapter 7

### Classification using Learning Vector Quantization

#### 7.1 Overview

We were directed to the Learning Vector Quantization ANN, because it has the architecture of the competitive layer plus the ability to have supervised training, also this kind of networks gives the opportunity to control more characteristics of the network. Hence in the competitive architecture we train the network for specific number of epochs without having an idea about mathematical goal, but using this kind of training we have a mathematical goal and we can trace the process epoch by epoch from the (goal vs. epochs) graph as will be shown later.

#### 7.2 Construction of LVQ ANN

The following MATLAB statement defines the LVQ parameters;

```
net = newlvq(PR,S1,PC,LR,LF);
```

PR	:	R x 2 matrix of min and max values for R input elements. or simply the ranges of the training set
S1	:	Number of hidden neurons. or number of hidden classes
PC	:	S2 element vector of typical class percentages or the probability of occurrence.
LR	:	Learning rate, default = 0.01.
LF	:	Learning function, default = 'learnlv2'. returns a new LVQ network. The learning function LF can be learnlv1 or learnlv2.

Due to the existence of the competitive layer in this architecture it has some limitations of the competitive network, such that we are limited by the number of the input neurons as will be shown later.

### 7.3 Network design for the original size images

The following MATLAB code will result the matrix of the training set images using their original size;

```
image_RGB=imread('filename');  
%where 'filename' is the name of the image file.  
  
image_gray=image_RGB(:,:,1);  
%converts the RGB to Grayscale.  
  
image_vector=image_gray(:);  
%converts the matrix to vector.  
  
training_set(:,1)=image_vector;  
% converts matrix to column vector.
```

Evaluating the previous code for 20 images of each class (using for loop function)\* will yield a matrix that has a number of rows equal to the number of elements of the grayscale image matrix, and number of columns equal to that of the total number of images. The training set was reduce because of the limitations of MATLAB.

Now, to initiate the network we have to set the target vector and we used the following code to do that keeping in mind that we had 20 images for each class in the training set; and the target matrix in this case is different from that of

---

\* The complete code supplied in appendix ().

the feedforward network, the target matrix of the LVQ network has a special shape that will be provided by the following code;

```
targets_classes_indices(1,1:1:20) = 1;  
%sets the first 59 elements in the target row vector to one which means the first class.  
  
targets_classes_indices (1,21:1:40) = 2;  
%sets the second 59 elements in the target row vector to two which means the second  
class.  
  
targets_classes_indices (1,41:1:60) = 3;  
%sets the third 59 elements in the target row vector to three which means the third  
class.  
  
targets_classes_vector = ind2vec(targets_classes_indices);  
%converts the targets vector from indices form into vector form.
```

Now the network can be initiated, using the following code;

```
net = newlvq(minmax(training_set),6,[1/3 1/3 1/3],0.001,'learnlv1');
```

Minmax(training_set)	:	Equal to the ranges matrix of the input training set.
6	:	Is the number of subclasses
[1/3 1/3 1/3]	:	The percentage of each class training set.
0.001	:	The training rate
'learnlv1'	:	The learning function

The training parameter of this network can be set by the following code;

```
net.trainparam.goal = 0.1;  
% sets the goal of minimum difference.  
  
net.trainparam.epochs = 1000;  
% sets the number of maximum iterations to be held unless the goal is met.
```

The training command is;

```
net = train(net, training_set, target_classes_vector);
```

Although the training set matrix area was reduced but the training process was not able to be completed, and we faced an ( OUT OF MEMORY ) type error again. And from this point we started to reduce the number of input elements using the wavelet transformation.

#### **7.4 LVQ Network design for reduced images size (one wavelet level)**

In order to make the input vectors more suitable for the ANN use, we started to use the wavelet transformation, which will reduce the image size keeping the most important details unchanged (as mentioned before).

The following MATLAB code will result the matrix of the training set images using the first level wavelet output;



```

image_RGB = imread('filename');
%where 'filename' is the name of the image file.

image_gray = image_RGB(:,:,1);
%converts the RGB to Grayscale.

[A1 A2 A3 A4] = dwt2(image_gray,'db4');
%computes the first level wavelet decomposition.

image_vector = A1(:);
%converts the matrix to vector.

training_set(:,1) = image_vector;
% to fill the training_set matrix

```

Evaluating the previous code for all images (using for loop function)\* will yield a matrix that has a number of rows equal to the number of elements of the approximate image matrix which is (134849), and number of columns equal to that of the total number of images of the training set which is 177.

Now, to initiate the network we have to set the target vector (as in the previous design) and we used the same code of the targets of the previous section, to do that keeping in mind that we have 59 images for each class in this case\*; and this code is applicable for any design uses LVQ (Learning Vector Quantization) algorithm in this research.

Now the network can be initiated, as the previous trial using the same code of the previous section;

---

\* The complete code supplied in appendix ().

```
net = newlvq(minmax(training_set), 6, [1/3 1/3 1/3], 0.001, 'learnlv1');
```

The training parameter of this network can be set by the following code;

```
net.trainparam.goal = 0.1;  
% sets the goal of minimum difference.  
  
net.trainparam.epochs = 1000;  
% sets the number of maximum iterations to be held unless the goal is met.
```

The training command is;

```
net = train(net, training_set, target_classes_vector);
```

Also the three previous segments of code are fixed in the LVQ algorithm\* unless any of the parameter is changed and we will mention any parameter change when it happened.

The limitations this time stopped the process at the training stage, the training we made were faced with an (OUT OF MEMORY) error too. This led us to another level of wavelet transformation.

---

\* The complete code supplied in appendix ()

## **7.5 LVQ Network design for reduced images size (two levels wavelet)**

The number of input elements was reduced again to (35037) element\*, with neglecting the less important features in the previous approximate image, by applying another level of wavelet transformation.

Although of this reduction in the input neurons number the network was unable to be trained this time too.

So we applied the wavelet transformer again to have three levels wavelet as result. And it seems that it will work this time (as will be shown).

## **7.6 LVQ Network design for reduced images size (three levels wavelet)**

The number of input elements was reduced again, in order to be adequate for the network to have acceptable training process and accuracy.

The following MATLAB code will result the matrix of the training set images using the third level wavelet transformation output;

---

\* The complete codes for this design are available in appendix ().

```

image_RGB = imread('filename');
%where 'filename' is the name of the image file.

image_gray = image_RGB(:,:,1);
%converts the RGB to Grayscale.

[A1 A2 A3 A4] = dwt2(image_gray,'db4');
%computes the first level wavelet decomposition.

[B1 B2 B3 B4] = dwt2(A1,'db4');
%computes the second level wavelet decomposition

[C1 C2 C3 C4] = dwt2(B1,'db4');
%computes the third level wavelet decomposition

image_vector = C1(:);
%converts the matrix to vector.

training_set(:,1) = image_vector;
% converts matrix to column vector.

```

Evaluating the previous code for all images (using for loop function)\* will yield the a matrix has a number of rows equal to the number of elements of the approximate image matrix which is (9440), and number of columns equal to that of the total number of images of the training set which is 177.

The target vector was as mentioned in the previous section still valid in this section. The initiation of the network and the training still valid from the previous section too, and we can do it simply by running the following code;

---

\* The complete code supplied in appendix ()



```

%Network initiation.
net = newlvq(minmax(training_set), 6, [1/3 1/3 1/3],
0.001, 'learnlv1');

%Training Parameters.
net.trainparam.goal = 0.1;
net.trainparam.epochs = 1000;

%Training command
net = train(net, training_set, target_classes_vector);

```

Although the previous command lines sets the training goal to 0.1 and the iterations epochs to 1000, the training process was failed to meet the goal and the performance graph was as shown in figure (33), but this time the training started at least.

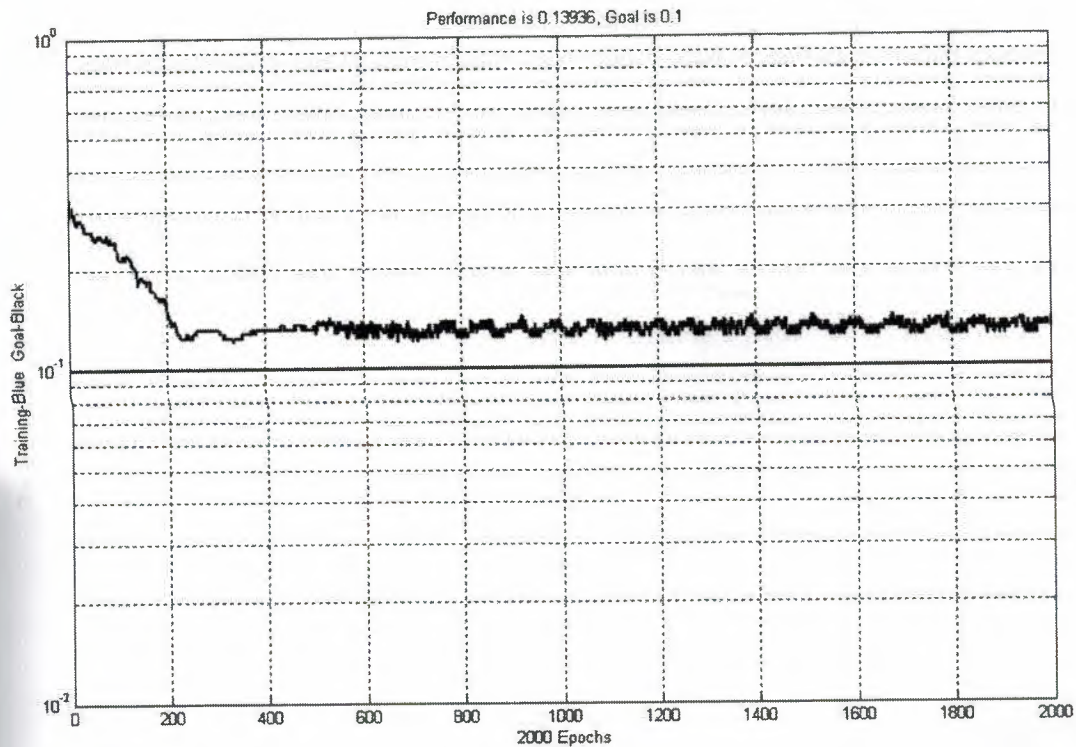


Figure (33): Performance graph of LVQ network with three levels wavelet and 0.1 goal.

The previous graph illustrates the relation between the performance of the network to the iterations number, as it is shown above the goal is not met, this doesn't mean that this network can not be trained. It can be trained but with a little bit higher goal. So that we trained it again with 0.135 goal.

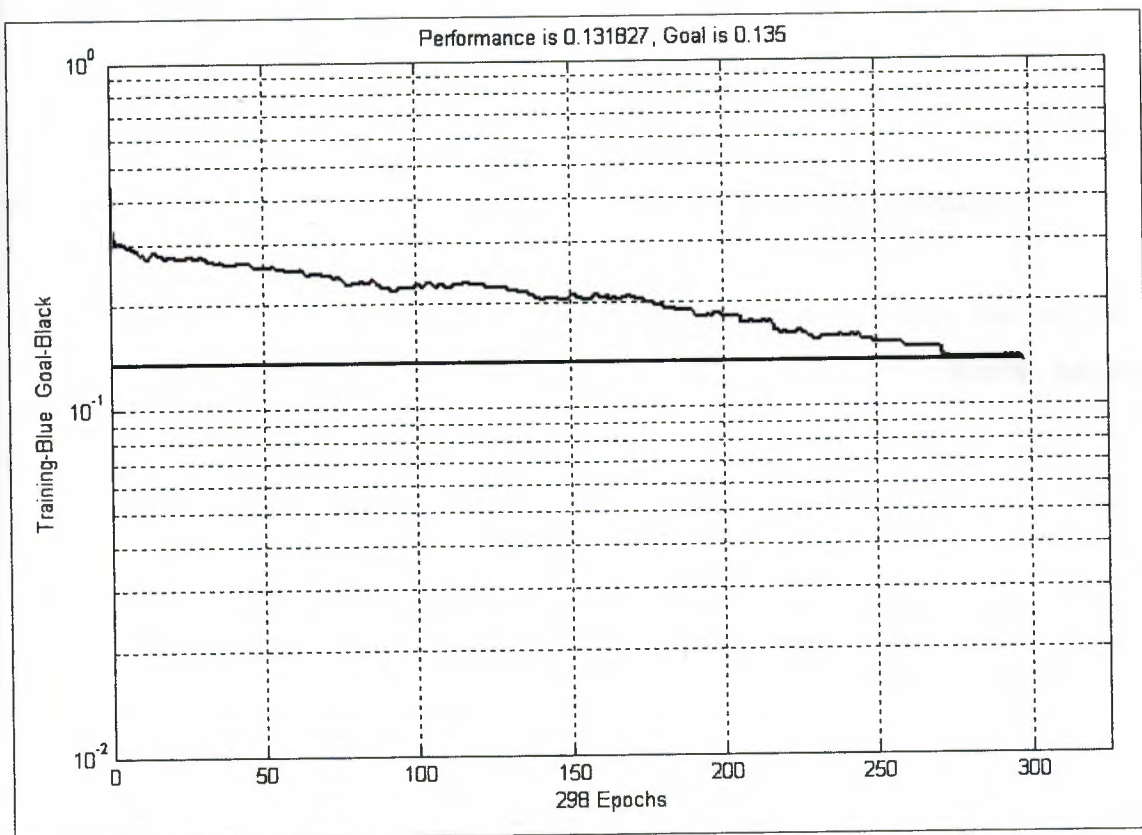


Figure (34): Performance graph of LVQ network with three levels wavelet and 0.135 performance.

The goal was met this time and after simulation the results of accuracy was as shown in table (27) which shows the accuracy of the trained network at performance of 0.135.

	Number of correctly classified images out of 59	% Accuracy
<b>Class A</b>	43	72.88 %
<b>Class B</b>	45	76.27 %
<b>Class C</b>	45	76.27 %
<b>Average Accuracy</b>		75.14 %

Table (27): The accuracy table of LVQ network with three wavelet levels and performance of 0.135.

The results in table (27), give a step forward for the whole research, since the process is easier and controllable through the goal setting and through the hidden neurons of the network (sub-classes)

So we completed the research on this architecture of networks to show its efficiency and the ability to be considered as final result of the research.

## **7.7 Network design for reduced images size (four levels wavelet)**

Although the previous design was trained successfully but we were interested in higher accuracy level so we applied the fourth wavelet decomposition level on the images.

The code that sets the training set, the desired output (targets), the network, the training parameters and the training can be done using the code of the previous design except that we need to modify the training set code to make the fourth level\*.

The training process was held on two goal values 0.1 and 0.125; the first one didn't meet the goal where it was met by the second one at performance of (0.124294), as shown in figure (35).

---

\* For complete code see Appendix ().



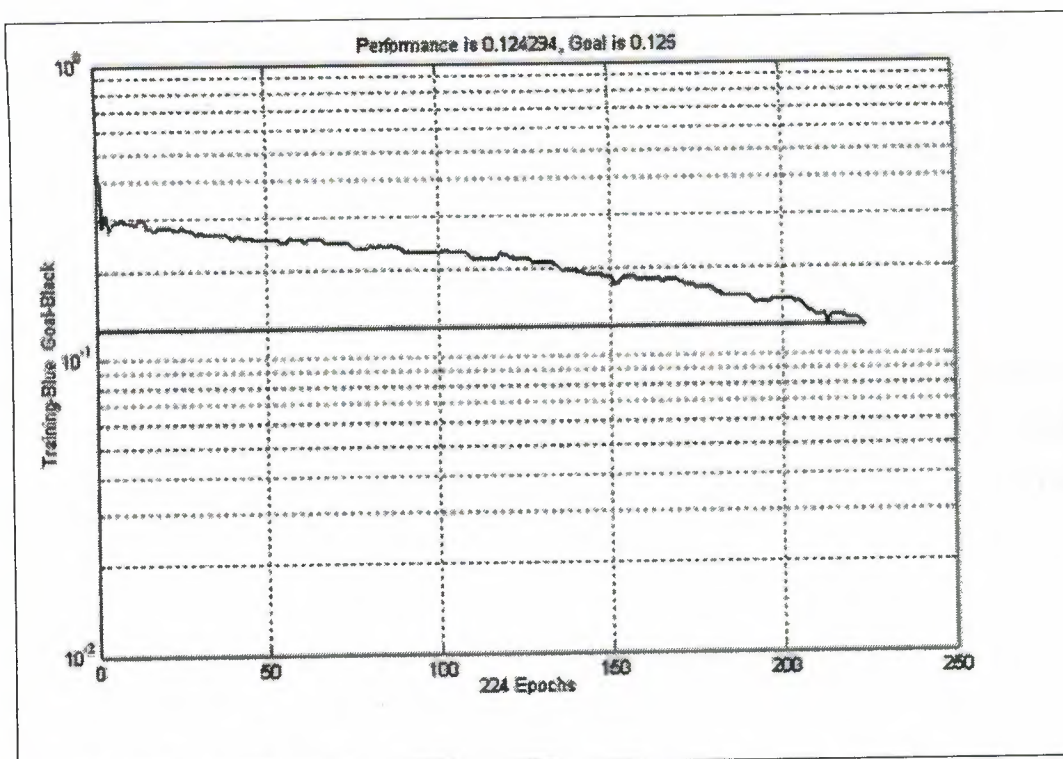


Figure (35): Performance graph of LVQ network with four levels wavelet and 0.125 performance.

The simulation result of this network was as shown in table (28). And these results show that the accuracy improved to 84.75% by this reduction in the input elements. And this may lead the research to another level of reduction that may increase the accuracy more than that.

	Number of correctly classified images out of 59	% Accuracy
<b>Class A</b>	49	83.05 %
<b>Class B</b>	51	86.44 %
<b>Class C</b>	50	84.75 %
<b>Average Accuracy</b>		84.75 %

Table (28): The accuracy table of LVQ network with four wavelet levels with performance of 0.124.

## 7.8 Network design for reduced images size (five levels wavelet)

The previous design results an accuracy level of about 85% which is still low level compared to accepted levels of classification networks. Hence we applied the fifth level of wavelet decomposition on the images.

The code that sets the training set, the desired output (targets), the network, the training parameters and the training can be done using the code of the previous design except that we need to modify the training set code to make the fifth level\*.

This network trained to meet the performance goal of 0.1, and this goal was met by the training within only 63 epochs. And the result of the simulation of this network is shown in table (29). While the performance graph is shown in figure (36).

	<b>Number of correctly classified images out of 59</b>	<b>% Accuracy</b>
<b>Class A</b>	52	88.14 %
<b>Class B</b>	51	86.44 %
<b>Class C</b>	52	88.14 %
<b>Average Accuracy</b>		87.57 %

Table (29): The accuracy table of LVQ network with five wavelet levels  
with performance of 0.1.

---

\* For complete code see Appendix ().

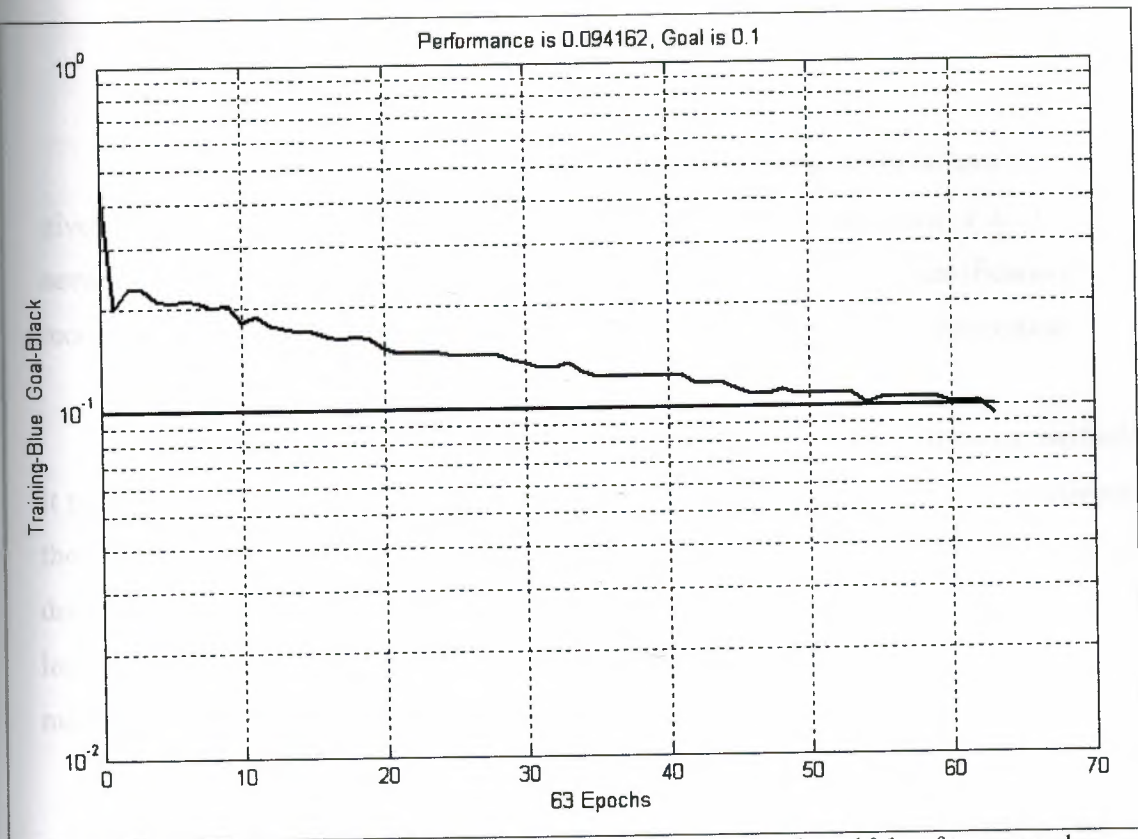


Figure (36): Performance graph of LVQ network with five levels wavelet and 0.1 performance goal.

## 7.9 summary

This chapter discussed the classification using learning vector quantization because its architecture has the architecture of competitive layer plus the ability to have supervised training; this chapter show the construction of the LVQ ANN using MATLAB code.

The LVQ ANN was applied on the original image but the training failed so the wavelet decomposition was considered, and it was started from the first wavelet decomposition which got failed and the same result for the second wavelet decomposition order, so it considered the third, forth, and fifth order of wavelet which was successfully trained and the fifth wavelet decomposition order proved that it's the best order in the classification.



## Conclusion

This project discussed the image classification using artificial neural network, it gives an explanation about the artificial neural network and the uses of artificial neural network in many fields for example in medicine, image classification, pattern recognition, banking and finance, language processing and character recognition.

This research discussed the methods of classification and show many methods but it proves that the most better method is classification using artificial neural network and the it also prove that classification using artificial neural network with wavelet levels decomposition is more better, since wavelet decomposition reduce the image size without losing the image data, in other words reducing the image size means reducing the number of inputs to the neural network so it will gives more clear results

For image classification this project used three kinds of neural networks FeedForward neural network, competitive layer neural network and learning vector quantization neural network.

Image classification using FeedForward neural network which is a supervised learning neural network, was failed for the original image classification which was with size  $(896 \times 592)$  and gives an error OUT OF MEMORY when the code was run in MATLAB , so the wavelet decomposition was considered but the training result of the network for the first, second, third, fourth, fifth and sixth level of wavelet decomposition was OUT OF MEMORY which mean that the image classification using this neural network was failed, so the competitive neural network was considered.

Image classification using competitive neural network, which is unsupervised neural network, was failed for the original image classification and this due to size of image which gives a large input to the neural network, and also the classification was failed for the first and the second level of wavelet decomposition, but the image classification started for the third level decomposition so we start to use this level with



many epochs in order to get more clear result, and it was the same for the forth and fifth level of wavelet decomposition , but the problem was that with each level every time it need to increase or decrease the epochs size and this is taking a time, since training the neural network with 100000 epochs took about 18 hours and half on a computer with 3.0 GHz microprocessor and with 1024MB ram, so we considered the learning vector quantization neural network.

This project directed to the Learning Vector Quantization ANN, because it has the architecture of the competitive layer plus the ability to have supervised training, also this kind of networks gives the opportunity to control more characteristics of the network. Hence in the competitive architecture we train the network for specific number of epochs without having an idea about mathematical goal, but using this kind of training we have a mathematical goal and we can trace the process epoch by epoch from the (goal vs. epochs), training the network for the original image was failed and also for the first and second level of wavelet decomposition. Using the third level of wavelet decomposition with 0.1 goals and with iteration epochs 1000 was failed but the classification was started then with same iteration epochs but with increasing the goal to 0.135 the performance graph show the goal was met. Using the forth wavelet decomposition with 0.1 goal the same problem the goal was not 100% met but a little increasing in the goal up to 0.125 will met the waned goal and this is shown in the performance graph, so we directed to the fifth level wavelet decomposition where the goal was met just 0.1 and this was shown in the performance graph.

According to this research we can conclude that the image classification using learning vector quantization is the better than the competitive layer and the FeedForward neural network, also wavelet analysis help in reducing the image size which reduce the inputs to the network. Also the as we more reduce the image we can reach the wanted goal with less values as was proved in this research

## List of Figures

Figure No.	Figure Title
1	The main 4 parts of the Nerve Cell (neuron)
2	The Artificial Neuron
3	Simple neuron with bias
4	General architecture of multi layer neural network
5	Artificial neural network layers
6	Supervised training process.
7	The architecture of feedforward network.
8	Architecture of competitive network.
9	Architecture of LVQ network.
10	An Example Learning Vector Quantization Network.
11	Demonstration of (a) a Wave and (b) a Wavelet.
12	Three-level wavelet decomposition tree.
13	Three-level wavelet reconstruction tree.
14	Wavelet families (a) Haar (b) Daubechies4 (c) Coiflet1 (d) Symlet2 (e) Meyer (f) Morlet (g) Mexican Hat.
15	the first class example
16	the second class example
17	the third class example
18	The original image displayed with 50% scale.
19	The reconstructed image displayed with 50% scale.
20	The reconstructed grayscale image displayed with 50% scale.
21	The original image 'LA_01.jpg' displayed with 50% scale.
22	The first level wavelet decomposition result displayed with 100% scale.
23	The second level wavelet decomposition result displayed with 100% scale.
24	Tree arrangement of five levels wavelet decomposition.
25	The decomposition of five levels wavelet.
26	Simple representation of feedforward ANN.
27	Statistics graph shows the amount of correctly classified images for Competitive ANN (9440 elements).
28	Epochs Vs. Average Error for Competitive ANN (9440 elements)
29	Statistics graph shows the amount of correctly classified images for Competitive ANN (2666 elements).
30	Epochs Vs. Average Error for Competitive ANN (2666 elements)
31	Statistics graph shows the amount of correctly classified images for Competitive ANN (850 elements).
32	Epochs Vs. Average Error for Competitive ANN (850 elements).
33	Performance graph of LVQ network with three levels wavelet and 0.1 goal.
34	Performance graph of LVQ network with three levels wavelet and 0.135 performance.
35	Performance graph of LVQ network with four levels wavelet and 0.125 performance.
36	Performance graph of LVQ network with five levels wavelet and 0.1 performance goal.



## List of Tables

Figure No.	Table Title
1	Neurons Transfer functions
2	ANN vs. traditional computing
3	Network Selector Table
4	The wavelet reduction procedure results
5	The reduction of matrix sizes due to multi level wavelet transformation
6	Summary of the feedforward results for one hidden neuron
7	Summary of the feedforward results for multiple hidden neuron
8	Results of Competitive ANN for 10000 epochs (9440 elements)
9	Results of Competitive ANN for 15000 epochs (9440 elements)
10	Results of Competitive ANN (9440 elements) for 30000 epochs
11	Results of Competitive ANN (9440 elements) for 50000 epochs
12	Results of Competitive ANN (9440) for 100000 epochs
13	Epochs vs. Average Error for Competitive ANN (9440 elements)
14	Results of Competitive ANN (2666 elements) for 5000 epochs
15	Results of Competitive ANN (2666 elements) for 10000 epochs
16	Results of Competitive ANN (2666 elements) for 15000 epochs
17	Results of Competitive ANN (2666 elements) for 30000 epochs
18	Results of Competitive ANN (2666 elements) for 50000 epochs
19	Epochs vs. Average Error for Competitive ANN (2666 elements)
20	Results of Competitive ANN (850 elements) for 1000 epochs
21	Results of Competitive ANN (850 elements) for 3000 epochs
22	Results of Competitive ANN (850 elements) for 5000 epochs
23	Results of Competitive ANN (850 elements) for 10000 epochs
24	Results of Competitive ANN (850 elements) for 20000 epochs
25	Results of Competitive ANN (850 elements) for 40000 epochs
26	Epochs vs. Average Error for Competitive ANN (850 elements)
27	The accuracy table of LVQ network with three wavelet levels and performance of 0.135.
28	The accuracy table of LVQ network with four wavelet levels with performance of 0.124.
29	The accuracy table of LVQ network with five wavelet levels with performance of 0.1.

# Appendix A

## MATLAB code

Training set Matrix (for original image)

```
% for class A images (59 images) where the images saved in the work space in the
form of 'LA_'i'.jpg' where I is the image number.
for i=1:1:59
image_RGB = imread(['LA_',num2str(i),'.jpg']); %converts the image to 3D-matrix.
image_gray = image_RGB(:,:,1); %converts the 3D-matrix to 2D-matrix.
image_vector = image_gray(:); % converts the matrix to column vector.
training_set(:,i) = image_vector; % fills the training set matrix.
end

% for class B images (59 images) where the images saved in the work space in the
form of 'LB_'i'.jpg' where I is the image number.
for i=1:1:59
image_RGB = imread(['LB_',num2str(i),'.jpg']); %converts the image to 3D-matrix.
image_gray = image_RGB(:,:,1); %converts the 3D-matrix to 2D-matrix.
image_vector = image_gray(:); % converts the matrix to column vector.
training_set(:,i+59) = image_vector; % fills the training set matrix.
end

% for class C images (59 images) where the images saved in the work space in the
form of 'LC_'i'.jpg' where I is the image number.
for i=1:1:59
image_RGB = imread(['LC_',num2str(i),'.jpg']); %converts the image to 3D-matrix.
image_gray = image_RGB(:,:,1); %converts the 3D-matrix to 2D-matrix.
image_vector = image_gray(:); % converts the matrix to column vector.
training_set(:,i+59) = image_vector; % fills the training set matrix.
End
```



### Training set Matrix (for one level WAVELET decomposition)

```
% for class A images (59 images) where the images saved in the work space in the form of
'LA_'i'.jpg' where I is the image number.
for i=1:1:59
image_RGB = imread(['LA_',num2str(i),'.jpg']); %converts the image to 3D-matrix.
image_gray = image_RGB(:,:,1); %converts the 3D-matrix to 2D-matrix.
[A1 A2 A3 A4]=dwt2(image_gray,'db4'); %computes the 1st level wavelet decomposition
image_vector = A1(:); % converts the matrix to column vector.
training_set(:,i) = image_vector; % fills the training set matrix.
end

% for class B images (59 images) where the images saved in the work space in the form of
'LB_'i'.jpg' where I is the image number.
for i=1:1:59
image_RGB = imread(['LB_',num2str(i),'.jpg']); %converts the image to 3D-matrix.
image_gray = image_RGB(:,:,1); %converts the 3D-matrix to 2D-matrix.
[A1 A2 A3 A4]=dwt2(image_gray,'db4'); %computes the 1st level wavelet decomposition
image_vector = A1(:); % converts the matrix to column vector.
training_set(:,i+59) = image_vector; % fills the training set matrix.
end

% for class C images (59 images) where the images saved in the work space in the form of
'LC_'i'.jpg' where I is the image number.
for i=1:1:59
image_RGB = imread(['LC_',num2str(i),'.jpg']); %converts the image to 3D-matrix.
image_gray = image_RGB(:,:,1); %converts the 3D-matrix to 2D-matrix.
[A1 A2 A3 A4]=dwt2(image_gray,'db4'); %computes the 1st level wavelet decomposition
image_vector = A1(:); % converts the matrix to column vector.
training_set(:,i+59) = image_vector; % fills the training set matrix.
end
```

### Training set Matrix (for two levels WAVELET decomposition)

```
% for class A images (59 images) where the images saved in the work space in the form of
'LA_'i'.jpg' where I is the image number.
for i=1:1:59
image_RGB = imread(['LA_',num2str(i),'.jpg']); %converts the image to 3D-matrix.
image_gray = image_RGB(:,:,1); %converts the 3D-matrix to 2D-matrix.
[A1 A2 A3 A4]=dwt2(image_gray,'db4'); %computes the 1st level wavelet decomposition
[B1 B2 B3 B4]=dwt2(A1,'db4'); %computes the 2nd level wavelet decomposition
image_vector = B1(:); % converts the matrix to column vector.
training_set(:,i) = image_vector; % fills the training set matrix.
end

% for class B images (59 images) where the images saved in the work space in the form of
'LB_'i'.jpg' where I is the image number.
for i=1:1:59
image_RGB = imread(['LB_',num2str(i),'.jpg']); %converts the image to 3D-matrix.
image_gray = image_RGB(:,:,1); %converts the 3D-matrix to 2D-matrix.
[A1 A2 A3 A4]=dwt2(image_gray,'db4'); %computes the 1st level wavelet decomposition
[B1 B2 B3 B4]=dwt2(A1,'db4'); %computes the 2nd level wavelet decomposition
image_vector = B1(:); % converts the matrix to column vector.
training_set(:,i+59) = image_vector; % fills the training set matrix.
end

% for class C images (59 images) where the images saved in the work space in the form of
'LC_'i'.jpg' where I is the image number.
for i=1:1:59
image_RGB = imread(['LC_',num2str(i),'.jpg']); %converts the image to 3D-matrix.
image_gray = image_RGB(:,:,1); %converts the 3D-matrix to 2D-matrix.
[A1 A2 A3 A4]=dwt2(image_gray,'db4'); %computes the 1st level wavelet decomposition
[B1 B2 B3 B4]=dwt2(A1,'db4'); %computes the 2nd level wavelet decomposition
image_vector = B1(:); % converts the matrix to column vector.
training_set(:,i+59) = image_vector; % fills the training set matrix.
end
```

### Training set Matrix (for three levels WAVELET decomposition)

```
% for class A images (59 images) where the images saved in the work space in the form of  
'LA_'i'.jpg' where I is the image number.  
for i=1:1:59  
    image_RGB = imread(['LA_',num2str(i),'.jpg']); %converts the image to 3D-matrix.  
    image_gray = image_RGB(:,:,1); %converts the 3D-matrix to 2D-matrix.  
    [A1 A2 A3 A4]=dwt2(image_gray,'db4'); %computes the 1st level wavelet decomposition  
    [B1 B2 B3 B4]=dwt2(A1,'db4'); %computes the 2nd level wavelet decomposition  
    [C1 C2 C3 C4]=dwt2(B1,'db4'); %computes the 3rd level wavelet decomposition  
    image_vector = C1(:); % converts the matrix to column vector.  
    training_set(:,i) = image_vector; % fills the training set matrix.  
end
```

```
% for class B images (59 images) where the images saved in the work space in the form of  
'LB_'i'.jpg' where I is the image number.  
for i=1:1:59  
    image_RGB = imread(['LB_',num2str(i),'.jpg']); %converts the image to 3D-matrix.  
    image_gray = image_RGB(:,:,1); %converts the 3D-matrix to 2D-matrix.  
    [A1 A2 A3 A4]=dwt2(image_gray,'db4'); %computes the 1st level wavelet decomposition  
    [B1 B2 B3 B4]=dwt2(A1,'db4'); %computes the 2nd level wavelet decomposition  
    [C1 C2 C3 C4]=dwt2(B1,'db4'); %computes the 3rd level wavelet decomposition  
    image_vector = C1(:); % converts the matrix to column vector.  
    training_set(:,i+59) = image_vector; % fills the training set matrix.  
end
```

```
% for class C images (59 images) where the images saved in the work space in the form of  
'LC_'i'.jpg' where I is the image number.  
for i=1:1:59  
    image_RGB = imread(['LC_',num2str(i),'.jpg']); %converts the image to 3D-matrix.  
    image_gray = image_RGB(:,:,1); %converts the 3D-matrix to 2D-matrix.  
    [A1 A2 A3 A4]=dwt2(image_gray,'db4'); %computes the 1st level wavelet decomposition  
    [B1 B2 B3 B4]=dwt2(A1,'db4'); %computes the 2nd level wavelet decomposition  
    [C1 C2 C3 C4]=dwt2(B1,'db4'); %computes the 3rd level wavelet decomposition  
    image_vector = C1(:); % converts the matrix to column vector.  
    training_set(:,i+59) = image_vector; % fills the training set matrix.  
end
```



### Training set Matrix (for four levels WAVELET decomposition)

```
% for class A images (59 images) where the images saved in the work space in the form of
'LA_'i'.jpg' where I is the image number.
for i=1:1:59
image_RGB = imread(['LA_',num2str(i),'.jpg']); %converts the image to 3D-matrix.
image_gray = image_RGB(:,:,1); %converts the 3D-matrix to 2D-matrix.
[A1 A2 A3 A4]=dwt2(image_gray,'db4'); %computes the 1st level wavelet decomposition
[B1 B2 B3 B4]=dwt2(A1,'db4'); %computes the 2nd level wavelet decomposition
[C1 C2 C3 C4]=dwt2(B1,'db4'); %computes the 3rd level wavelet decomposition
[D1 D2 D3 D4]=dwt2(C1,'db4'); %computes the 4th level wavelet decomposition
image_vector = D1(:); % converts the matrix to column vector.
training_set(:,i) = image_vector; % fills the training set matrix.
end
```

```
% for class B images (59 images) where the images saved in the work space in the form of
'LB_'i'.jpg' where I is the image number.
for i=1:1:59
image_RGB = imread(['LB_',num2str(i),'.jpg']); %converts the image to 3D-matrix.
image_gray = image_RGB(:,:,1); %converts the 3D-matrix to 2D-matrix.
[A1 A2 A3 A4]=dwt2(image_gray,'db4'); %computes the 1st level wavelet decomposition
[B1 B2 B3 B4]=dwt2(A1,'db4'); %computes the 2nd level wavelet decomposition
[C1 C2 C3 C4]=dwt2(B1,'db4'); %computes the 3rd level wavelet decomposition
[D1 D2 D3 D4]=dwt2(C1,'db4'); %computes the 4th level wavelet decomposition
image_vector = D1(:); % converts the matrix to column vector.
training_set(:,i+59) = image_vector; % fills the training set matrix.
end
```

```
% for class C images (59 images) where the images saved in the work space in the form of
'LC_'i'.jpg' where I is the image number.
for i=1:1:59
image_RGB = imread(['LC_',num2str(i),'.jpg']); %converts the image to 3D-matrix.
image_gray = image_RGB(:,:,1); %converts the 3D-matrix to 2D-matrix.
[A1 A2 A3 A4]=dwt2(image_gray,'db4'); %computes the 1st level wavelet decomposition
[B1 B2 B3 B4]=dwt2(A1,'db4'); %computes the 2nd level wavelet decomposition
[C1 C2 C3 C4]=dwt2(B1,'db4'); %computes the 3rd level wavelet decomposition
[D1 D2 D3 D4]=dwt2(C1,'db4'); %computes the 4th level wavelet decomposition
image_vector = D1(:); % converts the matrix to column vector.
training_set(:,i+59) = image_vector; % fills the training set matrix.
end
```



### Training set Matrix (for five levels WAVELET decomposition)

```
% for class A images (59 images) where the images saved in the work space in the form of
'LA_'i'.jpg' where I is the image number.
for i=1:1:59
image_RGB = imread(['LA_',num2str(i),'.jpg']); %converts the image to 3D-matrix.
image_gray = image_RGB(:,:,1); %converts the 3D-matrix to 2D-matrix.
[A1 A2 A3 A4]=dwt2(image_gray,'db4'); %computes the 1st level wavelet decomposition
[B1 B2 B3 B4]=dwt2(A1,'db4'); %computes the 2nd level wavelet decomposition
[C1 C2 C3 C4]=dwt2(B1,'db4'); %computes the 3rd level wavelet decomposition
[D1 D2 D3 D4]=dwt2(C1,'db4'); %computes the 4th level wavelet decomposition
[E1 E2 E3 E4]=dwt2(D1,'db4'); %computes the 5th level wavelet decomposition
image_vector = E1(:); % converts the matrix to column vector.
training_set(:,i) = image_vector; % fills the training set matrix.
end
```

```
% for class B images (59 images) where the images saved in the work space in the form of
'LB_'i'.jpg' where I is the image number.
for i=1:1:59
image_RGB = imread(['LB_',num2str(i),'.jpg']); %converts the image to 3D-matrix.
image_gray = image_RGB(:,:,1); %converts the 3D-matrix to 2D-matrix.
[A1 A2 A3 A4]=dwt2(image_gray,'db4'); %computes the 1st level wavelet decomposition
[B1 B2 B3 B4]=dwt2(A1,'db4'); %computes the 2nd level wavelet decomposition
[C1 C2 C3 C4]=dwt2(B1,'db4'); %computes the 3rd level wavelet decomposition
[D1 D2 D3 D4]=dwt2(C1,'db4'); %computes the 4th level wavelet decomposition
[E1 E2 E3 E4]=dwt2(D1,'db4'); %computes the 5th level wavelet decomposition
image_vector = E1(:); % converts the matrix to column vector.
training_set(:,i+59) = image_vector; % fills the training set matrix.
end
```

```
% for class C images (59 images) where the images saved in the work space in the form of
'LC_'i'.jpg' where I is the image number.
for i=1:1:59
image_RGB = imread(['LC_',num2str(i),'.jpg']); %converts the image to 3D-matrix.
image_gray = image_RGB(:,:,1); %converts the 3D-matrix to 2D-matrix.
[A1 A2 A3 A4]=dwt2(image_gray,'db4'); %computes the 1st level wavelet decomposition
[B1 B2 B3 B4]=dwt2(A1,'db4'); %computes the 2nd level wavelet decomposition
[C1 C2 C3 C4]=dwt2(B1,'db4'); %computes the 3rd level wavelet decomposition
[D1 D2 D3 D4]=dwt2(C1,'db4'); %computes the 4th level wavelet decomposition
[E1 E2 E3 E4]=dwt2(D1,'db4'); %computes the 5th level wavelet decomposition
image_vector = E1(:); % converts the matrix to column vector.
training_set(:,i+59) = image_vector; % fills the training set matrix.
end
```

### Network initiation for Competition Layer

```
ranges=(minmax(training_set));  
%minmax will take the maximum value and the minimum value of the (training_set)  
matrix rows, and put these values in the (ranges) matrix.  
  
net=newc(Ranges,3);  
%Since the number of classes in our design is three, so the number of neurons in the  
competitive layer will always be (3) regardless the number of input.
```

### Network initiation for LVQ Layer

```
ranges=(minmax(training_set));  
%minmax will take the maximum value and the minimum value of the (training_set)  
matrix rows, and put these values in the (ranges) matrix.  
  
net = newlvq(minmax(training_set),6,[1/3 1/3 1/3],0.001,  
'learnlv1');  
%Minmax(training_set) Equal to the ranges matrix of the input training set.  
%6 Is the number of subclasses  
%[1/3 1/3 1/3] The percentage of each class training set.  
%0.001 The training rate  
%'learnlv1' The learning function
```

### Network initiation for LVQ Layer

```
net.trainparam.epochs=1000;  
%Defines the maximum number of times the complete data set may be used  
for training,  
  
net.trainparam.show=100;  
%Defines the time between status reports of the training function.  
  
net.trainparam.goal=1e-2;  
%Defines the maximum error accepted in the training process.  
  
net=train(net,trainin_set,targets);  
%This command will train the network with the data matrix as its input, and  
the target matrix as its output in case of the Competitive network we omit  
the targets.
```

### Target matrix for Feedforward classification

```
%Class one outputs:
targets(1,1:20)=1;
targets(1,21:60)=0;

%Class two outputs:
targets(2,1:20)=0;
targets(2,21:40)=1;
targets(2,41:60)=0;

%Class three outputs:
targets(3,1:40)=0;
targets(3,41:60)=1;
```

### Target matrix for LVQ classification

```
targets_classes_indices(1,1:1:20) = 1;
%sets the first 59 elements in the target row vector to one which means the first class.

targets_classes_indices (1,21:1:40) = 2;
%sets the second 59 elements in the target row vector to two which means the second
class.

targets_classes_indices (1,41:1:60) = 3;
%sets the third 59 elements in the target row vector to three which means the third class.

targets_classes_vector = ind2vec(targets_classes_indices);
%converts the targets vector from indices form into vector form.
```

### Network initiation for Feedforward

```
ranges=(minmax(training_set));
%minmax will take the maximum value and the minimum value of the (training_set)
matrix rows, and put these values in the (ranges) matrix.

net=newff(ranges,[265216,3],{'tansig','purelin'},'trainlm');
%This command will create the neural network with all the specification mentioned
above in the network description.
```



Full code for feedforward ( for original image)

```
net=network;
net.numInputs = 1;
net.numLayers = 2;
net.inputs{1}.size = 530432;
for i=1:1:59
    image_RGB = imread(['LA_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    image_vector = image_gray(:);
    training_set(:,i+59)= image_vector;
end
for i=1:1:59
    image_RGB = imread(['LB_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    image_vector = image_gray(:);
    training_set(:,i+59)= image_vector;
end
for i=1:1:59
    image_RGB = imread(['LC_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    image_vector = image_gray(:);
    training_set(:,i+59)= image_vector;
end
%class one outputs
targets(1,1:20) = 1;
targets(1,21:60) = 0;
%class two outputs
targets(2,1:20) = 0;
targets(2,21:40) = 1;
targets(2,41:60) = 0;
%class three outputs
targets(3,1:40) = 0;
targets(3,41:60) = 1;
ranges = (minmax(training_set));
net = newff(ranges,[265216,3],{'tansig','purelin'},'trainlm');
net.trainparam.epochs = 1000;
net.trainparam.show = 100;
net.trainparam.goal = 1e-2;
net=train(net,[0 1],[0 1]);
```

Full code for feedforward (using one level wavelet decomposition)

```
net=network;
net.numInputs = 1;
net.numLayers = 2;
net.inputs{1}.size = 135148;
for i =1:1:59
    image_RGB = imread(['LA_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]= dwt2(image_gray,'db4');
    image_vector = A1(:);
    training_set(:,i+59)= image_vector
end
for i =1:1:59
    image_RGB = imread(['LB_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]= dwt2(image_gray,'db4');
    image_vector = A1(:);
    training_set(:,i+59)= image_vector;
end
for i =1:1:59
    image_RGB = imread(['LC_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]= dwt2(image_gray,'db4');
    image_vector = A1(:);
    training_set(:,i+59)= image_vector;
end
%class one outputs
targets(1,1:20) = 1;
targets(1,21:60) = 0;
%class two outputs
targets(2,1:20) = 0;
targets(2,21:40) = 1;
targets(2,41:60) = 0;
%class three outputs
targets(3,1:40) = 0;
targets(3,41:60) = 1;
ranges = (minmax(training_set));
net = newff(ranges,[67800,3],{'tansig','purelin'},'trainlm');
net.trainparam.epochs = 1000
net.trainparam.show = 100
net.trainparam.goal = 1e-2
pause
net=train(net,training_set,targets)
```

Full code for feedforward (using two level wavelet decomposition)

```
net=network;
net.numInputs = 1;
net.numLayers = 2;
net.inputs{1}.size = 35037;
for i =1:1:59
    image_RGB = imread(['LA_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]= dwt2(image_gray,'db4');
    [B1 B2 B3 B4]= dwt2(A1,'db4');
    image_vector = B1(:);
    training_set(:,i+59) = image_vector;
end
for i =1:1:59
    image_RGB = imread(['LB_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]= dwt2(image_gray,'db4');
    [B1 B2 B3 B4]= dwt2(A1,'db4');
    image_vector = B1(:);
    training_set(:,i+59)= image_vector;
end
for i =1:1:59
    image_RGB = imread(['LC_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]= dwt2(image_gray,'db4');
    [B1 B2 B3 B4]= dwt2(A1,'db4');
    image_vector = B1(:);
    training_set(:,i+59)= image_vector;
end
%class one outputs
targets(1,1:20) = 1;
targets(1,21:60) = 0;
%class two outputs
targets(2,1:20) = 0;
targets(2,21:40) = 1;
targets(2,41:60) = 0;
%class three outputs
targets(3,1:40) = 0;
targets(3,41:60) = 1;
ranges = (minmax(training_set));
net = newff(ranges,[17710,3],{'tansig','purelin'},'trainlm');
net.trainparam.epochs = 1000
net.trainparam.show = 100
net.trainparam.goal = 1e-2
pause
net=train(net,training_set,targets)
```



Full code for feedforward (using three level wavelet decomposition)

```
net=network;
net.numInputs = 1;
net.numLayers = 2;
net.inputs{1}.size = 9440;
for i=1:1:59
    image_RGB = imread(['LA_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]= dwt2(image_gray,'db4');
    [B1 B2 B3 B4]= dwt2(A1,'db4');
    [C1 C2 C3 C4]= dwt2(B1,'db4');
    image_vector = C1(:);
    training_set(:,i) = image_vector;
end
for i=1:1:59
    image_RGB = imread(['LB_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]= dwt2(image_gray,'db4');
    [B1 B2 B3 B4]= dwt2(A1,'db4');
    [C1 C2 C3 C4]= dwt2(B1,'db4');
    image_vector = C1(:);
    training_set(:,i+59)= image_vector;
end
for i=1:1:59
    image_RGB = imread(['LC_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]= dwt2(image_gray,'db4');
    [B1 B2 B3 B4]= dwt2(A1,'db4');
    [C1 C2 C3 C4]= dwt2(B1,'db4');
    image_vector = C1(:);
    training_set(:,i+59)= image_vector;
end
%class one outputs
targets(1,1:20) = 1;
targets(1,21:60) = 0;
%class two outputs
targets(2,1:20) = 0;
targets(2,21:40) = 1;
targets(2,41:60) = 0;
%class three outputs
targets(3,1:40) = 0;
targets(3,41:60) = 1;
ranges = (minmax(training_set));
net = newff(ranges,[4820,3],{'tansig','purelin'},'trainlm');
net.trainparam.epochs = 1000
net.trainparam.show = 100
net.trainparam.goal = 1e-2
pause
net=train(net,training_set,targets)
```

Full code for feedforward (using four level wavelet decomposition)

```
net=network;
net.numInputs = 1;
net.numLayers = 2;
net.inputs{1}.size = 2666;
for i=1:1:59
    image_RGB = imread(['LA_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]= dwt2(image_gray,'db4');
    [B1 B2 B3 B4]= dwt2(A1,'db4');
    [C1 C2 C3 C4]= dwt2(B1,'db4');
    [D1 D2 D3 D4]= dwt2(C1,'db4');
    image_vector = D1(:);
    training_set(:,i+59) = image_vector;
end
for i=1:1:59
    image_RGB = imread(['LB_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]= dwt2(image_gray,'db4');
    [B1 B2 B3 B4]= dwt2(A1,'db4');
    [C1 C2 C3 C4]= dwt2(B1,'db4');
    [D1 D2 D3 D4]= dwt2(C1,'db4');
    image_vector = D1(:);
    training_set(:,i+59)= image_vector;
end
for i=1:1:59
    image_RGB = imread(['LC_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]= dwt2(image_gray,'db4');
    [B1 B2 B3 B4]= dwt2(A1,'db4');
    [C1 C2 C3 C4]= dwt2(B1,'db4');
    [D1 D2 D3 D4]= dwt2(C1,'db4');
    image_vector = D1(:);
    training_set(:,i+59)= image_vector;
end
%class one outputs
targets(1,1:20) = 1;
targets(1,21:60) = 0;
%class two outputs
targets(2,1:20) = 0;
targets(2,21:40) = 1;
targets(2,41:60) = 0;
%class three outputs
targets(3,1:40) = 0;
targets(3,41:60) = 1;
ranges = (minmax(training_set));
net = newff(ranges,[1440c,3],{'tansig','purelin'},'trainlm');
net.trainparam.epochs = 1000
net.trainparam.show = 100
```

```
net.trainparam.goal = 1e-2
net=train(net,training_set,targets)
```

Full code for feedforward network (using five level wavelet decomposition)

```
net=network;
net.numInputs = 1;
net.numLayers = 2;
net.inputs{1}.size = 2666;
for i =1:1:59
    image_RGB = imread(['LA_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]= dwt2(image_gray,'db4');
    [B1 B2 B3 B4]= dwt2(A1,'db4');
    [C1 C2 C3 C4]= dwt2(B1,'db4');
    [D1 D2 D3 D4]= dwt2(C1,'db4');
    [E1 E2 E3 E4]= dwt2(D1,'db4');
    image_vector = E1(:);
    training_set(:,i+59) = image_vector;
end
for i =1:1:59
    image_RGB = imread(['LB_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]= dwt2(image_gray,'db4');
    [B1 B2 B3 B4]= dwt2(A1,'db4');
    [C1 C2 C3 C4]= dwt2(B1,'db4');
    [D1 D2 D3 D4]= dwt2(C1,'db4');
    [E1 E2 E3 E4]= dwt2(D1,'db4');
    image_vector = E1(:);
    training_set(:,i+59)= image_vector;
end
for i =1:1:59
    image_RGB = imread(['LC_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]= dwt2(image_gray,'db4');
    [B1 B2 B3 B4]= dwt2(A1,'db4');
    [C1 C2 C3 C4]= dwt2(B1,'db4');
    [D1 D2 D3 D4]= dwt2(C1,'db4');
    [E1 E2 E3 E4]= dwt2(D1,'db4');
    image_vector = E1(:);
    training_set(:,i+59)= image_vector;
end
%class one outputs
targets(1,1:20) = 1;
targets(1,21:60) = 0;
%class two outputs
targets(2,1:20) = 0;
targets(2,21:40) = 1;
targets(2,41:60) = 0;
%class three outputs
```

```

targets(3,1:40) = 0;
targets(3,41:60) = 1;
ranges = (minmax(training_set));
net = newff(ranges,[1440c,3],{'tansig','purelin'},'trainlm');
net.trainparam.epochs = 1000
net.trainparam.show = 100
net.trainparam.goal = 1e-2
pause
net=train(net,training_set,targets)

```

Full Code using the competitive network ( for original image)

```

for i=1:1:59
    image_RGB = imread(['LA_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    image_vector = image_gray(:);
    training_set(:,i)= image_vector;
end
for i=1:1:59
    image_RGB = imread(['LB_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    image_vector = image_gray(:);
    training_set(:,i+59)= image_vector;
end
for i=1:1:59
    image_RGB = imread(['LC_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    image_vector = image_gray(:);
    training_set(:,i+59)= image_vector;
end
ranges=[0 1];
net=newc(ranges,3);
net.trainparam.epochs=10000;
net=train(net,training_set);

```

Full Code using the competitive network(using one level wavelet decomposition)

```

for i=1:1:59
    image_RGB = imread(['LA_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]=dwt2(image_gray,'db4');
    image_vector = A1(:);
    training_set(:,i+59)= image_vector;
end
for i=1:1:59
    image_RGB = imread(['LB_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]=dwt2(image_gray,'db4');

```



```

    image_vector = A1(:);
    training_set(:,i+59)= image_vector;
end
for i =1:1:59
    image_RGB = imread(['LC_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]=dwt2(image_gray,'db4');
    image_vector = A1(:);
    training_set(:,i+59)= image_vector;
end
ranges=(minmax(training_set));
net=newc(ranges,3);
net.trainparam.epochs=10000;
net=train(net,training_set);

```

Full Code using the competitive network ( using two level wavelet decomposition)

```

for i =1:1:59
    image_RGB = imread(['LA_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]=dwt2(image_gray,'db4');
    [B1 B2 B3 B4]=dwt2(A1,'db4');
    image_vector = B1(:);
    training_set(:,i+59)= image_vector;
end
for i =1:1:59
    image_RGB = imread(['LB_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]=dwt2(image_gray,'db4');
    [B1 B2 B3 B4]=dwt2(A1,'db4');
    image_vector = B1(:);
    training_set(:,i+59)= image_vector;
end
for i =1:1:5
    image_RGB = imread(['LC_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]=dwt2(image_gray,'db4');
    [B1 B2 B3 B4]=dwt2(A1,'db4');
    image_vector = B1(:);
    training_set(:,i+59)= image_vector;
end
ranges=(minmax(training_set));
net=newc(ranges,3);
net.trainparam.epochs=10000;
net=train(net,training_set);

```

Full Code using the competitive network(using three level wavelet decomposition)

```
for i = 1:1:59
    image_RGB = imread(['LA_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]=dwt2(image_gray,'db4');
    [B1 B2 B3 B4]=dwt2(A1,'db4');
    [C1 C2 C3 C4]=dwt2(B1,'db4');
    image_vector = C1(:);
    training_set(:,i+59)= image_vector;
end
for i = 1:1:59
    image_RGB = imread(['LB_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]=dwt2(image_gray,'db4');
    [B1 B2 B3 B4]=dwt2(A1,'db4');
    [C1 C2 C3 C4]=dwt2(B1,'db4');
    image_vector = C1(:);
    training_set(:,i+59)= image_vector;
end
for i = 1:1:59
    image_RGB = imread(['LC_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]=dwt2(image_gray,'db4');
    [B1 B2 B3 B4]=dwt2(A1,'db4');
    [C1 C2 C3 C4]=dwt2(B1,'db4');
    image_vector = C1(:);
    training_set(:,i+59)= image_vector;
end
ranges=(minmax(training_set));
net=newc(ranges,3);
net.trainparam.epochs=10000;
net=train(net,training_set);
```

Full Code using the competitive network (using four level wavelet decomposition)

```
for i = 1:1:59
    image_RGB = imread(['LA_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]=dwt2(image_gray,'db4');
    [B1 B2 B3 B4]=dwt2(A1,'db4');
    [C1 C2 C3 C4]=dwt2(B1,'db4');
    [D1 D2 D3 D4]=dwt2(C1,'db4');
    image_vector = D1(:);
    training_set(:,i+59)= image_vector;
end
for i = 1:1:59
    image_RGB = imread(['LB_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]=dwt2(image_gray,'db4');
```

```

[B1 B2 B3 B4]=dwt2(A1,'db4');
[C1 C2 C3 C4]=dwt2(B1,'db4');
[D1 D2 D3 D4]=dwt2(C1,'db4');
image_vector = D1(:);
training_set(:,i+59)= image_vector;
end
for i =1:1:59
    image_RGB = imread(['LC_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]=dwt2(image_gray,'db4');
    [B1 B2 B3 B4]=dwt2(A1,'db4');
    [C1 C2 C3 C4]=dwt2(B1,'db4');
    [D1 D2 D3 D4]=dwt2(C1,'db4');
    image_vector = D1(:);
    training_set(:,i+59)= image_vector;
end
ranges=(minmax(training_set));
net=newc(ranges,3);
net.trainparam.epochs=10000;
net=train(net,training_set);

```

Full Code using the competitive network (using five level wavelet decomposition)

```

for i =1:1:59
    image_RGB = imread(['LA_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]=dwt2(image_gray,'db4');
    [B1 B2 B3 B4]=dwt2(A1,'db4');
    [C1 C2 C3 C4]=dwt2(B1,'db4');
    [D1 D2 D3 D4]=dwt2(C1,'db4');
    [E1 E2 E3 E4]=dwt2(D1,'db4');
    image_vector = E1(:);
    training_set(:,i+59)= image_vector;
end
for i =1:1:59
    image_RGB = imread(['LB_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]=dwt2(image_gray,'db4');
    [B1 B2 B3 B4]=dwt2(A1,'db4');
    [C1 C2 C3 C4]=dwt2(B1,'db4');
    [D1 D2 D3 D4]=dwt2(C1,'db4');
    [E1 E2 E3 E4]=dwt2(D1,'db4');
    image_vector = E1(:);
    training_set(:,i+59)= image_vector;
end
for i =1:1:59
    image_RGB = imread(['LC_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]=dwt2(image_gray,'db4');
    [B1 B2 B3 B4]=dwt2(A1,'db4');

```

```

[C1 C2 C3 C4]=dwt2(B1,'db4');
[D1 D2 D3 D4]=dwt2(C1,'db4');
[E1 E2 E3 E4]=dwt2(D1,'db4');
image_vector = E1(:);
training_set(:,i+59)= image_vector;
end
ranges=(minmax(training_set));
net=newc(ranges,3);
net.trainparam.epochs=10000;
net=train(net,training_set);

```

Full code using LVQ ( for original image)

```

for i=1:1:59
    image_RGB = imread(['LA_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    image_vector = image_gray(:);
    training_set(:,i+59)= image_vector;
end
for i=1:1:59
    image_RGB = imread(['LB_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    image_vector = image_gray(:);
    training_set(:,i+59)= image_vector;
end
for i=1:1:59
    image_RGB = imread(['LC_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    image_vector = image_gray(:);
    training_set(:,i+59)= image_vector;
end
targets_classes_indices(1,1:1:20) = 1;
targets_classes_indices(1,21:1:40) = 2;
targets_classes_indices(1,41:1:60) = 3;
targets_classes_vector = ind2vec(targets_classes_indices);
targets=full(targets_classes_vector);
ranges=[530432*118];
net = newlvq(minmax(ranges),6,[1/3 1/3 1/3],0.001,'learnlv1');
net.trainparam.goal = 0.1
net.trainparam.epochs = 1000;
net = train(net,training_set,targets_classes_vector)

```

Full code using LVQ ( using one level wavelet decomposition )

```

for i=1:1:59
    image_RGB = imread(['LA_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]=dwt2(image_gray,'db4');
    image_vector = A1(:);

```



```

training_set(:,i+59)= image_vector;
end
for i =1:1:59
    image_RGB = imread(['LB_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]=dwt2(image_gray,'db4');
    image_vector = A1(:);
    training_set(:,i+59)= image_vector;
end
for i =1:1:59
    image_RGB = imread(['LC_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]=dwt2(image_gray,'db4');
    image_vector = A1(:);
    training_set(:,i+59)= image_vector;
end

targets_classes_indices(1,1:1:20)=1;
targets_classes_indices(1,21:1:40)=2;
targets_classes_indices(1,41:1:60)=3;
targets=full(targets_classes_indices);
pause
targets_classes_vector = ind2vec(targets_classes_indices);
net=newlvq(minmax(training_set),6,[1/3 1/3 1/3],0.001,'learnlv1');
net.trainparam.goal = 0.1;
net.trainparam.epochs = 1000;
pause
net = train(net,training_set,targets_classes_vector)

```

Full code using LVQ ( using two level wavelet decomposition )

```

for i =1:1:59
    image_RGB = imread(['LA_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]=dwt2(image_gray,'db4');
    [B1 B2 B3 B4]=dwt2(A1,'db4');
    image_vector = B1(:);
    training_set(:,i+59)= image_vector;
end
for i =1:1:59
    image_RGB = imread(['LB_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]=dwt2(image_gray,'db4');
    [B1 B2 B3 B4]=dwt2(A1,'db4');
    image_vector = B1(:);
    training_set(:,i+59)= image_vector;
end
for i =1:1:59
    image_RGB = imread(['LC_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);

```

```

[A1 A2 A3 A4]=dwt2(image_gray,'db4');
[B1 B2 B3 B4]=dwt2(A1,'db4');
image_vector = B1(:);
training_set(:,i+59)= image_vector;
end
targets_classes_indices(1,1:1:20)=1;
targets_classes_indices(1,21:1:40)=2;
targets_classes_indices(1,41:1:60)=3;
targets=full(targets_classes_indices);
targets_classes_vector = ind2vec(targets_classes_indices);
net=newlvq(minmax(training_set),6,[1/3 1/3 1/3],0.001,'learnlv1');
net.trainparam.goal = 0.1;
net.trainparam.epochs = 1000;
net = train(net,training_set,targets_classes_indices)

```

Full code using LVQ ( using three level wavelet decomposition )

```

for i=1:1:59
    image_RGB = imread(['LA_',num2str(i),'jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]=dwt2(image_gray,'db4');
    [B1 B2 B3 B4]=dwt2(A1,'db4');
    [C1 C2 C3 C4]=dwt2(B1,'db4');
    image_vector = C1(:);
    training_set(:,i+59)= image_vector;
end
for i=1:1:59
    image_RGB = imread(['LB_',num2str(i),'jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]=dwt2(image_gray,'db4');
    [B1 B2 B3 B4]=dwt2(A1,'db4');
    [C1 C2 C3 C4]=dwt2(B1,'db4');
    image_vector = C1(:);
    training_set(:,i+59)= image_vector;
end
for i=1:1:59
    image_RGB = imread(['LC_',num2str(i),'jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]=dwt2(image_gray,'db4');
    [B1 B2 B3 B4]=dwt2(A1,'db4');
    [C1 C2 C3 C4]=dwt2(B1,'db4');
    image_vector = C1(:);
    training_set(:,i+59)= image_vector;
end

targets_classes_indices(1,1:1:20)=1;
targets_classes_indices(1,21:1:40)=2;
targets_classes_indices(1,41:1:60)=3;
targets=full(targets_classes_indices);
targets_classes_vector = ind2vec(targets_classes_indices);

```

```

net=newlvq(minmax(training_set),6,[1/3 1/3 1/3],0.001,'learnlv1');
net.trainparam.goal = 0.1;
net.trainparam.epochs = 1000;
net = train(net,training_set,targets_classes_indices)

```

Full code using LVQ ( using four level wavelet decomposition )

```

for i =1:1:59
    image_RGB = imread(['LA_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]=dwt2(image_gray,'db4');
    [B1 B2 B3 B4]=dwt2(A1,'db4');
    [C1 C2 C3 C4]=dwt2(B1,'db4');
    [D1 D2 D3 D4]=dwt2(C1,'db4');
    image_vector = D1(:);
    training_set(:,i+59)= image_vector;
end
for i =1:1:59
    image_RGB = imread(['LB_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]=dwt2(image_gray,'db4');
    [B1 B2 B3 B4]=dwt2(A1,'db4');
    [C1 C2 C3 C4]=dwt2(B1,'db4');
    [D1 D2 D3 D4]=dwt2(C1,'db4');
    image_vector = D1(:);
    training_set(:,i+59)= image_vector;
end
for i =1:1:59
    image_RGB = imread(['LC_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]=dwt2(image_gray,'db4');
    [B1 B2 B3 B4]=dwt2(A1,'db4');
    [C1 C2 C3 C4]=dwt2(B1,'db4');
    [D1 D2 D3 D4]=dwt2(C1,'db4');
    image_vector = D1(:);
    training_set(:,i+59)= image_vector;
end
targets_classes_indices(1,1:1:20)=1;
targets_classes_indices(1,21:1:40)=2;
targets_classes_indices(1,41:1:60)=3;
targets=full(targets_classes_indices);
targets_classes_vector = ind2vec(targets_classes_indices);
net=newlvq(minmax(training_set),6,[1/3 1/3 1/3],0.001,'learnlv1');
net.trainparam.goal = 0.1;
net.trainparam.epochs = 1000;
net = train(net,training_set,targets_classes_indices)

```



Full code using LVQ ( using five level wavelet decomposition )

```
for i=1:1:59
    image_RGB = imread(['LA_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]=dwt2(image_gray,'db4');
    [B1 B2 B3 B4]=dwt2(A1,'db4');
    [C1 C2 C3 C4]=dwt2(B1,'db4');
    [D1 D2 D3 D4]=dwt2(C1,'db4');
    [E1 E2 E3 E4]=dwt2(D1,'db4');
    image_vector = E1(:);
    training_set(:,i+59)= image_vector;
end
for i=1:1:59
    image_RGB = imread(['LB_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]=dwt2(image_gray,'db4');
    [B1 B2 B3 B4]=dwt2(A1,'db4');
    [C1 C2 C3 C4]=dwt2(B1,'db4');
    [D1 D2 D3 D4]=dwt2(C1,'db4');
    [E1 E2 E3 E4]=dwt2(D1,'db4');
    image_vector = E1(:);
    training_set(:,i+59)= image_vector;
end
for i=1:1:59
    image_RGB = imread(['LC_',num2str(i),'.jpg']);
    image_gray = image_RGB(:,:,1);
    [A1 A2 A3 A4]=dwt2(image_gray,'db4');
    [B1 B2 B3 B4]=dwt2(A1,'db4');
    [C1 C2 C3 C4]=dwt2(B1,'db4');
    [D1 D2 D3 D4]=dwt2(C1,'db4');
    [E1 E2 E3 E4]=dwt2(D1,'db4');
    image_vector = E1(:);
    training_set(:,i+59)= image_vector;
end
targets_classes_indices(1,1:1:20)=1;
targets_classes_indices(1,21:1:40)=2;
targets_classes_indices(1,41:1:60)=3;
targets=full(targets_classes_indices);
targets_classes_vector = ind2vec(targets_classes_indices);
net=newlvq(minmax(training_set),6,[1/3 1/3 1/3],0.001,'learnlv1');
net.trainparam.goal = 0.1;
net.trainparam.epochs = 1000;
net = train(net,training_set,targets_classes_vector)
```



## REFERENCES

- [1] <http://www.dacs.dtic.mil/techs/neural/neural2.html>
- [2] <http://www.mathworks.com/access/helpdesk/help/toolbox/wavelet/wavelet.shtml>
- [3] <http://www.mathworks.com/access/helpdesk/help/toolbox/nnet/newc.html>
- [4] <http://www.mathworks.com/access/helpdesk/help/toolbox/nnet/newff.html>
- [5] <http://www.mathworks.com/access/helpdesk/help/toolbox/nnet/newlvq.html>

## ABSTRACT

Since the beginning, humankind has sought to use elements in the surrounding environment to make life easier and the tasks at hand more efficient. In keeping with this tradition, people have toyed with and explored the concept of using machines to solve problems since ancient times, Only in this 20th century have significant advances occurred, making the possibility of an actual manifestation of artificial intelligence more and more a reality.

This project explores the theoretical and particular underpinning of Neural Networks and its applications, the reader of this project will come away with an appreciation for the basic concepts of Neural Network., and with an idea about Image classification using Artificial Neural Networks field and the use of its applications.

This projects includes three kinds of network are using for image classification, also this project is supplied with MATLAB code which help in implementing and training the neural network for image classification and the reader of this project will learn about the wavelet decomposition which help in compression or reducing the image size without losing the image data.

# Table of Contents

<b>ACKNOWLEDGEMENTS</b>	i
<b>ABSTRACT</b>	ii
<b>TABLE OF CONTENTS</b>	iii
<b>INTROUDUCTION</b>	1
<b>CHAPTER 1: ARTIFICIAL NEURAL NETWORKS</b>	
1.1 Overview	5
1.2 Introduction to Artificial Neural Network	5
1.2.1 Artificial Neural Networks	6
1.2.2 Analogy of the Brain	6
1.2.3 Artificial Neurons and how they work	7
1.3 Components of Artificial Neural Network.	9
1.3.1 Neurons	9
1.3.2 Layers	9
1.3.3 Connections (weights)	10
1.3.4 Transfer Function	10
1.4 Artificial Neural Network: Operation Mode and Training Mode	11
1.4.1 Operation Mode	11
1.4.2 Training Mode	13
1.4.2.1 Supervised Training.	13
1.4.2.2 Unsupervised Training (Adaptive Training)	14
1.5 Artificial Neural Networks versus Traditional Computing	16
1.6 The Artificial Neural Network Applications	16
1.6.1 Banking and Financial	17
1.6.2 Language Processing	17
1.6.3 Character Recognition	17
1.6.4 Image (data) Compression	18

1.6.5 Pattern Recognition	18
1.7 Neural networks structure	19
1.7.1 FeedForward-Back Propagation Neural Network	19
1.8 Summery	22

## CHAPTER 2: CLASSIFICATION METHOD

2.1 Overview	23
2.2 Statistical Methods	23
2.2.1 Maximum likelihood Classification	23
2.2.2 Minimum distance Classification	24
2.3 Classification using Artificial Neural Network	24
2.3.1 Competitive Neural Network	24
2.3.1.1 Basic Operation of Competitive Layer	25
2.3.1.2 Source of error in the competitive layer network	26
2.3.1.3 Bias Learning Rule (learncon)	26
2.3.2 Learning Vector Quantization(LVQ)	27
2.3.2.1 Architecture of Learning Vector Quantization Network	27
2.3.2.2 Training of LVQ Networks	29
2.3.2.3 Drawbacks of LVQ Networks	30
2.4 Summary	30

## CHAPTER 3: INTRODUCTION TO WAVELET ANALYSIS

3.1 Overview	32
3.2 The need of wavelet transformation	32
3.3 The Drawback of Fourier Transform	33
3.4 The solution of the limitation of the Fourier Transform	33
3.5 The appearance of Wavelet analysis	34
3.6 Wavelet Computing	35
3.6.1 The Continuous Wavelet Transform and the wavelet series	36



3.6.2	The Discrete Wavelet Transform	37
3.6.3	DWT and Filter Banks	38
3.7	Wavelet Families	40
3.8	Summary	41

## **CHAPTER 4: DESIGN OBJECTIVE AND PREPROCESSING**

4.1	Overview	42
4.2	Image characteristics	43
4.3	The MTLAB Wavelet Toolbox usage	47
4.3.1	The decomposition process	48
4.3.2	The reconstruction Process	48
4.4	Images preprocessing; the use of WAVELET transformer	49
4.5	Summary	55

## **CHAPTER 5: FEEDFORWARD DESIGN**

5.1	Overview	56
5.2	The FeedForward Construction	56
5.3	FeedForward ANN Design using the original size	59
5.4	The use of Wavelet transformation	64
5.5	The end results of FeedForward design	66
5.6	Summary	67

## **CHAPTER 6: CLASSIFICATION USING COMPETITIVE LAYER ANN**

6.1	Overview	68
6.2	Construction of Competitive Layer ANN	68
6.3	Network design for the original size images	69
6.4	Network design for reduced images size	71
6.4.1	Network design using third wavelet decomposition	72

6.4.2	Network design using fourth level wavelet decomposition	77
6.4.3	Network design using fifth level wavelet decomposition	82
6.5	Summary	87

## **Chapter 7: CLASSIFICATION USING LEARNING VECTOR QUANTIZATION**

7.1	Overview	88
7.2	Construction of LVQ ANN	88
7.3	Network design for the original size images	89
7.4	LVQ Network design for reduced images size (one wavelet level)	91
7.5	LVQ Network design for reduced images size (two levels wavelet)	94
7.6	LVQ Network design for reduced images size (three levels wavelet)	94
7.7	Network design for reduced images size (four levels wavelet)	99
7.8	Network design for reduced images size (five levels wavelet)	101
7.9	Summary	102

<b>CONCLUSION</b>	<b>103</b>
<b>LIST OF FIGURES</b>	<b>105</b>
<b>LIST OF TABLES</b>	<b>106</b>
<b>APPENDIX</b>	<b>I</b>
<b>REFERENCES</b>	<b>R</b>