# NEAR EAST UNIVERSITY

## GRADUATE SCHOOL OF APPLIED AND SOCIAL SCIENCES

# FACE RECOGNITION USING EIGENFACES BACKPROPAGATION NEURAL NETWORKS

**ALAA ADNAN ELEYAN**

**Master Thesis**

**Department of Electrical & Electronic Engineering**

**Nicosia 2004**

**Alaa Adnan Hassan Eleyan: Face Recognition Using Eigenfaces & Backpropagation Neural Networks**

**Approval of the Graduate School of Applied and Social Sciences**

**Prof. Dr. Fakhraddin Mamedov**
**Director**

**We certify this thesis is satisfactory for the award of the Degree of Master of Science in Electrical & Electronic Engineering**

**Examining Committee in charge:**

| | |
|---|---|
| **Prof. Dr. Fakhraddin Mamedov:** | Committee Chairman, Dean of Engineering Faculty, NEU |
| **Assoc. Prof. Dr. Rahib Abiyev:** | Committee Member, Chairman of Computer Engineering Department, NEU |
| **Assoc. Prof. Dr. Sameer Ikhdair:** | Committee Member, Electrical and Electronic Engineering Department, NEU |
| **Assoc. Prof. Dr. Adnan Khashman:** | Supervisor, Chairman of Electrical & Electronic Engineering Department, NEU |

# ACKNOWLEDGMENTS

First of all, I would like to thank my advisor Assoc. Prof. Dr. Adnan Khashman, who has been very helpful throughout the years, starting with his image processing and neural networks lectures, and ending up with the preparation of this thesis.

A great deal of thanks goes to my family and my big brother Emad, for their endless support and encouragement, even through the most demanding phases of this research.

Thanks to the fifteen students who helped me in preparation of my own face database.

I don't forget her to send my deep thanks to Assist. Prof. Dr. Hasan Dimirel and Sabina Hosic for their help in Matlab code.

And finally, a special thanks goes to my future wife for her enormous support and understanding throughout the past two years.

# ABSTRACT

A face authentication system based on principal component analysis and neural networks is developed in this thesis. The system consists of three stages; preprocessing, principal component analysis, and recognition. In preprocessing stage, the author captured faces of fifteen people who are NEU students. Principal component analysis is applied to find the aspects of face which are important for identification. Eigenvectors and eigenfaces are calculated from initial face image set. New faces are projected onto the space expanded by eigenfaces and represented by weighted sum of eigenfaces. These weights are used to identify the faces.

Neural network is used to recognize a face by using these weights. In this work, a large neural network was built for all people on the data base. The input face is projected onto the eigenface space first and new descriptor is obtained. The new descriptor is used as input to the neural network, trained earlier. The output with maximum value is selected and the corresponding face reported as the holder.

# TABLE OF CONTENTS

# INTRODUCTION

The face is our primary focus of attention in social intercourse, playing a major role in conveying identity and emotion. Although the ability to infer intelligence or character from facial appearance is suspect, the human ability to recognize faces is remarkable. We can recognize thousands of faces learned throughout our life time and identify familiar faces at a glance even after years of separation. This skill is quite robust, despite large changes in the visual stimulus due to viewing conditions, expression, aging, and destructions such as glasses, beards or changes in hair style.

Face recognition has become an important issue in many applications such as security, credit card verification and criminal identification. For example, the ability to model a particular face and distinguish it from a large number of stored face models would make it possible to vastly improve criminal identification. Even ability to merely detect faces, as opposed to recognizing them, can be important. Detecting faces in photographs for automating color film development can be very useful, since the effect of many enhancement and noise reduction techniques depends on the image content.

A formal method of classifying faces was first proposed by Francis Galton in 1988 [1, 2]. During the 1980's work on face recognition remained largely dormant. Since 1990's the research interest in face recognition has grown significantly as a result of the following facts:

- The increase in emphasis on civilian/commercial research projects.
- The re-emergence of neural networks classifiers with emphasis on real time computation and adaptation.
- The availability of real time hardware.
- The increasing need for surveillance related applications due to drug trafficking, terrorist activities, etc.

Although it is clear that people are good at face recognition, it is not at all obvious how faces are encoded or decoded by the human brain. Developing a

computational model of face recognition is quite difficult, because faces are complex, multidimensional visual stimulus. Therefore, face recognition is a very high level computer vision task, in which many early techniques can be involved.

The objectives of the work presented within this thesis are to:

- Develop a software capable of recognize faces using eigenfaces approach.
- In order to train and test the software the author built his own faces database.
- Use neural networks in recognition instead of Euclidian distance method.

This thesis is organized in the following manner: Chapter one has general information and basic concepts of pattern and face recognition. Chapter two explains the neural networks concept, the biological inspiration, architecture, and how it works. The back propagation method explained in details. Chapter three gives basic information about the proposed face recognition method eigenfaces with listing of the mathematical equations to apply the method. Chapter 4 contains the neural network part. It shows the block diagram of the proposed system with tables contain the experimental results drawn from the research. Finally, conclusion and possible direction of future work are given.

# CHAPTER ONE: BASIC CONCEPTS O F PATTERN AND FACE RECOGNITION

## 1.1. Overview

This chapter is dedicated to basic principals of pattern and face recognition. A brief summary about face recognition history together with an explanation about pattern and face recognition systems together with a block diagrams of those systems. The block diagram of face recognition system where explained in details. Moreover the problems that may arise during the face recognition process were listed.

## 1.2. Pattern Recognition

The need for improved information systems has become more conspicuous, since information is an essential element in decision making, and the world is generating increasing amounts of information in various forms with different degrees of complexity. One of the major problems in the design of modern information systems is automatic pattern recognition. Recognition is regarded as a basic attribute of human beings, as well as other living organisms. A pattern is the description of an object. A human being is a very sophisticated information system, partly because he possesses a superior pattern recognition capability. According to the nature of the patterns to be recognized, recognition acts can be divided into two major types [3].

- **Recognition of concrete items.** This may be referred to as sensory recognition, which includes visual and aural pattern recognition. This recognition process involves the identification and classification of spatial and temporal patterns. Examples of spatial patterns are characters, fingerprints, physical objects, and images. Temporal patterns include speech waveforms, time series, electrocardiograms and target signatures.
- **Recognition of abstract items.** On the other hand, an old argument, or a solution to a problem can be recognized. This process involves the recognition of abstract items and can be termed conceptual recognition. Recognition of concrete patterns by human beings may be considered as a psycho physiological problem which involves a relationship between a person and a physical stimulus. Human recognition is in reality a

question of estimating the relative odds that the input data can be associated with one of a set of known statistical populations which depend on our past experience and which form the clues and the a priori information for recognition. Thus, the problem of pattern recognition may be regarded as one of discriminating the input data between populations via the search for features or invariant attributes among members of a population.

### 1.2.1. Pattern Classes and Patterns

Pattern recognition can be defined as the categorization of input data into identifiable classes via the extraction of significant features or attributes of the data from a background of irrelevant detail.

A **pattern class** is a category determined by some given common attributes or features. The features of a pattern class are the characterizing attributes common to all patterns belonging to that class. Such features are often referred to as intraset features. The features which represent the differences between pattern classes may be referred to as the interset features.

A **pattern** is the description of any member of a category representing a pattern class. For convenience, patterns are usually represented by a vector such as:

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \ldots\ldots \\ \ldots\ldots \\ X_n \end{bmatrix} \tag{1.1}$$

where each element $X_j$, $j=1,\ldots..n$, represents a feature of that pattern. It is often useful to think of a pattern vector as a point in an n-dimensional Euclidean space.

patterns will present little difficulty. Automatic recognition may be reduced to a simple matching process or a table look-up scheme. However, in most pattern recognition problems which arise in practice, the determination of a complete set of discriminatory features is extremely difficult, if not impossible.

- The third problem in pattern recognition system design involves the determination of the optimum decision procedures, which are needed in the identification and classification process. After the observed data from patterns to be recognized have been expressed in the form of pattern points or measurement vectors in the pattern space, we want the machine to decide to which pattern class these data belong. Let the system be capable of recognizing M different pattern classes. Then the pattern space can be considered as consisting of M regions, each of which encloses the pattern points of a class. The recognition problem can now be viewed as that of generating the decision boundaries which separate the M pattern classes on the basis of the observed measurement vectors. These decision boundaries are generally determined by decision functions.

### 1.2.3. Training and Learning

The decision functions can be generated in a variety of ways. When complete priority knowledge about the patterns to be recognized is available, the decision function may be determined with precision on the basis of this information. When only qualitative knowledge about the patterns is available, reasonable guesses of the forms of the decision functions can be made. In this case the decision boundaries may be far from correct, and it is necessary to design the machine to achieve satisfactory performance through a sequence of adjustments.

The more general situation is that there exists little, if any, priority knowledge about the patterns to be recognized. Under these circumstances pattern recognizing machines are best designed using training or learning procedure. Arbitrary decision functions are initially assumed, and through a sequence of iterative training steps these decision functions are made to approach optimum or satisfactory forms.

It is important to keep in mind that learning or training takes place only during the design (or updating) phase of a pattern recognition system. Once acceptable results

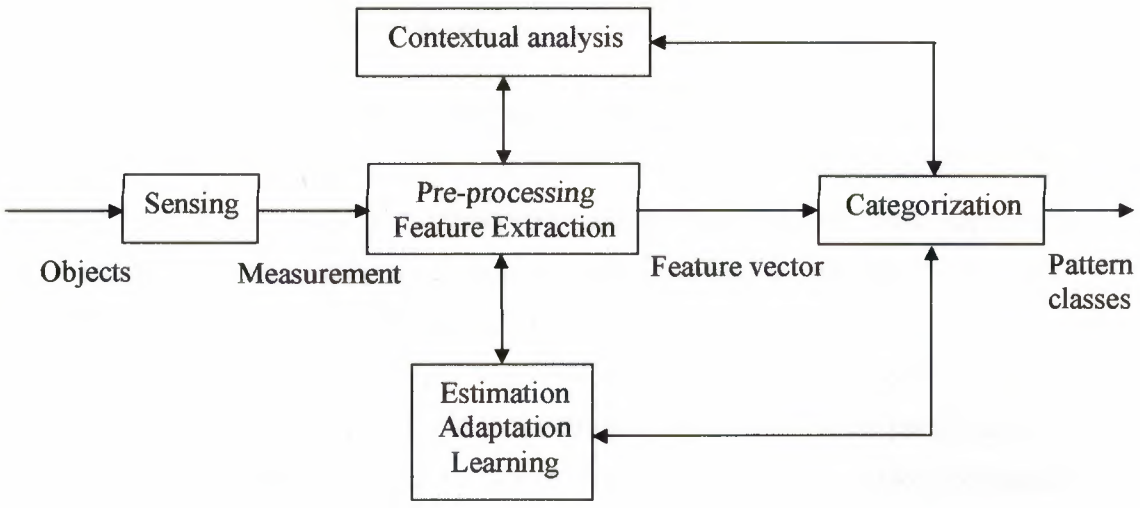


**Figure 1.2.** Functional block diagram of an adaptive pattern recognition system.

Correct recognition will depend on the amount of discriminating information contained in the measurements and the effective utilization of this information. In some applications, contextual information is indispensable in achieving accurate recognition. For instance, in the recognition of cursive handwritten characters and the classification of fingerprints, contextual information is extremely desirable. When we wish to design a pattern recognition system which is resistant to distortions, flexible under large pattern deviations, and capable of self-adjustment, we are confronted with the adaptation problem.

### 1.2. Face Recognition

Face recognition is a pattern recognition task performed specifically on faces. It can be described as classifying a face either "known" or "unknown", after comparing it with stored known individuals. It is also desirable to have a system that has the ability of learning to recognize unknown faces.

Computational models of face recognition must address several difficult problems. This difficulty arises from the fact that faces must be represented in a way that best utilizes the available face information to distinguish a particular face from all other faces. Faces pose a particularly difficult problem in this respect because all faces are similar to one another in that they contain the same set of features such as eyes, nose, mouth arranged in roughly the same manner.

### 1.2.1 Background and Related Work

Much of the work in computer recognition of faces has focused on detecting individual features such as the eyes, nose, mouth, and head outline, and defining a face model by the position, size, and relationships among these features. Such approaches have proven difficult to extend to multiple views and have often been quite fragile, requiring a good initial guess to guide them. Research in human strategies of face recognition, moreover, has shown that individual features and their immediate relationships comprise an insufficient representation to account for the performance of adult human face identification [4]. Nonetheless, this approach to face recognition remains the most popular one in the computer vision literature.

Bledsoe [5,6] was the first to attempt semi-automated face recognition with a hybrid human-computer system that classified faces on the basis of fiducially marks entered on photographs by hand. Parameters for the classification were normalized distances and ratios among points such as eye corners, mouth corners, nose tip, and chin point. Later work at Bell Labs developed a vector of up to 21 features, and recognized faces using standard pattern classification techniques.

Fischler and Elschlager [7], attempted to measure similar features automatically. They described a linear embedding algorithm that used local feature template matching and a global measure of fit to find and measure facial features. This template matching approach has been continued and improved by the recent work of Yuille and Cohen [8]. Their strategy is based on deformable templates, which are parameterized models of the face and its features in which the parameter values are determined by interactions with the face image.

Connectionist approaches to face identification seek to capture the configurationally nature of the task. Kohonen [9] and Kononen and Lehtio [10] describe an associative network with a simple learning algorithm that can recognize face images and recall a face image from an incomplete or noisy version input to the network. Fleming and Cottrell [11] extend these ideas using nonlinear units, training the system by backpropagation.

Others have approached automated face recognition by characterizing a face by a set of geometric parameters and performing pattern recognition based on the parameters. Kanade's [12] face identification system was the first system in which all steps of the recognition process were automated, using a top-down control strategy directed by a generic model of expected feature characteristics. His system calculated a set of facial parameters from a single face image and used a pattern classification technique to match the face from a known set, a purely statistical approach depending primarily on local histogram analysis and absolute gray-scale values.

Recent work by Burt [13] uses a smart sensing approach based on multiresolution template matching. This coarse to fine strategy uses a special purpose computer built to calculate multiresolution pyramid images quickly, and has been demonstrated identifying people in near real time.

S. H. Lin, S. Y. Kung and L. J. Lin [14] proposed a face recognition system based on probabilistic decision-based neural networks (PDBNN). The PDBNN face recognition system consists of three modules: First, a *face detector* finds the location of a human face in an image. Then an *eye localizer* determines the positions of both eyes in order to generate meaningful feature vectors. The facial region proposed contains eyebrows, eyes, and nose, but excluding mouth. (Eye-glasses will be allowed.) Lastly, the third module is a *face recognizer*. The PDBNN can be effectively applied to all the three modules. It adopts a hierarchical network structures with nonlinear basis functions and a competitive credit-assignment scheme.

S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back [15] presented a hybrid neural-network solution which compares favorably with other methods. The system combines local image sampling, a self-organizing map (SOM) neural network, and a convolutional neural network. The SOM provides a quantization of the image samples into a topological space where inputs that are nearby in the original space are also nearby in the output space, thereby providing dimensionality reduction and invariance to minor changes in the image sample, and the convolutional neural network provides for partial invariance to translation, rotation, scale, and deformation. The convolutional network extracts successively larger features in a hierarchical set of layers.

M. Zhang, and J. Fulcher [16] introduced their Artificial Neural Network group-based adaptive tolerance (GAT) tree model for translation-invariant face recognition. GAT trees use a two-stage divide-and-conquer tree type approach. The first stage determines general properties of the input, such as whether the facial image contains glasses or a beard. The second stage identifies the individual. Face perception classification, detection of front faces with glasses and/or beards.

Bouttour [17] developed a human face recognition system using MLP's. Flocchini [18] described an image processing and neural network system which is capable of recognizing human faces from different prospective. Their system is likewise limited in the number of faces it can recognize, but it does so with an accuracy of between 80% and 100%. A new pattern classification method called Neural Tree Networks (NTN) was proposed by sanker [19]. The NTN consists of neural networks connected in a tree architecture. The use of a neural network at each tree node, as in NTN, results in better classification performance. The results show that the NTN compares favorably to both neural networks and decision trees.

Kerin and Stonham [20] presented a novel approach to face recognition using digital neural networks with self-organizing capabilities. An automatic training procedure based on self-organization is defined. This partitions the data into disjoint classes containing recognizable attributes and can be used to replace a human operator in selecting valid data for training purposes.

### 1.2.2 Outline of a Typical Face Recognition System

In Figure 1.3, the outline of a typical face recognition system is given. This outline heavily carries the characteristics of a typical pattern recognition system that was presented in Figure 1.2.

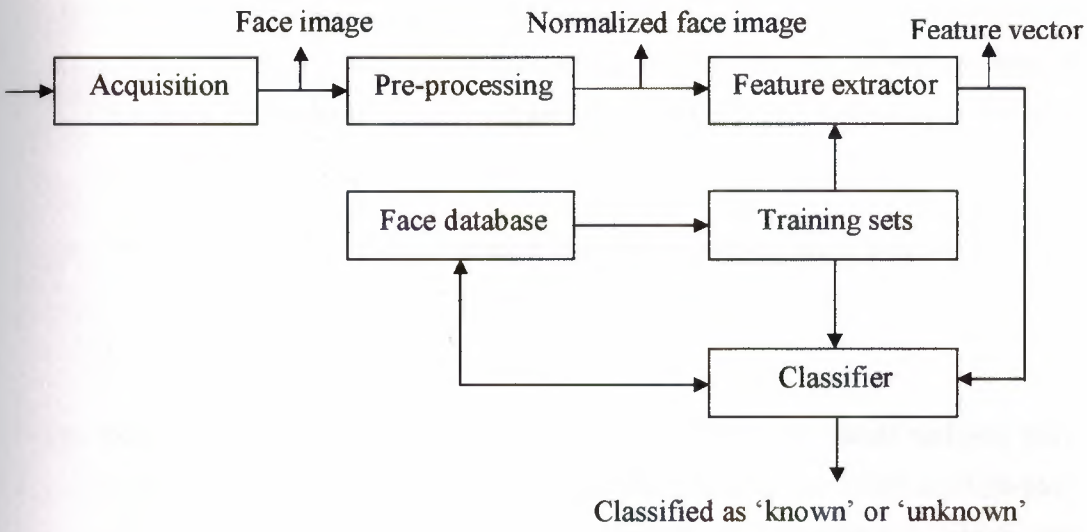


**Figure 2.3.** Outline of a typical face recognition system.

There are six main functional blocks, whose responsibilities are given below:

- **The acquisition module.** This is the entry point of the face recognition process. It is the module where the face image under consideration is presented to the system. In other words, the user is asked to present a face image to the face recognition system in this module. An acquisition module can request a face image from several different environments: The face image can be an image file that is located on a magnetic disk, it can be captured by a frame grabber or it can be scanned from paper with the help of a scanner.

- **The pre-processing module.** In this module, by means of early vision techniques, face images are normalized and, if desired, they are enhanced to improve the recognition performance of the system. Some or all of the following pre-processing steps may be implemented in a face recognition system:

    - **Image size normalization**. It is usually done to change the acquired image size to a default image size such as 128 x 128, on which the face recognition system operates. This is mostly encountered in systems where face images are treated as a whole like the one proposed in this thesis.

- **Histogram equalization.** It is usually done on too dark or too bright images in order to enhance image quality and to improve face recognition performance. It modifies the dynamic range (contrast range) of the image and as a result, some important facial features become more apparent.

- **Median filtering.** For noisy images especially obtained from a camera or from a frame grabber, median filtering can clean the image without loosing information.

- **High-pass filtering.** Feature extractors that are based on facial outlines, may benefit the results that are obtained from an edge detection scheme. High-pass filtering emphasizes the details of an image such as contours which can dramatically improve edge detection performance.

- **Background removal.** In order to deal primarily with facial information itself, face background can be removed. This is especially important for face recognition systems where entire information contained in the image is used. It is obvious that, for background removal, the preprocessing module should be capable of determining the face outline.

- **Translational and rotational normalizations.** In some cases, it is possible to work on a face image in which the head is somehow shifted or rotated. The head plays the key role in the determination of facial features. Especially for face recognition systems that are based on the frontal views of faces, it may be desirable that the preprocessing module determines and if possible, normalizes the shifts and rotations in the head position.

- **Illumination normalization.** Face images taken under different illuminations can degrade recognition performance especially for face recognition systems based on the principal component analysis in which entire face information is used for recognition. A picture can be equivalently viewed as an array of reflectivity's r(x). Thus, under a uniform illumination I, the corresponding picture is given by

$$\Phi(x) = Ir(r) \tag{1.2}$$

The normalization comes in imposing a fixed level of illumination $I_0$ at a reference point $X_0$ on a picture. The normalized picture is given by

$$\Phi(X) = \frac{I_0 \Phi(X)}{I(X_0)} \tag{2.3}$$

In actual practice, the average of two reference points, such as one under each eye, each consisting of $2 \times 2$ array of pixels can be used.

- **The feature extraction module.** After performing some pre-processing (if necessary), the normalized face image is presented to the feature extraction module in order to find the key features that are going to be used for classification. In other words, this module is responsible for composing a feature vector that is well enough to represent the face image.

- **The classification module.** In this module, with the help of a pattern classifier, extracted features of the face image is compared with the ones stored in a face library (or face database). After doing this comparison, face image is classified as either known or unknown.

- **Training set.** Training sets are used during the "learning phase" of the face recognition process. The feature extraction and the classification modules adjust their parameters in order to achieve optimum recognition performance by making use of training sets.

- **Face library or face database.** After being classified as "unknown", face images can be added to a library (or to a database) with their feature vectors for later comparisons. The classification module makes direct use of the face library.

### 1.2.3. Problems that May Occur During Face Recognition

Due to the dynamic nature of face images, a face recognition system encounters various problems during the recognition process. It is possible to classify a face recognition system as either "robust" or "weak" based on its recognition performances under these circumstances. The objectives of a robust face recognition system are given below:

- **Scale invariance.** The same face can be presented to the system at different scales as shown in Figure 1.4-b. This may happen due to the focal distance between the face and the camera. As this distance gets closer, the face image gets bigger.

- **Shift invariance.** The same face can be presented to the system at different perspectives and orientations as shown in Figure 1.4-c. For instance, face images of the same person could be taken from frontal and profile views. Besides, head orientation may change due to translations and rotations.

- **Illumination invariance.** Face images of the same person can be taken under different illumination conditions such as, the position and the strength of the light source can be modified like the ones shown in Figure 1.4-d.

- **Emotional expression and detail invariance.** Face images of the same person can differ in expressions when smiling or laughing. Also, like the ones shown in Figure 1.4-e, some details such as dark glasses, beards or moustaches can be present.

- **Noise invariance.** A robust face recognition system should be insensitive to noise generated by frame grabbers or cameras. Also, it should function under partially occluded images.

A robust face recognition system should be capable of classifying a face image as "known" under even above conditions, if it has already been stored in the face database.

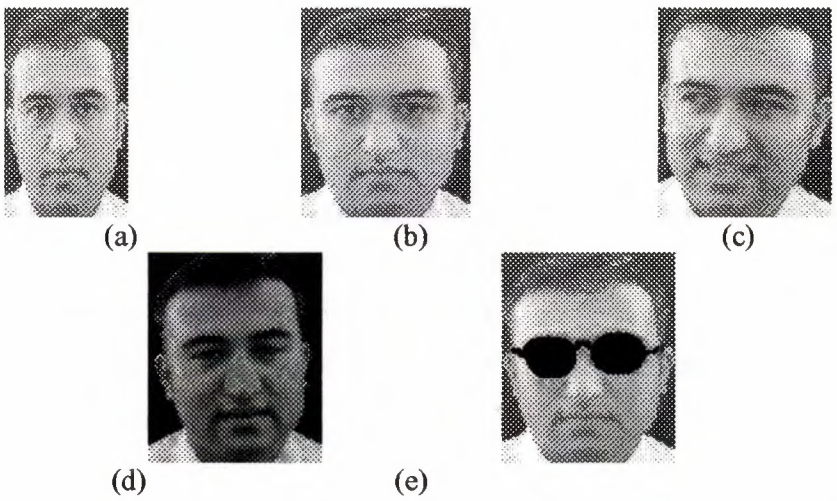


**Figure 1.4.** (a) Original face image. (b) Scale variance. (c) Orientation variance. (d)Illumination variance. (e) Presence of details.

## 1.3. Summary

In this chapter a brief explanation about pattern and face recognition systems together with a block diagram of those systems. Fundamental problems on pattern recognition design were discussed. The block diagram of face recognition system where explained in details together with a list to the problems that may arise during the face recognition process.

# CHAPTER TWO: NEURAL NETWORKS

## 2.1. Overview

Neural networks have seen an explosion of interest over the last few years, and are being successfully applied across an extraordinary range of problem domains, in areas as diverse as finance, medicine, engineering, geology and physics. Indeed, anywhere that there are problems of prediction, classification or control, neural networks are being introduced.

This chapter presents an overview of Neural Networks, well known Neural Networks architecture, their operational principles, and their learning algorithms such as Back-propagation which is the most commonly used. It introduces the terminology and the equations that describe the Back-propagation network training and operation, and lists its advantages and disadvantages.

## 2.2. Architecture of Neural Networks

Often, it is convenient to visualize neurons as arranged in layers. Typically, neurons in the same layer behave in the same manner. Key factors in determining the behavior of a neuron are its activation function and the pattern of weighted connections over which it sends and receives signals. Within each layer, neurons usually have the same activation function and the same pattern of connections to other neurons. To be more specific, in many neural networks, the neurons within a layer are either fully interconnected or not interconnected at all. If any neuron in a layer (for instance, the layer of hidden units) is connected to a neuron in another layer (say, the output layer), then each hidden unit is connected to every output neuron.

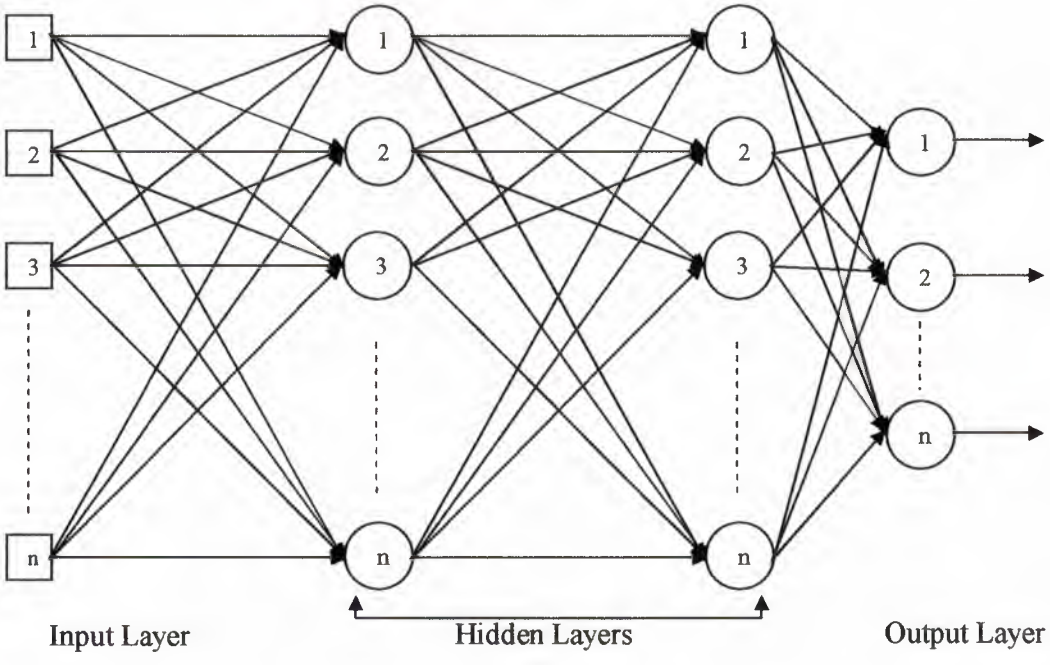


**Figure 2.1.** Architectural graph of a feed-forward neural network.

The arrangement of neurons into layers and the connection patterns within and between layers is called the *net architecture.* Many neural nets have an input layer in which the activation of each unit is equal to an external input signal. The net illustrated in Figure 2.1 consists of input units, output units, and two hidden units (the units that are neither an input unit nor an output unit).

Neural nets are often classified as single layer or multilayer. In determining the number of layers, the input units are not counted as a layer, because they perform no computation. Equivalently, the number of layers in the net can be defined to be the number of layers of weighted interconnect links between the slabs of neurons. This view is motivated by the fact that the weights in a net contain extremely important information. The net shown in Figure 2.1 has three layers of weights which is an example *of feed forward* net in which the signals flow from the input units to the output units, in a forward direction.

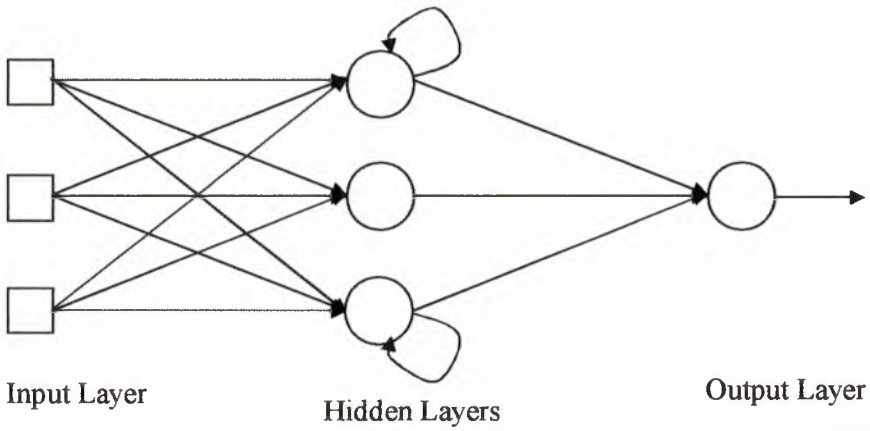


**Figure 2.2.** Recurrent neural networks

The fully interconnected competitive net in figure 2.2 is an example of a *recurrent* net, in which there are closed-loop signal paths from a unit back to it.

### 2.2.1. Single-Layer Net

A single-layer net has one layer of connection weights. Often, the units can be distinguished as input units, which receive signals from the outside world, and output units, from which the response of the net can be read. In the typical single layer net shown in Figure 2.3, the input units are fully connected to output units but are not connected to other input units, and the output units are not connected to other output units. By contrast, the Hopfield net architecture, shown in Figure 2.3, is an example of a single-layer net in which all units function as both input and output units. For pattern classification, each output unit corresponds to a particular category to which an input vector may or may not belong.

For a single layer net, the weights for one output unit do not influence the weights for other output units. For pattern association, the same architecture can be used, but now the overall pattern of output signals gives the response pattern associated with the input signal, that caused it to be produced. These two examples illustrate the fact that the same type of net can be used for different problems, depending on the interpretation of the response of the net.

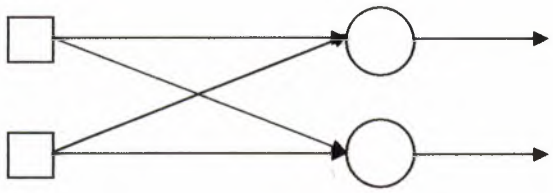

**Figure 2.3.** A single layered

### 2.2.2. Multilayer Net

A multilayer net is a net with one or more layers or levels of nodes between the input units and the output units. Typically, there is a layer of weights between two adjacent levels of units, (input, hidden, or output). As shown in Figure 2.2, multilayer nets can solve more complicated problems than, can single-layer nets, but training may be more difficult. However, in some cases, training may be more successful, because it is possible to solve a problem that a single-layer net cannot, be trained to perform correctly at all.

## 2.3. Learning Algorithms

There is no unique learning algorithm for the design of the neural networks. Rather, a "kit of tools" are represented by a diverse variety of learning algorithms, each of which efforts advantages of its own. Basically, learning algorithms differ from each other in the way in which the adjustment $\Delta w_{kj}$ to the synaptic weight $w_{kj}$ is formulated. Another factor to be considered is the manner in which the neural network relates to its environment.

### 2.3.1. Supervised Learning

An essential ingredient of supervised or active learning is the availability of an external teacher (supervisor) as indicated in Figure 2.4.

Conceptually, it is thought that the supervisor or the teacher as having knowledge of the environment that is represented by a set of input-output examples. The environment is, however, unknown to the neural network. Suppose that the teacher and the Neural Networks are both exposed to a training vector drown from the environment.

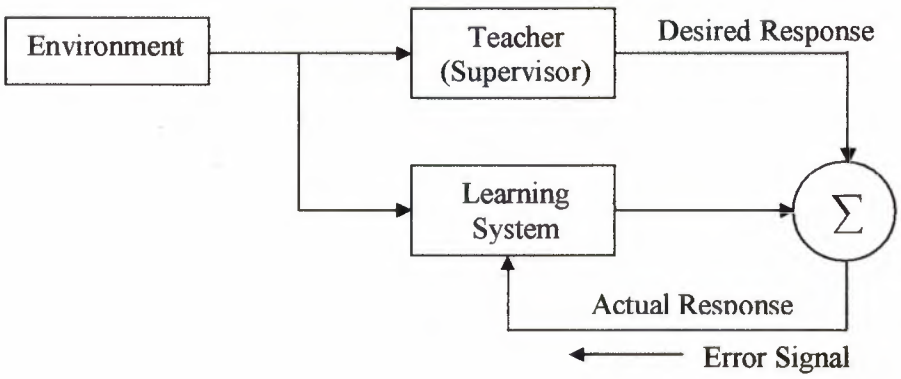


**Figure 2.4.** Block diagram of supervised learning system

By virtue of built in knowledge, the teacher is able to provide the neural network with a desired or target response for that training vector. Indeed the desired response represents the optimum action to be performed by the Neural Network. The network parameters are adjusted under the combined influence of the training vector and the error signal; the error signal is defined as the difference between the actual response of the network and the desired response. This adjustment is carried out iteratively in a step-by-step fashion with the aim of eventually making the neural network emulate the teacher, the emulation is presumed to be optimal in some statistical sense. In other words, knowledge of the environment available to the teacher is transferred to the neural network as filly as possible. When this condition is reached then the teacher may be dispensed and the neural network deals with the environment completely by it self (i.e. in an unsupervised fashion).

The supervised learning is a closed-loop feedback system. The mean-squared error defined as a function of the free parameters may be used as a performance measure of the system. This function may be visualized as a multidimensional error-performance surface. The true error surface is averaged over all possible input-output examples. Any given operation of the system under the teacher's supervision is represented as a point on the error surface. For the system to improve performance over time and therefore learn from the teacher, the operating point has to move down successively toward a minimum point of the error surface. A supervised learning system is able to do this by virtue of some useful information it has about the gradient of the error surface corresponding to the current behavior of the system. The gradient of an error surface at

any point is a vector that points in the direction of the steepest descent. In fact in the case of supervised learning from example; the system uses an instantaneous estimate of the gradient vector, with the example indices presumed to be those of time.

The use of such an estimate results in a motion of operating point on the error surface that is typically in the form of a "random walk." Nevertheless, given an algorithm designed to minimize the cost function of interest, and given an adequate set of input-output examples and enough time permitted to do the training, a supervised learning system is usually able to perform such tasks as pattern classification and function approximation satisfactorily.

An example of supervised learning algorithms is the back-propagation learning algorithm. Supervised learning algorithm can be performed in an off-line or on-line manner. In the off-line case, a separate computational facility is used to design the supervised learning system. Once the desired performance is accomplished, the design is "frozen", which means that the Neural Network operates in a static manner.

On the other hand in on-line learning, the learning procedure is implemented solely within the system it self, not requiring a separate computational facility. In other words, learning is accomplished in real time, with the result that the neural network is dynamic.

An advantage of supervised learning, regardless of whether it is performed off-line or on-line, is the fact that without the teacher, a neural network can not learn new strategies for particular situation that are not covered by the set of examples used to train the network. This limitation can be over come, by the use of reinforcement learning.

### 2.3.2. Unsupervised Learning

Once the network has become tuned to the statistical regularities of the input data, it develops the ability to form internal representations for encoding features of the input by creating new classes automatically. To perform unsupervised leaning, a competitive leaning rule may he used, for example, a N.N that consists of two layers may be used, namely an input layer and a competitive layer. The input layer receives the available

data. The competitive layer consists of neurons that compete with each other for the opportunity to respond to features contained in the input data. In its simplest form, the network operates in accordance with a "winner-takes-all" strategy. In such a strategy the neurons with greatest total input wins the competition and turns on; all the other neurons then switch off.

In an unsupervised learning or self-organized learning there is no external teacher or critic to oversee the learning process, as indicated in the Figure 2.5. In other words, there are no specific examples of the function to be learned by the network. Rather, provision is made for a task-independent measure of the quality of representation that the network is required to learn, and the free parameters of the network are optimized with respect to that measure.

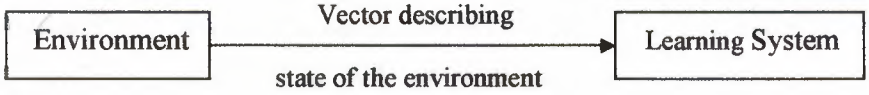


**Figure 2.5.** Block diagram of unsupervised learning.

### 2.3.3 Supervised Versus Unsupervised

Among the algorithm used to perform supervised learning, the back-propagation algorithm has emerged as the most widely used and successful algorithm for the design to multilayer feed-forward networks. There are two distinct phases to the operation of back-propagation learning: the forward phase and the backward phase. In the forward phase the input signals propagate through the network layer-by-layer, eventually preceding some response of the output of the network. The actual response so produced is compared with a desired (target) response, generating error signals that are then propagated in a backward direction through the network. In this backward phase of operation, the free parameters of the network are adjusted so as to minimize the sum of squared errors. Back-propagation learning has been applied successfully to solve some difficult problems such as speech recognition from text, handwriting digit recognition, and adaptive control. Unfortunately, back-propagation and other supervised learning algorithms may be limited by their poor scaling behavior. To understand this limitation, an example of multilayer feed-forward network consisting of $L$ computation layer is

considered. The effect of a synaptic weight in the first layer on the output of the network depends on its interaction with approximately $F_i$ to the power $L$ other synaptic weights, where $F_i$ is the fan-in, defined as the average number of incoming links of neurons in the network. Hence, as the size of the network increases, the network becomes more computationally intensive, and so the time required to train the network grows exponentially and the learning process becomes unacceptable slow.

One possible solution to the scaling problem described here is to use an unsupervised learning procedure. In particular, if a self-organizing process is able to be apply in a sequential manner, one layer of a time; it is feasible to train deep networks in time that is linear in the number of layers. Moreover, with the ability of the self-organizing network to form internal representations that model the underlying structure of the input data in a more explicit or simple form, it is hoped that the transformed version of the sensory input would be easier to interpret, so that correct responses could be associated with the network's internal representations of the environment more quickly. In other words, the use of supervised learning procedures may provide a more acceptable solution than unsupervised learning alone.

### 2.4. Back-Propagation Algorithm

The most common supervised learning algorithm is the back-propagation (BP) algorithm, which is also called generalized *delta rule.* It is a gradient descent algorithm that is normally used to train the Multilayer Perceptron (MLP) network.

Basically the error back-propagation process consists of two passes through the different layers of the network: a forward pass and a backward pass. In the forward *pass* an activity pattern (input vector) is applied to the sensory nodes of the network, and its effect propagates layer by layer. Finally a set of outputs is produced as the actual response of the network. During the forward pass the synaptic weights of the network are all fixed. During the backward pass on the other hand the synaptic weights are all adjusted in accordance with the error-correction rule. Specifically, the actual response of the network is subtracted from a desired (target) response to produce an error signal. This error signal is then propagated backward through the network against the direction of synaptic connections, hence the name "error back-propagation". The synaptic

weights are adjusted so as to make the actual response of the network move closer to the desired response.

The elements that combine to form the topology for a back propagation neural network model will be presented next. The equations that describe the network training and operation can be divided into two categories. First, the *feed-forward calculations.* These are used in both training mode and in the operation of the trained neural network. Second, the *error back propagation calculations.* These are applied only during training. But before we present the two categories of calculations, another important element must be described. This is the activation function that the algorithm will be based upon [21].

### 2.4.1. The Activation Function

An artificial neuron, as shown in Figure 2.6, is the fundamental building block in a back propagation network. The input to the neuron is obtained as the weighted sum given by

$$net = \sum_{i=1}^{n} O_i W_i \tag{2.1}$$

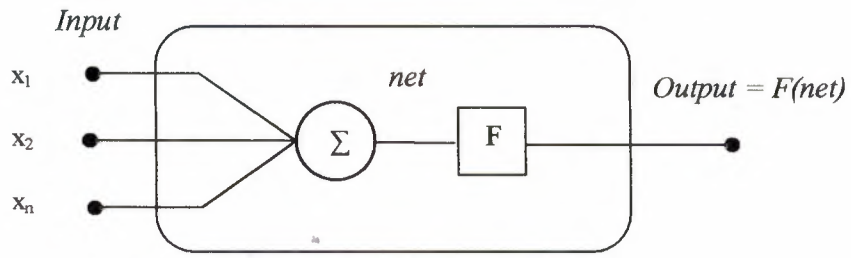


**Figure 2.6.** Artificial neuron.

In Figure 2.6, $F$ is the activation function, which has a sigmoid form. The simplicity of the derivative of the sigmoid function justifies it s popularity and use as an activation function in training algorithms. With a sigmoid activation function, the output of the neuron is given by

$$out = F(net) \tag{2.2}$$

$$F(net) = \frac{1}{(1 + \exp(-net))} \tag{2.3}$$

### 2.4.2 Feed Forward Calculations

Figure 2.7 shows the most common configuration of a back propagation neural network. This is the simple three layer back propagation model. Each neuron is represented by a circle and each interconnection, with its associated weight, by an arrow. The neurons labeled *b* are bias neurons.

Normalization of the input data prior to training is necessary. The values of the input data into the input layer must be in the range (0 - 1). The stages of the feed forward calculations can be described according to the layers. The suffixes *i, h* and *j* are used for *input, hidden* and *output* respectively.

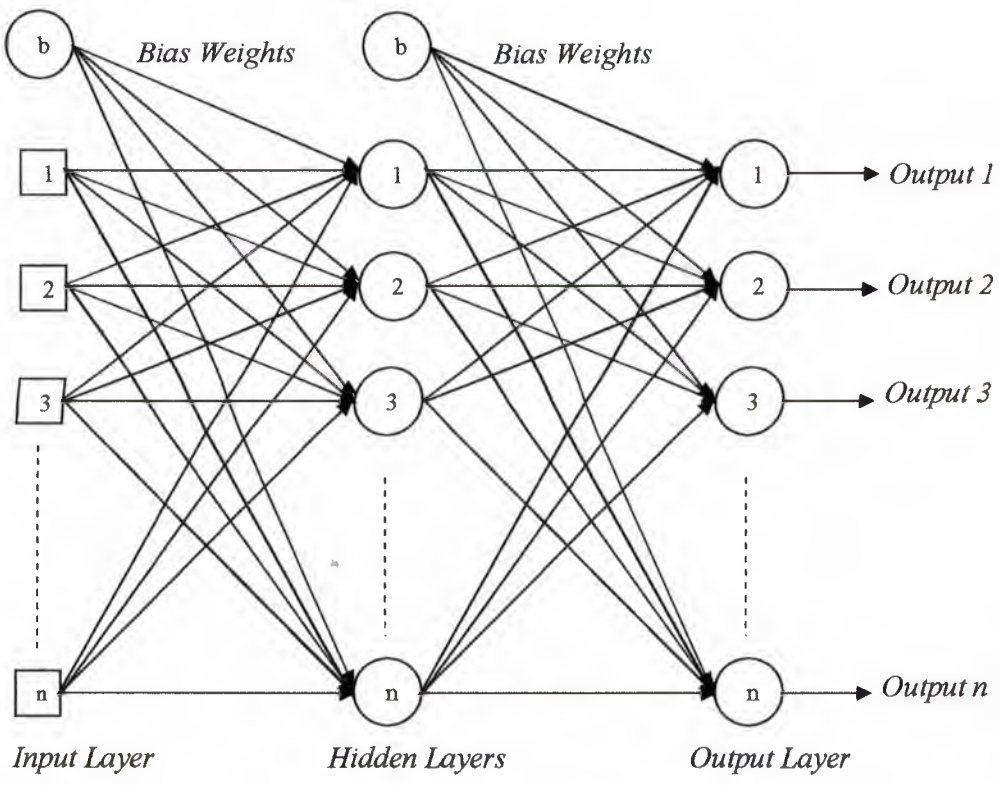


**Figure 2.7**. Back propagation network structure.

### 2.4.2.1 Input Layer( i )

Figure 2.8 shows a neuron in the input layer. The output of each input layer neuron is exactly equal to the normalized input.

$$Input - Layer\ Output = O_i = I_i \tag{2.4}$$

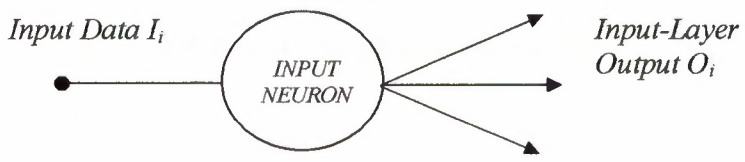


**Figure 2.8**. An input layer neuron.

### 2.4.2.2 Hidden Layer (h)

Figure 2.9 describes a neuron in the hidden layer. The signal presented to a neuron in the hidden layer is equal to the sum of all the outputs of the input layer neurons multiplied by their associated connection weights, as follows:

$$Hidden - Layer\ Input_h = I_h = \sum_i W_{hi} O_i \tag{2.5}$$

Each output of a hidden neuron is calculated using the sigmoid function. This is described by

$$Hidden - Layer\ Output_h = O_h = \frac{1}{1 + \exp(-I_h)} \tag{2.6}$$

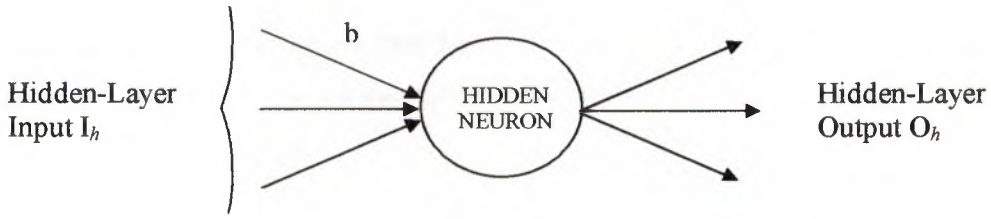


**Figure 2.9.** A Hidden layer neuron.

### 2.4.2.3 Output Layer (j)

Figure 2.10 describes a neuron in the output layer. The signal presented to a neuron in the output layer is equal to the sum of all the outputs of the hidden layer neurons multiplied by their associated connection weights plus the bias weights at each neuron, and is given by

$$Output-Layer\ Input_j = I_j = \sum_h W_{jh} O_h \tag{2.7}$$

Each output of an output neuron is calculated using the sigmoid function in a similar manner as in the hidden layer. This is described by

$$Output-Layer\ Output_j = O_j = \frac{1}{1+\exp(-I_j)} \tag{2.8}$$

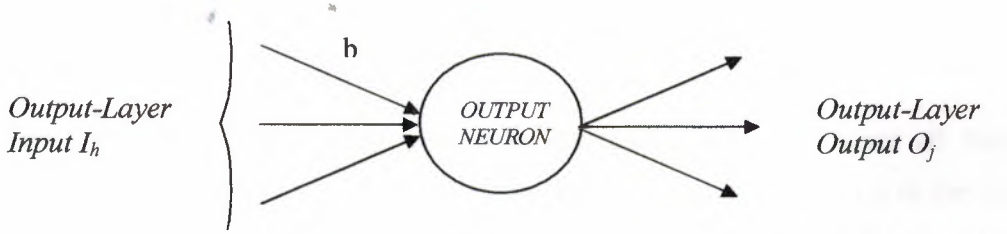


**Figure 2.10.** An output layer neuron.

The set of calculations that has been described so far in the feed forward calculations can be carried out during the training phase as well as during the testing / running phase.

### 2.4.3 Error Back Propagation Calculations

The error back propagation calculations are applied only during the training of the neural network. Vital elements in these calculations are described next. These include the error signal, some essential parameters and weight adjustment.

#### 2.4.3.1 Signal Error

During the network training, the feed forward output state calculation is combined with backward error propagation and weight adjustment calculations that represent the network's learning. Central to the concept of training a neural network is the definition of network *error*. Rumelhart and McClelland define an error term that depends on the difference between the output value an output neuron is supposed to have, called the target value $T_j$, and the value it actually has as a result of the feed forward calculations, $O_j$. The error term represents a measure of how well a network is training on a particular training set.

Equation (2.9) presents the definitions for the error. The subscript $p$ denotes the value for a given pattern:

$$E_p = \sum_{j=1}^{n_j} (T_{pj} - O_{pj})^2 \qquad (2.9)$$

The aim of the training process is to minimize this error over all training patterns. From equation (2.8), it can be seen that the output of a neuron in the output layer is a function of its input, or $O_j = f(I_j)$. The first derivative of this function, $f'(I_j)$ is an important element in error back propagation. For output layer neurons, a quantity called the *error signal* is represented by $\Delta j$ which is defined in the following equation.

$$\Delta_j = f'(I_j)(T_j - O_j) = (T_j - O_j)O_j(1 - O_j) \tag{2.10}$$

where $\Delta_j$ is error value. It is propagated back and appropriate weight adjustments are performed. This is done by accumulating the Δ's for each neuron for the entire training set, add them, and propagate back the error based on the grand total Δ. This is called *batch (epoch) training.*

#### 2.4.3.2 Essential Parameters

There are two essential parameters that do affect the learning capability of the neural network. First, the *learning coefficient* η which defines the learning 'power' of a neural network. Second, *the momentum factor* α which defines the speed at which the neural network learns. This can be adjusted to a certain value in order to prevent the neural network from getting caught in what is called *local minima.* Both rates can have a value between 0 and 1.

#### 2.4.3.3 Weight Adjustment

Each weight has to be set to an initial value. Random initialization is usually performed. Weight adjustment is performed in stages. Starting at the end of the feed forward phase, and going backward to the inputs of the hidden layer.

#### 2.4.3.3.1 Output-Layer Weights Update

The weights that feed the output layer ($W_{jh}$) are updated using

$$W_{jh}(new) = W_{jh}(old) + \eta\Delta_j O_h \tag{2.11}$$

This also includes the bias weights at the output layer neurons. However, in order to avoid the risk of the neural network getting caught in a local minima, the momentum term can be added as in

$$W_{jh}(new) = W_{jh}(old) + \eta\Delta_j O_h + \alpha[\delta W_{jh}(old)] \tag{2.12}$$

where $\delta W_{jh}$ *(old)* stands for the previous weight change.

#### 2.4.3.3.2 Hidden-Layer Weights Update

The error term for an output layer is defined in equation (2.10). For the hidden layer, it is not as simple to figure out a definition for the error term. However, a definition by Rumelhart and McClelland describes the error term for a hidden neuron as in

$$\Delta_h = f'(I_h)\sum_{j=0}^{n_j} W_{jh}\Delta_j \tag{2.13}$$

$$\Delta_h = O_h(1-O_h)\sum_{j=0}^{n_j} W_{jh}\Delta_j \tag{2.14}$$

The weight adjustments for the connections feeding the hidden layer from the input layer are now calculated with the aid of equation (2.12) in a similar manner to those feeding the output layer. These adjustments are calculated using

$$W_{hi}(new) = W_{hi}(old) + \eta\Delta_h O_i + \alpha[\delta W_{hi}(old)] \tag{2.15}$$

The bias weights at the hidden layer neurons are updated, similarly, using equation (2.15).

### 2.4.4. Levenberg-Marquardt back-propagation

A simple steepest descent minimization algorithm based on first-order minimization is slow (requires many repetitions of the training epochs to achieve a satisfactory minimization of the error) due to computation of error serially on the layer by layer basis. To increase speed of training and to avoid oscillation around a local minimum the weight update is performed using combination techniques such as adaptive learning rate $\eta$ and momentum term $\alpha$ as in equation (2.15)

The momentum method tries to get some curvature information about the error surface by averaging the gradient locally.

A second group algorithm for minimization of error $E_p$ is based on optimization techniques that basically use second order derivative of performance index or cost function J(w) [22,23]. A Taylor series expansion of J(w):

$$J(w_{n+1}) = J(w) + \Delta w\, \nabla J(w) + \frac{1}{2}\Delta w \nabla^2 J(w) + \ldots\ldots \tag{2.16}$$

where gradient of performance index is

$$g = \frac{\partial E}{\partial w} = \nabla J(w) \tag{2.17}$$

and Hessian matrix

$$H = \frac{\partial^2 E}{\partial^2 w} = \nabla^2 J(w) \tag{2.18}$$

The Levenberg-Marquardt algorithm was designed to approach second-order training speed without having to compute the Hessian matrix. When the performance function has the form of a sum of squares (as is typical in training feed forward networks), then the Hessian matrix can be approximated as

$$H = J^T J \tag{2.19}$$

and the gradient can be computed as

$$g = J^T E \tag{2.20}$$

where $J$ is the Jacobian matrix that contains first derivatives of the network errors with respect to the weights and biases, and **e** is a vector of network errors. The Jacobian matrix can be computed through a standard Backpropagation technique that is much less complex than computing the Hessian matrix.

The Levenberg-Marquardt algorithm uses this approximation to the Hessian matrix in the following Newton-like update:

$$x_{k+1} = x_k - [J^T J + \mu I]^{-1} J^T E \tag{2.21}$$

where μ is a nonnegative scalar, that controls the magnitude and direction of I; I is an identity matrix. When the scalar μ is zero, this is just Newton's method, using the approximate Hessian matrix. When μ is large, this becomes gradient descent with a small step size. Newton's method is faster and more accurate near an error minimum, so the aim is to shift towards Newton's method as quickly as possible. Thus, μ is decreased after each successful step (reduction in performance function) and is increased only when a tentative step would increase the performance function. In this way, the performance function will always be reduced at each iteration of the algorithm.

### 2.4.5. Advantages of back-propagation

The back-propagation algorithm is an example of a connectionist paradigm that relies on local computations to discover the information-processing capabilities of neural networks. This form of computational restriction is referred to as the locality constraint; in this sense the computation performed by a neuron is influenced solely by those neurons that are in physical contact with it. The use of local computations in the design of artificial neural networks is usually advocated for three principal reasons:

1. Artificial neural networks that perform local computations are often held up as metaphors for biological neural networks

2. The use of local computations permits a graceful degradation in performance due to hardware errors, and therefore a fault-tolerant network design.

3. Local computations favor the use of parallel architectures as an efficient method for the implementation of artificial neural networks.

Taking a look at these points it is obvious that, the third point is perfectly justified in the case of back-propagation learning, where it has successfully implemented on parallel computers by many investigators, and VLSI architecture has been for the

hardware realization of multilayer perceptrons. The second point is justified so as certain precautions are taken in the application of back-propagation algorithm, such as injecting small numbers of "transient faults" at each step. For the first point, relating to the biological plausibility of back-propagation learning, it has indeed questioned as follow:

1. The reciprocal synaptic connections between neurons of a multilayer perceptron may assume weights that are excitatory or inhibitory.

2. In a multilayer perceptron, hormonal and other types of global communication are ignored.

3. In back propagation learning, a synaptic weight is modified by a pre-synaptic activity and an error signal independent of postsynaptic activity.

4. The implementation of back-propagation learning requires the rapid transmission of information backward along the axon.

5. Back-propagation learning implies the existence of a "teacher." which in the context of the brain would presumably be another set of neurons with novel properties.

### 2.4.5. Disadvantages of back-propagation

Consequently, the rate of convergence in back-propagation algorithm tends to be relatively slow, which in turn makes it computationally expensive. The local convergence rate of the back-propagation algorithm is linear, which justify the rank-deficient in the Jacobean and Hessian matrices; these are the consequences of the intrinsically ill-conditioned nature of neural network training problems. Saarinen interprets the linear local convergence rates of the back-propagation learning on two ways:

1. It is a vindication of back-propagation in the sense that higher order methods may not converge much faster while requiring more computational effort.

2. Large-scale neural network training problems are so inherently difficult to

perform that no supervised learning strategy is feasible, and other approaches such as the use of processing may be necessary.

Another peculiarity of the error surface that impacts the performance of back-propagation algorithm is the presence of local minima in addition of global minima. Since the back-propagation is a hill-climbing technique, it runs the risk of being trapped in a local minimum. It is undesirable to have the learning process terminate at a local minimum, especially if it is located far above a global minimum. But stacking in a local minimum is rarely a practical problem for back-propagation learning.

Another problem of learning by back-propagation that has to be overcome is scaling, which addresses the issue of how well the network behaves as the computational task increase in size and complexity. One way of alleviating the scaling problem is to develop insight into the problem at hand and use it to put ingenuity into the architectural design. Another way to deal with scaling problem is to reformulate the back-propagation learning process with modularity built into the network architecture.

## 2.5. Applications of Neural Networks in Commercial Products

### 2.5.1 Business Applications

- *Marketing*
    - Evaluates direct marketing decisions
    - Forecasts demand of airline flights
    - Predicting sales of soft drinks
    - Optimize marketing Strategy
    - Direct mail marketing
    - Determine which customers should receive catalogs
    - Predictive modeling system for direct marketers
    - Web advertising placement server
    - Finding the connection between the sales of diapers and beer
- *Real Estate*
    - Real estate appraisal

- Property valuation system

### 2.5.2. Document and Form Processing Applications

- *Machine Printed Character Recognition*
  - Optical character recognition
  - Check reader
- *Hand Printed Character Recognition*
  - Handwritten character recognition for faxes
  - Forms processor
  - Handwritten character recognizer
  - Input to pen-based computers
  - Input to Windows based computers
- *Cursive Handwriting Character Recognition*
  - Input for personal data assistant
  - Cursive handwriting system for PCs
- *Graphic Recognition*
  - Translation of chemistry drawings into connection tables
  - Object recognition
  - Face recognition

### 2.5.3. Financial Applications

- *Market Trading*
  - Stock forecasting
  - Trends analysis in international markets
  - Stock price forecasting
  - Bond portfolio management
  - Manage cattle futures trading
  - Corporate bond rating
  - Pension fund management

- Mutual fund management
- Portfolio management
- Technical analysis and prediction of stocks and other financial data
- Monetary market trading
- Common stock valuation

- *Fraud Detection*

  - Check approval
  - Credit card fraud detection
  - Identify deviations in spending habits (Master Card)
  - Signature verification from checks
  - Identify deviations in spending habits (Visa)

- *Credit Rating*

  - Credit scoring
  - Forecasting credit worthiness
  - Mortgage underwriting and risk-management

### 2.5.4. Energy Industry Applications

- *Electrical Load Forecasting*

  - Electrical load forecasting
  - Energy demand forecasting
  - Load forecasting (Power)
  - Short- and long-term load forecasting
  - Load estimation

- *Hydroelectric Dam Operation*

  - Dam-displacement prediction

- *Natural Gas*

  - Predicting gas index prices

### 2.5.5. Manufacturing Applications

- *Process Controllers*
    - Controller for continuous casting
    - Process control systems
    - Controller for continuous casting
    - Process control of rolling mills
    - Refinery process control system
- *Quality Control Systems*
    - Beer testing
    - Loudspeaker defect classifications
    - Tire testing
    - Predicting quality of plastics
    - Predicting quality of paper
    - Diesel knock testing, paint inspection (Volvo)

### 2.5.6. Medicine and Health Care Applications

- *Image Analysis*
    - Pap smear diagnostic aid
- *Drug Development*
    - Protein analysis for drug development
- *Resource Allocation*
    - Predict the severity of illness and use of hospital resources

### 2.5.7. Science and Engineering Applications

- *Chemical Engineering*
    - Optimize set of ingredients and processing attributes
    - Prediction of Secondary structure of Protein.

- Analysis of chemical data
- Spectroscopy

- *Electrical Engineering*
  - Optimized circuit routing
  - Timbre evaluation, model estimation and audio processing
  - Noise removal system
- *Weather*
  - Quantitative weather forecasting

### 2.6.8. Food Industry Applications

- *Odor/Aroma Analysis*
  - Odor analysis via electronic nose
  - Aroma/odor detection and recognition
  - Cooking control via electronic nose in microwave oven
- *Product Development*
  - Search for improved chemical formulations
- *Quality Assurance*
  - Beer testing
  - Purity testing, pulp wash detection
  - Quality control of potato chips

### 2.6.9. Transportation and Communication Applications

- *Transportation*
  - Fault detection
- *Communications*
  - Echo cancellation system : in use for more than 20 years (AT&T)

## 2.7. Summary

Neural networks consist of set of neurons or processing units which are suitable for many tasks. Neural Networks can modify its behavior in response to their environments. In other words they are capable of learning, and once they are trained they can make their decisions. The learning can be supervised or unsupervised. In supervised learning input-output vectors are presented to the network, whereas in unsupervised learning the network detects similarities between inputs and group them accordingly. This chapter has presented an overview of NN, well known NN architecture, their operational principles, and their learning algorithms such as Back propagation which is the most commonly used, its equations, advantages and disadvantages were listed.

# CHAPTER THREE: FACE RECOGNITION USING EIGENFACES

## 3.1. Overview

This chapter is dedicated to the explanation of a principal component analysis method. For principal component analysis eigenfaces method will be used for extracting feature vectors. The mathematical equations which were used for finding the eigenfaces are listed with explanation about it.

## 3.2. Eigenfaces approach

Much of the previous work on automated face recognition [24, 25] has ignored the issue of just what aspects of the face stimulus are important for face recognition. This suggests the use of an information theory approach of coding and decoding of face images, emphasizing the significant local and global features. Such features may or may not be directly related to our intuitive notion of face features such as the eyes, nose, lips, and hair.

In the language of information theory, the relevant information in a face image is extracted, encoded as efficiently as possible, and then compared with a database of models encoded similarly. A simple approach to extracting the information contained in an image of a face is to somehow capture the variation in a collection of face images, independent of any judgment of features, and use this information to encode and compare individual face images.

In mathematical terms, the principal components of the distribution of faces, or the eigenvectors of the covariance matrix of the set of face images, treating an image as point (or vector) in a very high dimensional space is sought. The eigenvectors are ordered, each one accounting for a different amount of the variation among the face images.

These eigenvectors can be thought of as a set of features that together characterize the variation between face images. Each image location contributes more or less to each eigenvector, so that it is possible to display these eigenvectors as a sort of ghostly face image which is called an "eigenface".

The data set face images is shown in figure 3.1. These face images where taken by the author himself from March 2004 till May 2004 and they belong to 15 students in Near East University. By taking the summation of all faces in the training set and dividing it by the number of faces on the set, average face image or the mean face of the whole training set will be produces as shown in figure 3.2. In Figure 3.3 the corresponding eigenvalues were drawn. Each eigenface deviates from uniform gray where some facial feature differs among the set of training faces. Eigenfaces can be viewed as a sort of map of the variations between faces.

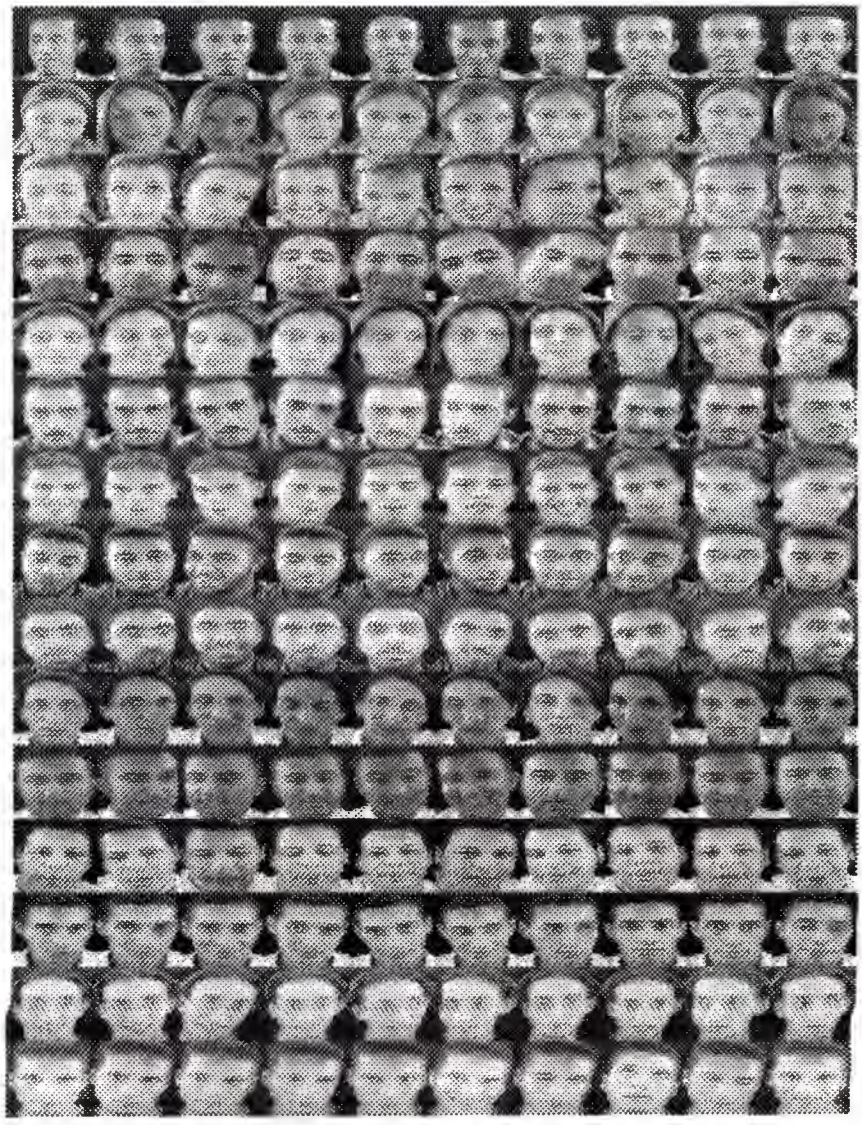

**Figure 3.1.** The data set face images of the author.

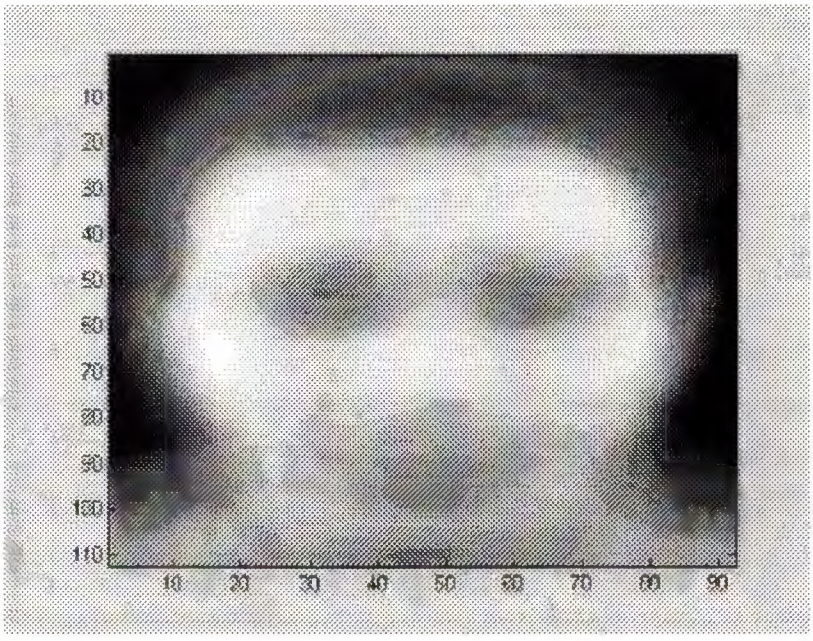

**Figure 3.2.** Average face image of the training set.

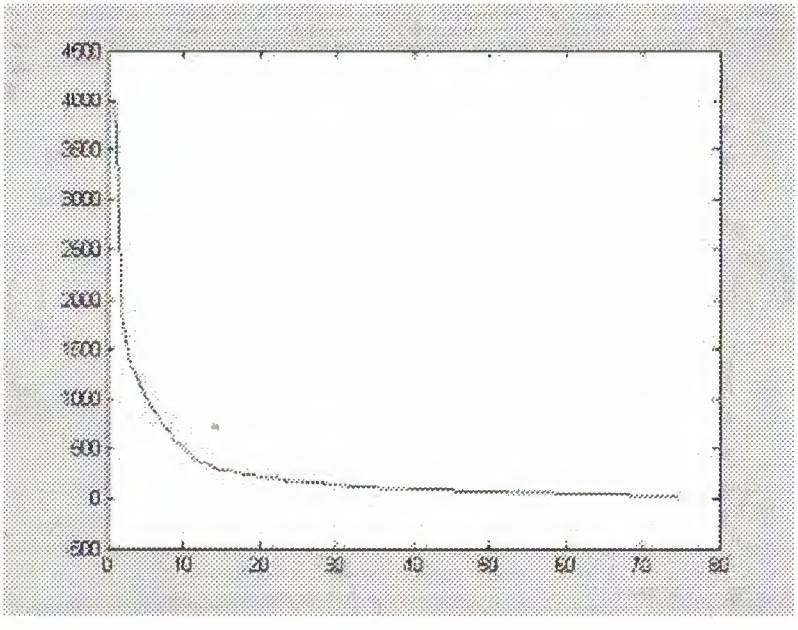

**Figure 3.3.** Eigenvalues corresponding to Eigenfaces.

Each individual face can be represented exactly in terms of a linear combination of the eigenfaces. Each face can also be approximated using only the "best" eigenfaces,

those that have the largest eigenvalues, and which therefore account for the most variance within the set of face images. The best M eigenfaces span an M-dimensional subspace which we call the "face space" of all possible images.

Kirby and Sirovich [26, 27] developed a technique for efficiently representing pictures of faces using principal component analysis. Starting with an ensemble of original face images, they calculated a best coordinate system for image compression, where each coordinate is actually an image that they termed an "eigenpicture". They argued that, at least in principle, any collection of face images can be approximately reconstructed by storing a small collection of weights for each face, and a small set of standard pictures (the eigenpictures). The weights describing each face are found by projecting the face image onto each eigenpicture.

In this thesis, we have followed the method which was proposed by M. Turk and A. Pentland [28] in order to develop a face recognition system based on the eigenfaces approach. They argued that, if a multitude of face images can be reconstructed by weighted sum of a small collection of characteristic features or eigenpictures, perhaps an efficient way to learn and recognize faces would be to build up the characteristic features by experience over time and recognize particular faces by comparing the feature weights needed to approximately reconstruct them with the weights associated with known individuals. Therefore, each individual is characterized by a small set of feature or eigenpicture weights needed to describe and reconstruct them. This is an extremely compact representation when compared with the images themselves.

### 3.3. Calculating Eigenfaces

Let a face image I(x,y) be a two-dimensional N × N array of 8-bit intensity values. An image may also be considered as a vector of dimension $N^2$ , so that a typical image of size 112 × 92 becomes a vector of dimension 10,304, or equivalently a point in 10,304-dimensional space. An ensemble of images maps to a collection of points in this huge space. Images of faces, being similar in overall configuration, will not be randomly distributed in this huge image space and thus can be described by a relatively low dimensional subspace. The main idea of the principle component analysis (or

Karhunen-Loeve expansion) is to find the vectors that best account for the distribution of face images within the entire image space.

These vectors define the subspace of face images, which we call "face space". Each vector is of length $N^2$ , describes an N × N image, and is a linear combination of the original face images. Because these vectors are the eigenvectors of the covariance matrix corresponding to the original face images, and because they are face-like in appearance, we refer to them as "eigenfaces". Some examples of eigenfaces which we had as an output of our program after applying the eigenface algorithm are shown in figure 3.4 and figure 3.5.

The eigenfaces in figure 3.4 are clearer as they have the highest eigenvalues, while the ones in figure 3.5 are very hard to identify as they have the worst eigenvalues.

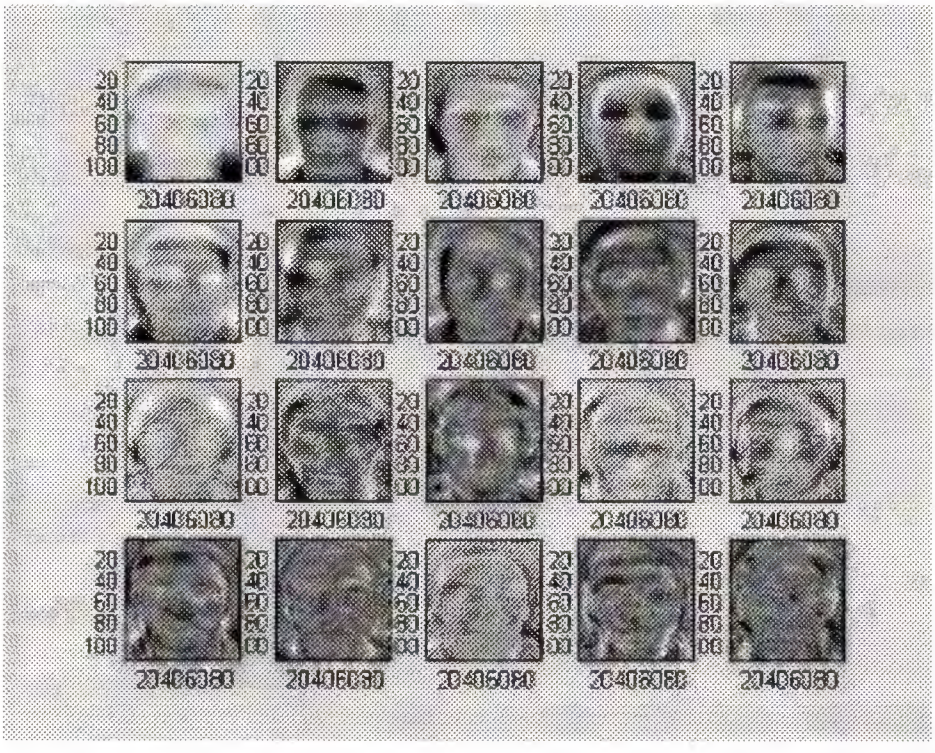

**Figure 3.4.** First 20 eigenfaces with the highest eigenvalues.

**Figure 3.5.** Last 20 eigenfaces with the lowest eigenvalues.

### 3.3.1. Definitions:

An N x N matrix A is said to have an eigenvector X, and corresponding eigenvalue λ if

$$AX = \lambda X \tag{3.1}$$

Evidently, Eq. (3.1) can hold only if

$$\det|A - \lambda I| = 0 \tag{3.2}$$

where I is the identity matrix

If Eq. (3.2) is expanded out, an $N^{th}$ degree polynomial in λ whose root are the eigenvalues. This proves that there are always N (not necessarily distinct) eigenvalues. Equal eigenvalues coming from multiple roots are called "Degenerate".

A matrix is called *symmetric* if it is equal to its transpose,

$$A = A^T \ or \quad a_{ij} = a_{ji} \tag{3.3}$$

It is termed *orthogonal* if its transpose equals its inverse,

$$A^T A = AA^T = I \tag{3.4}$$

Finally, a real matrix is called *normal* if it commutes with is transpose,

$$AA^T = A^T A \tag{3.5}$$

After giving some insight on the terms that are going to be used in the evaluation of the eigenfaces, we can deal with the actual process of finding these eigenfaces.

Let the training set of face images be $\Gamma_1,\Gamma_2,\ldots,\Gamma_M$ then the average of the set is defined by

$$\Psi = \frac{1}{M}\sum_{n=1}^{M}\Gamma_n \tag{3.6}$$

Each face differs from the average by the vector

$$\Phi_i = \Gamma_i - \Psi \tag{3.7}$$

An example of the training set is shown in figure 3.1, with the average face of the training set shown in figure 3.2.

This set of very large vectors is then subject to principal component analysis, which seeks a set of M orthonormal vectors, $U_n$ , which best describes the distribution of the data. The kth vector, $U_k$ , is chosen such that

$$\lambda_k = \frac{1}{M}\sum_{n=1}^{M}\left(U_k^T \Phi_n\right)^2 \tag{3.8}$$

is a maximum, subject to

$$U_I^T U_k = \delta_{Ik} = \begin{cases} 1, & if\ I = k \\ 0, & otherwise \end{cases} \tag{3.9}$$

The vectors $U_k$ and scalars $\lambda_k$ are the eigenvectors and eigenvalues, respectively of the covariance matrix

$$C = \frac{1}{M}\sum_{n=1}^{M}\Phi_n\Phi_n^T = AA^T \tag{3.10}$$

where the matrix $A = [\Phi_1\ \Phi_2 \ldots \Phi_M]$. The covariance matrix C, however is $N^2 \times N^2$ real symmetric matrix, and determining the $N^2$ eigenvectors and eigenvalues is an intractable task for typical image sizes. We need a computationally feasible method to find these eigenvectors.

If the number of data points in the image space is less than the dimension of the space ($M<N^2$), there will be only M-1, rather than $N^2$, meaningful eigenvectors. The remaining eigenvectors will have associated eigenvalues of zero. We can solve for the $N^2$ dimensional eigenvectors in this case by first solving the eigenvectors of an M × M matrix such as solving 16 × 16 matrix rather than a 16,384 × 16,384 matrix and then, taking appropriate linear combinations of the face images $\Phi_i$.

Consider the eigenvectors $v_i$ of $A^TA$ such that

$$A^TAv_i = \mu_i v_i \tag{3.11}$$

Premultiplying both sides by A, we have

$$AA^TAv_i = \mu_i Av_i \tag{3.12}$$

from which we see that $Av_i$ are the eigenvectors of $C = AA^T$.

Following these analysis, we construct the M × M matrix $L = A^TA$, where $L_{mn} = \Phi^T_m\Phi_n$, and find the M eigenvectors, $v_i$, of $L$. These vectors determine linear combinations of the M training set face images to form the eigenfaces $U_l$.

$$U_l = \sum_{k=1}^{M} v_{lk}\Phi_k, \quad l = 1,\ldots,M \tag{3.13}$$

With this analysis, the calculations are greatly reduced, from the order of the number of pixels in the images ($N^2$) to the order of the number of images in the training set (M). In practice, the training set of face images will be relatively small ($M << N^2$), and the calculations become quite manageable. The associated eigenvalues allow us to

rank the eigenvectors according to their usefulness in characterizing the variation among the images.

The success of this algorithm is based on the evaluation of the eigenvalues and eigenvectors of the real symmetric matrix L that is composed from the training set of images. Root searching in the characteristic equation, Eq. (3.2) is usually a very poor computational method for finding eigenvalues.

### 3.4 Using Eigenfaces to Classify a Face Image

The eigenface images calculated from the eigenvectors of L span a basis set with which to describe face images. Sirovich and Kirby [24, 25] evaluated a limited version of this framework on an ensemble of M = 115 images of Caucasian males digitized in a controlled manner, and found that 40 eigenfaces were sufficient for a very good description of face images. With M' = 40 eigenfaces, RMS pixel by pixel errors in representing cropped versions of face images were about 2%.

In practice, a smaller M' can be sufficient for identification, since accurate reconstruction of the image is not a requirement. Based on this idea, the proposed face recognition system lets the user specify the number of eigenfaces (M') that is going to be used in the recognition. For maximum accuracy, the number of eigenfaces should be equal to the number of images in the training set. But, it was observed that, for a training set of fourteen face images, seven eigenfaces were enough for a sufficient description of the training set members.

In this framework, identification becomes a pattern recognition task. The eigenfaces span an M' dimensional subspace of the original $N^2$ image space. The M' significant eigenvectors of the $L$ matrix are chosen as those with the largest associated eigenvalues.

A new face image ($\Gamma$) is transformed into its eigenface components (projected onto "face space") by a simple operation,

$$w_k = U_k^T(\Gamma - \Psi) \tag{3.14}$$

for k = 1,...,M'. This describes a set of point by point image multiplications and summations, operations performed at approximately frame rate on current image processing hardware, with a computational complexity.

The weights form a feature vector,

$$\Omega^T = [w_1\ w_2 \ldots . w_M] \tag{3.15}$$

that describes the contribution of each eigenface in representing the input face image, treating the eigenfaces as a basis set for face images. The feature vector is then used in a standard pattern recognition algorithm to find which of a number of predefined face classes, if any, best describes the face. The face classes $\Omega_i$ can be calculated by averaging the results of the eigenface representation over a small number of face images (as few as one) of each individual.

### 3.5 Rebuilding a Face Image with Eigenfaces

A face image can be approximately reconstructed (rebuilt) by using its feature vector and the eigenfaces as

$$\Gamma' = \Psi + \Phi_i \tag{3.16}$$

where

$$\Phi_i = \sum_{i=1}^{M} w_i U_i \tag{3.17}$$

is the projected image.

Eq. (3.16) tells that the face image under consideration is rebuilt just by adding each eigenface with a contribution of $w_i$ Eq. (3.17) to the average of the training set images. The degree of the fit or the "rebuild error ratio" can be expressed by means of the Euclidean distance between the original and the reconstructed face image as given in

$$\text{Rebuild error ratio} = \frac{\|\Gamma' - \Gamma\|}{\|\Gamma\|} \tag{3.18}$$

It has been observed that, rebuild error ratio increases as the training set members differ heavily from each other. This is due to the addition of the average face image. When the members differ from each other (especially in image background) the average face image becomes messier and this increases the rebuild error ratio.

### 3.6 Summary of the Eigenface Approach

The Eigenfaces approach to face recognition can be summarized in the following steps:

1. Form a face library that consists of the face images of known individuals.
2. Choose a training set that includes a number of images (M) for each person with some variation in expression and in the lighting.(see figure 3.1)
3. Calculate the M × M matrix L, find its eigenvectors and eigenvalues, and choose M' eigenvectors with highest associated eigenvalues.
4. Combine the normalized training set of images according to Eq. (3.13) to produce M' eigenfaces. Store these eigenfaces for later use.(see figure 3.4 and figure 3.5)
5. For each member in the face library, compute and store a feature vector according to Eq. (3.15).

### 3.7. Comparison of the Eigenfaces Approach to Feature Based Face Recognition

These two different approaches are compared based on the following aspects of face recognition:

- **Speed and simplicity.**

Feature based face recognition involves complex computations such as deformable templates and active contour models. The evaluation of these parameters is very time consuming even on today's computers. Eigenfaces approach is superior in its speed and reasonably simple implementation. In Eq. (3.14), it is seen that the evaluation of a feature vector involves merely additions and multiplications. On a machine that is capable of executing an addition and a multiplication in one clock cycle, this feature evaluation and comparison can be done in real time.

- **Learning capability.**

Feature based face recognition systems are generally trained to optimize their parameters in a supervised manner. That is, the system designer presents known individuals to the system and checks system response. In the eigenfaces approach, training is done in an unsupervised manner. User selects a training set that represents the rest of the face images. Eigenfaces are obtained from the training set members and feature vectors are formed.

- **Face background.**

Background of the face images is extremely important in the eigenface approach. Eigenfaces and feature vectors are evaluated by image multiplication and additions. As a result of this, entire information contained in the face image is used. If this information changes due to face background, recognition performance can significantly decrease. In order to avoid this, a "background removal" algorithm can be implemented in a preprocessing step. Feature based face recognition algorithms can be less sensitive to face background in case, they generally locate face contours in order to extract facial features.

- **Scale and orientation.**

The recognition performance decreases quickly as head size or orientation is misjudged. The head size and orientation in the input image must be close to that of the eigenfaces for the system to work well. In order to overcome this problem, multiscale eigenfaces can be used or the head can be removed from the rest of the image, and then scaled or rotated to meet the specifications of the eigenfaces. Again, feature based face recognition algorithms can score better in this comparison because they find facial features by using deformable templates and active contour models that are less sensitive to scale and orientation.

- **Presence of small details.**

Feature based face recognition algorithms can suffer when some details are present on the face image such as dark glasses or beards. For a feature based face recognition system, it is quite impossible to extract the features that are related to the eyes when a dark glass is present on the face. Also, active contour models can suffer when a beard is present on the face while locating the face contour. Eigenfaces approach excels in this aspect of face recognition. Small changes in face images such as glasses, beards or moustaches does not cause a decrease in the face recognition performance because the information that is present in the rest of the face image makes it enough to be classified correctly.

### 3.8. Summary

In this chapter the idea of eigenfaces approach were presented in details. The mathematical equations for finding eigenfaces, eigenvectors and projecting it in Eigenspace were listed. After calculating eigenfaces, the feature vectors are calculated for the faces in the database. These feature vectors can be used as neural network training data.

# CHAPTER FOUR: THE DEVELOPED SOFTWARE & RESULTS

## 4.1. Overview

The aim of this chapter is to introduce the usage of neural networks for face recognition. It shows the process of training and simulation of the neural networks using Matlab neural networks toolbox with summary of the procedures. The features and parameters of the proposed system where discussed, while all the different results of the system after manipulating its parameters are calculated and listed tables.

## 4.2. Database Details

The 150 face images where taken against a dark background by the author between March 2004 and May 2004 belong to 15 students in Near East University. Each student has had his/her face image captured 10 times. Each of the 10 images represents a person with different facial expressions (open / closed eyes, smiling / not smiling), facial details (glasses / no glasses), varying lighting, and head pose (tilting and rotation). An example is shown in figure 4.1. All the images in the database were preprocessed by resizing them to 92 by 112 and convert them to grayscale images with bmp format (see figure 4.2). The whole database is shown in figure 4.3.

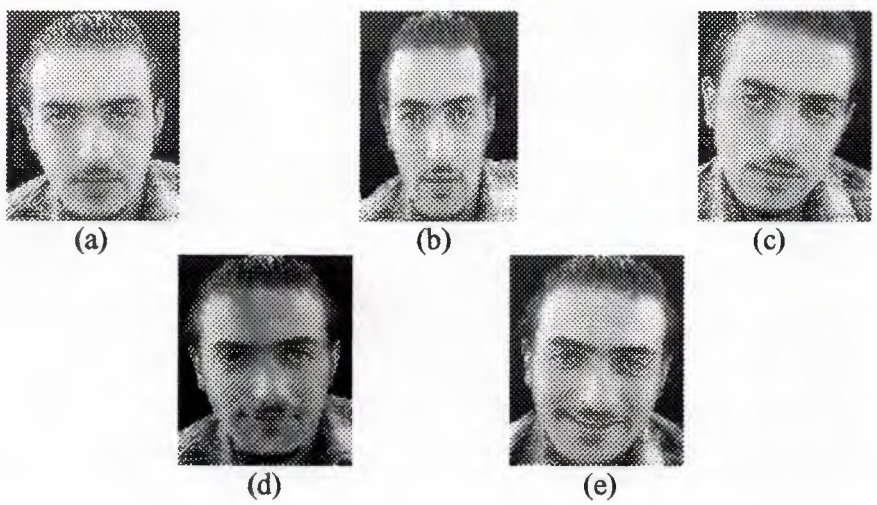


**Figure 4.1** (a) Original face image. (b) Scale variance. (c) Orientation variance. (d)Illumination variance. (e) Change in expression.

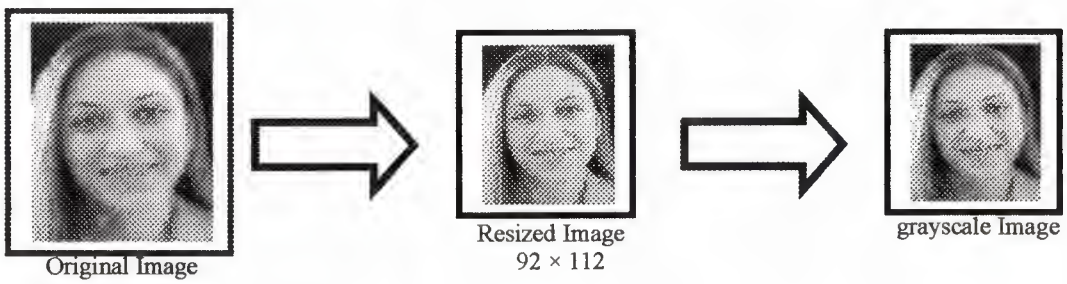


**Figure 4.2** Preprocessing Procedures

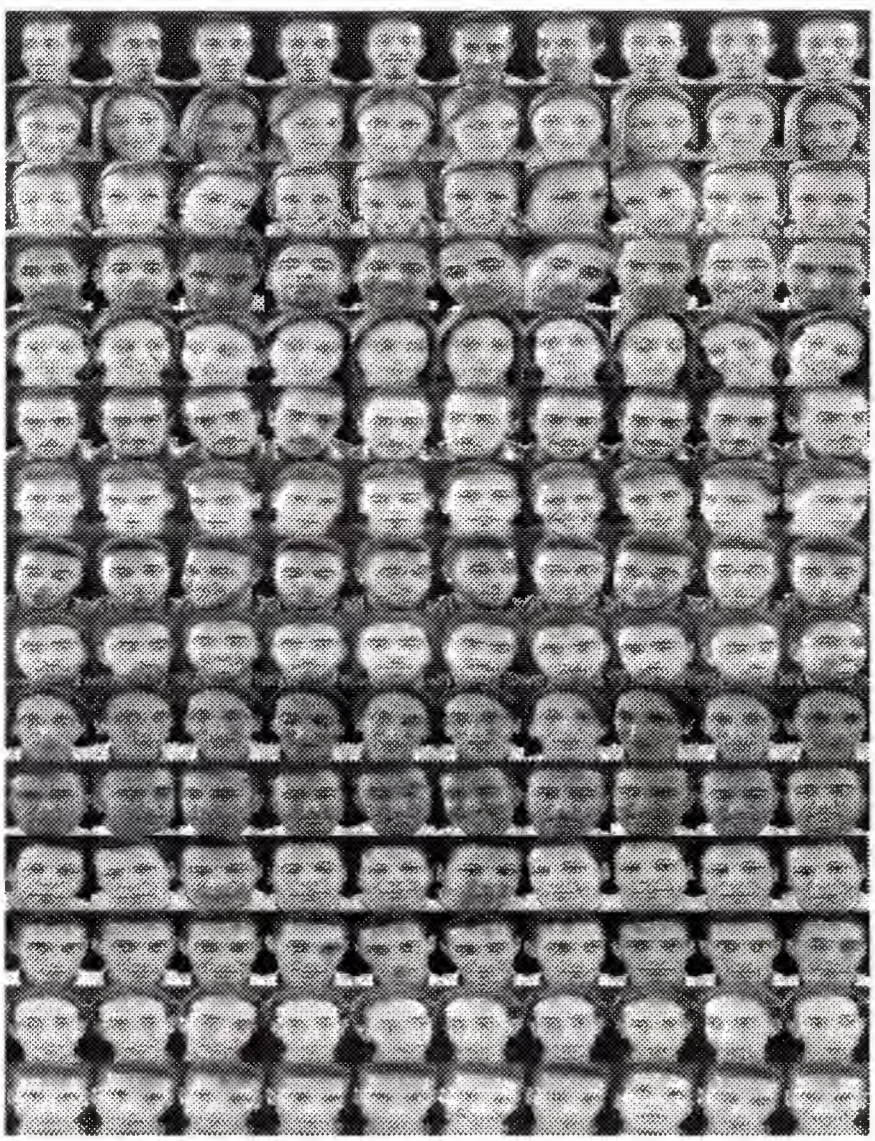

**Figure 4.3** The data set face images of the author.

### 4.2.1. Camera Details

The used camera were a BENQ digital camera-1300 is equipped with the following features:

| | |
|---|---|
| **Total pixels** | 1.36M |
| **PC Camera function** | Yes |
| **Digital Camera Function** | Yes |
| **Image Resolution** | 1280 * 1024 Pixel, 1024 * 768 Pixel, 640 * 480 Pixel |
| **Lens** | Free focus, 50 mm equivalent |
| **Focusing Range** | Normal: 1.5m ~ Infinity , Macro: 30~50cm, Text: 11~13cm |
| **Digital Zoom** | Yes |
| **Flash** | Built-in |
| **File Format** | Compressed JPEG |
| **Connectivity** | Mini USB |
| **LCD Monitor** | Status LCD |
| **System Requirements** | Win98/2000/ME/XP |

## 4.3. Usage of Neural Network for Recognition

### 4.3.1. Feedforward neural networks

In this kind of networks, the neurons are organized in the form of layers. The neurons in a layer get input from previous layer and feed their output to the next layer. In this kind of networks connections to the neurons in the same or previous layers are not permitted. Figure 4.4 shows the neural network architecture for all of the 15 people.

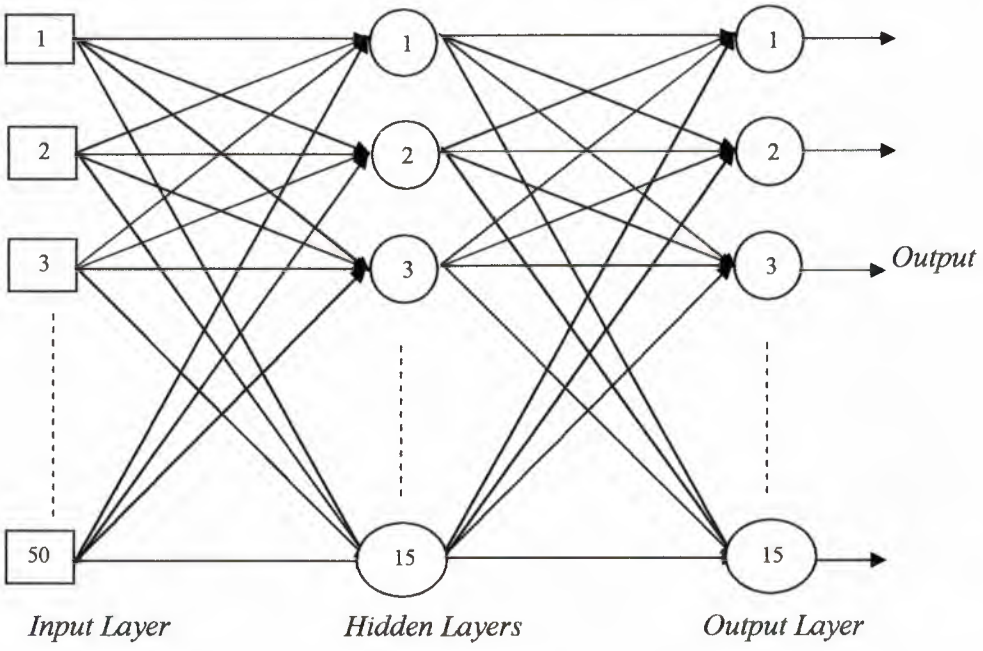


**Figure 4.4** Architecture of the proposed Neural Network.

### 4.3.2. Training and Simulation of Neural Network for Recognition

Neural networks have been trained to perform complex functions in various fields of application including pattern recognition, identification, classification, speech, vision, and control systems.

In this thesis, there is one neural network for all people in the database (see figure 4.4). After calculating eigenfaces, the feature vectors are calculated for the faces in the database. These feature vectors (each feature vector consists of 75 elements) are used as inputs to train the neural network. According to figure 4.5 it's clear that the eigenvalues approach to zero nearly after the first 50 values and according to this the first 50 values of each feature vector will be used as input for the neural network. Figure 4.6 shows the schematic diagram for the neural network training.

When new image is taken for recognition, its feature vectors are calculated from the eigenfaces found before, and this image gets its new descriptors. These new descriptors are fed to the neural network and the network is simulated with these descriptors. The network outputs are compared. By looking at the maximum output the new face is decided to belong to the person with this maximum output. For example, if the output is O = [0 0 1 0 .... 0]. Then the person is the third person in the database. See figure 4.7.

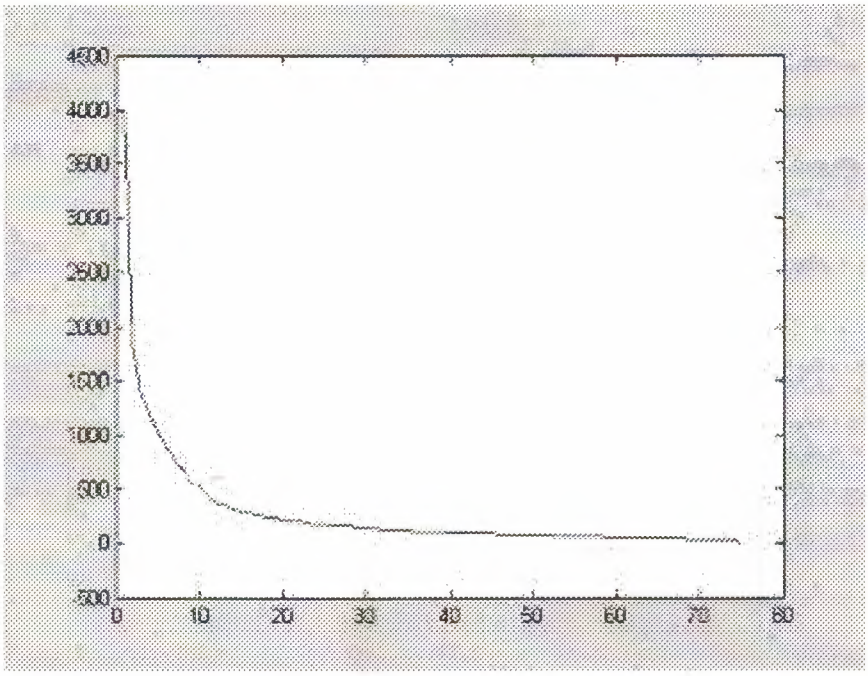


**Figure 4.5.** Eigenvalues corresponding to eigenfaces

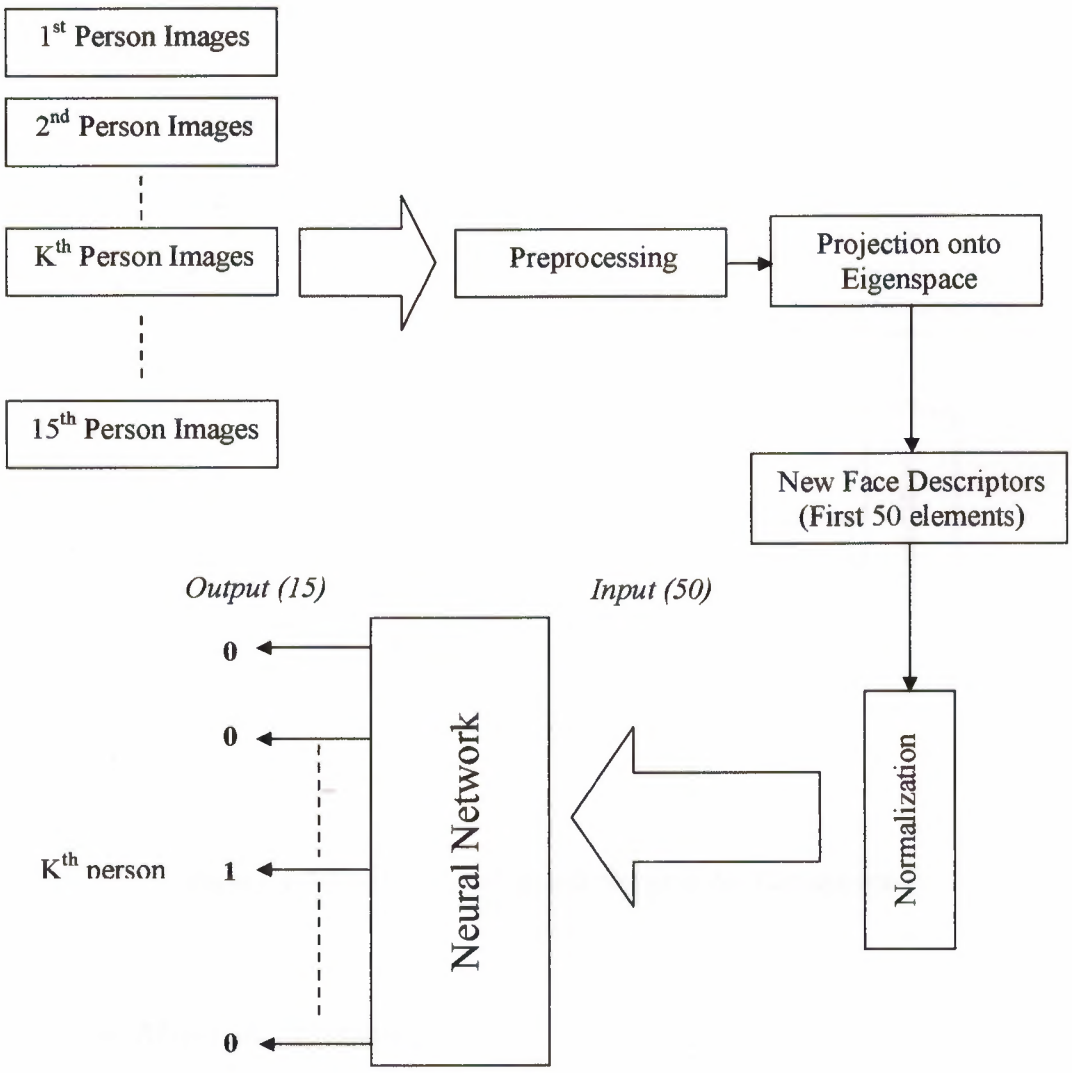


**Figure 4.6.** Training of the Neural Network

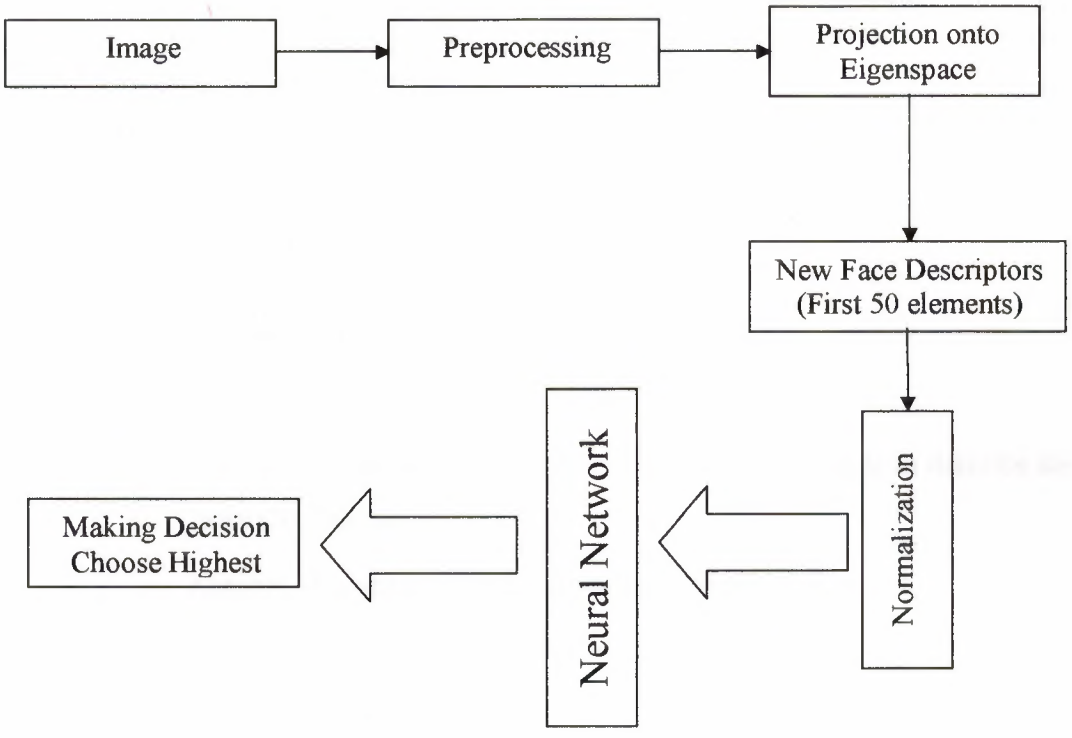


**Figure 4.7** Simulations of Neural Network for Recognition

### 4.4. Neural Networks Parameters

The proposed method is tested on the author database. This database has more than one face images with different conditions (expression, illumination, scale... etc.)

The following section, detailed information for this database and its performance results for the proposed face recognition method are given.

There is one neural network used for this database with outputs equal to number of people in it. The initial parameters of neural network used in tests are as follows.

*Neural Network Parameters:*

- Type: feedforward back propagation network.
- Backprop weight/bias learning function: learngdm

  Learning occurs according to LEARNGDM's learning parameters, shown here with their default values:

  - Learning rate: LP.lr = 0.01.
  - Momentum constant: LP.mc = 0.9.
- Number of layers: 3 (input layer, one hidden layer, output layer)
  - Number of neurons in input layer: number of eigenfaces to describe the faces (50 neurons).
  - Number of neurons in hidden layer :15 neurons
  - Number of neurons in output layer: 15 neurons
- Transfer function : Tansig

  TANSIG Hyperbolic tangent sigmoid transfer function. It calculates its output according to:

  $$n = 2/(1+\exp(-2*n))-1 \quad (4.1)$$
- Training function: Trainlm

  Trainlm is a network training function that updates weight and bias values according to Levenberg-Marquardt optimization.
- Number of iteration used in training: 300 iterations.
- Performance function: mse (Mean squared error ).

## 4.5. Summary of the Neural Network Phase

The Neural Network Phase using eigenfaces approach to face recognition can be summarized in the following steps:

6. Create one neural network for all people in the database.
7. Train this neural network using features as inputs.
8. For each new face image to be identified, calculate its feature vector according to Eq. (3.15).
9. Use these 50 element length feature vectors or new descriptors as network inputs and simulate the network with these inputs.
10. Select the element with the maximum value in the output vector; this element will be treated as one while the rest are zeros. The order of this element is same as the order of the persons in the database. So if we have the output as [0 0 1 0……0], this means that the person is number 3 in the database.

## 4.6. Experimental Results

In author database there are 10 different images of each 15 distinct persons. For some persons, the images were taken at different facial expressions (open / closed eyes, smiling / not smiling), facial details (glasses / no glasses), varying lighting, and head pose (tilting and rotation). All images were taking against a dark homogenous background. Figure 4.3 shows the whole set of 15 individuals, 10 images per person.

The neural network with the parameters mentioned in section 4.4 where used for successful training. An example of the output of simulation of the neural network with the $10^{th}$ person face image is shown in figure 4.8. The horizontal axis shows the number of people in the database which is 15 people. After training, the maximum output of the neural network refers to the $10^{th}$ person in the dataset.

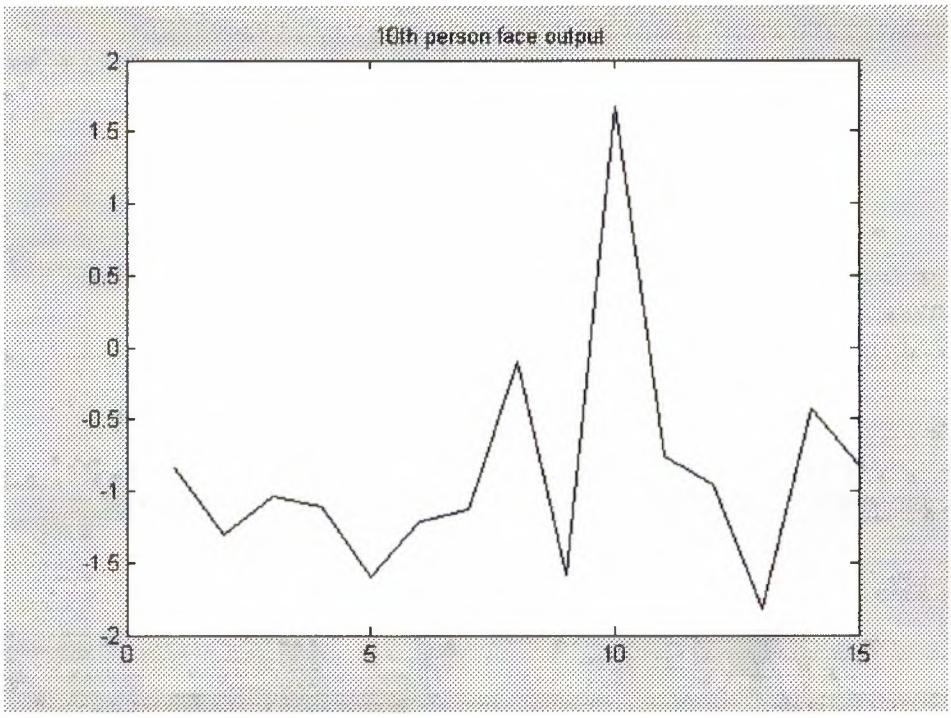


**Figure 4.8** Neural Network Simulation Output for $10^{th}$ Person Face

A graph of the neural network outputs after being trained is shown in figure 4.9. After the neural network has been trained it has been tested with the remaining images of each person of the face database as shown in table 4.1.

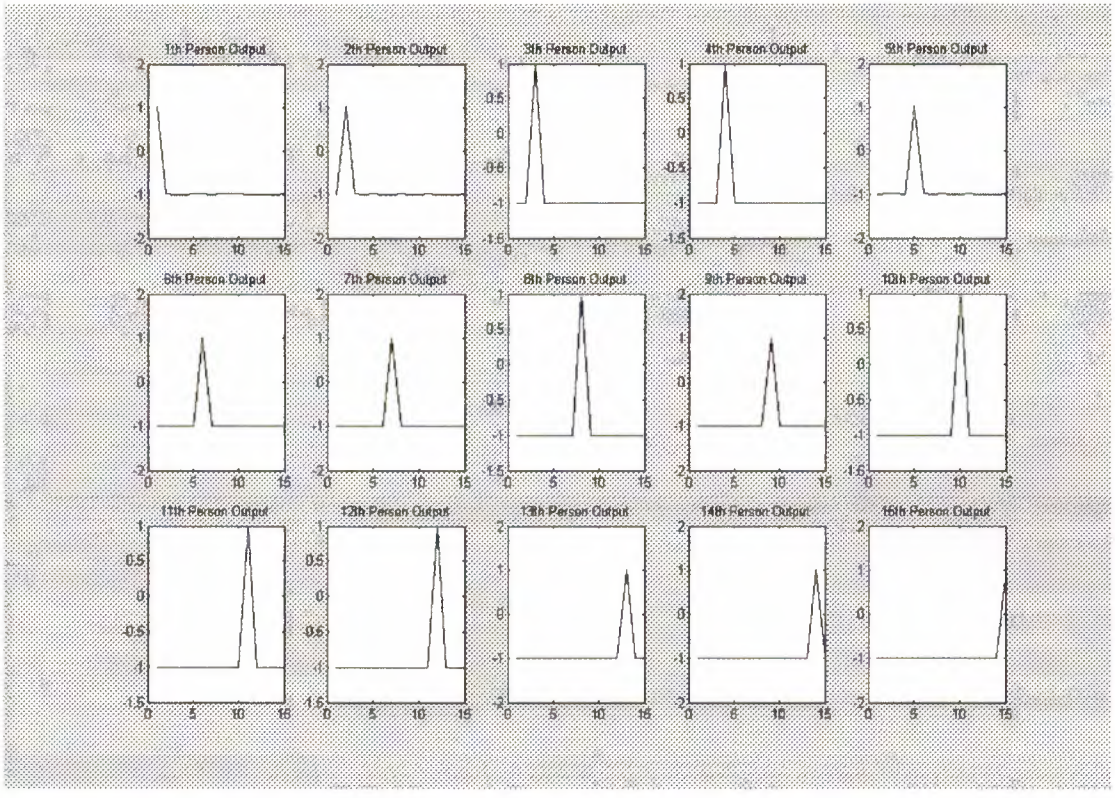


**Figure 4.9** the Neural Network Outputs after Training

Since the number of people in the database is 15, one neural network for all 15 people was created with 15 outputs. Within the ten images, the first 5 are used for training the neural networks, and then the neural network is tested using the remaining five images.

The recognition performance increase with the number of faces used to train the networks, so the recognition rates are calculated, with using different number of faces used in training, and given in table 4.1.

**Table 4.1** Recognition Rate Using Different Number of Training and Testing Images

| Number Of Images Used In Training (Per Person) | Number Of Images Used In Testing (Per Person) | Recognition Rate (%) | Number of unrecognized faces |
|---|---|---|---|
| 1 | 9 | 10 | 135 faces / 150 faces |
| 2 | 8 | 28.67 | 107 faces / 150 faces |
| 3 | 7 | 56.67 | 65 faces / 150 faces |
| 4 | 6 | 80 | 30 faces / 150 faces |
| 5 | 5 | 98 | 3 faces / 150 faces |

The number of eigenfaces that are used to describe the faces also affects the recognition rate, so the tests are done with different number of eigenfaces and the results are given in Table 4.2

**Table 4.2** Recognition Rate Using Different Number of Eigenfaces

| Number Of Eigenfaces | Number Of Neurons In Hidden Layer | Recognition Rate (%) |
|---|---|---|
| 15 | 15 | 76 |
| 25 | 15 | 88.7 |
| 50 | 15 | 98 |

Number of images used in training : 5

Number of images used in testing : 5

The values of learning rate and momentum constant also affect the recognition rate, so the tests are done with different values of learning rate and momentum constant and the results are given in Table 4.3.

**Table 4.3** Recognition Rate Using Different Learning Rates and Momentum Constants

| Learning rate | Momentum costant | Recognition Rate (%) |
|---|---|---|
| 0.01 | 0.95 | 95.3 |
| 0.15 | 0.6 | 95.3 |
| 0.001 | 0.95 | 94 |
| 0.01 | 0.85 | 96.67 |
| 0.15 | 0.8 | 96.67 |
| 0.005 | 0.95 | 97.3 |
| 0.01 | 0.9 | 98 |

Number of images used in training: 5

Number of images used in testing : 5

The maximum result we had was 98 % which reflect the power of the using Neural Networks in recognition instead of Euclidian distance method in recognition where classification is performed by comparing the feature vector of the face library members with the feature vector of the input face image. This comparison is base on the Euclidian distance between the two persons to be smaller than a user defined threshold.

## 4.7. Summary

In the previous chapter it was shown how the images where preprocessed and projected into Eigenspace to have its feature vectors. This chapter explained how these feature vectors and after normalizing them to be between 0 and 1 are used as inputs to train the neural network. Then the highest output among all neural network outputs will reflect that this face belongs to the person who has the corresponding order in the database. Results of the developed system after manipulating some of the system parameters such as eigenfaces number, learning rate, momentum constant, or the training and testing images number listed in tables.

# CONCLUSION

Principal component analysis, approaches to the face recognition problem by means of information theory concepts. The most relevant information that is contained in a face image is extracted. Eigenfaces method is a principal component analysis approach, where the eigenvectors of the covariance matrix of a small set of characteristic pictures are sought. These eigenvectors are called eigenfaces due to their similarity of face images.

Recognition is performed by assigning weight vectors to face images, according to their contributions to the face space spanned by the eigenfaces. This approach excels in its speed, simplicity and learning capability.

A robust face recognition system should be insensitive to

- Changes in illumination,
- Changes in head orientation and scale,
- Presence of facial details such as glasses, beards,
- Face background.

Experimental results have shown that, the proposed face recognition method was very sensitive to face background and head orientations. Changes in the illumination did not cause a major problem to the system. Besides, presence of small details such as glasses was too far from being a real challenge to the system. There exists a trade off between the correct recognition rate and the threshold value. As the number of eigenfaces involved in the recognition process increases, misclassification rate begins to decrease, possibly resulting in misses.

The program was developed and successfully tested using the author's own face database with acceptable recognition rate reach 98 % after manipulating some parameters like number of trained and tested faces per person, Eigenfaces, momentum rate and learning rate.

# REFERENCES


[1] Galton., F., "Personal Identification and Description", 1,1 Nature, pp.173-177,21 June1988

[2] Galton., F., "Personal Identification and Description-II", Nature, 201-203,28 June1988

[3] Gonzalez, R. C., and Tou, J. T., "Pattern Recognition Principles", Addison-Wesley Publishing Company, 1974.

[4] Carey, S., and Diamond, R., "From Piecemeal to Configurational Representation of Faces", Science 195, pp. 312-313, 1977.

[5] Bledsoe, W. W., "The Model Method in Facial Recognition", Panoramic Research Inc. Palo Alto, CA, Rep. PRI:15, August 1966.

[6] Bledsoe, W. W., "Man-Machine Facial Recognition", Panoramic Research Inc. Palo Alto, CA, Rep. PRI:22, August 1966.

[7] Fischler, M. A., and Elschlager, R. A., "The Representation and Matching of Pictorial Structures", IEEE Trans. on Computers, c-22.1, 1973.

[8] Yuille, A. L., Cohen, D. S., and Hallinan, P. W., "Feature Extraction From Faces Using Deformable Templates", Proc. of CVPR, 1989.

[9] Kohonen, T., "Self-Organization and Associative Memory", Berlin: Springer-Verlag, 1989.

[10] Kohonen, T., and Lehtio, P., "Storage And Processing of İnformation in Distributed Associative Memory Systems", 1981.

[11] Fleming, M., and Cottrell, G., "Categorization of Faces Using Unsupervised Feature Extraction", Proc. of IJCNN, Vol. 90(2), 1990.

[12] Kanade, T., "Picture Processing System By Computer Complex and Recognition of Human Faces", Dept. of Information Science, Kyoto University, 1973.

[13] Burt, P., "Smart Sensing within A Pyramid Vision Machine", Proc. of IEEE, Vol. 76(8), pp. 139-153, 1988.

[14] Lin S., Kung S., Lin L., "Face Recognition: Detection by Probabilistic Decision-Based Neural Network", IEEE Trans. on NN, Vol. 8, No. 1, January 1997.

[15] Lawrence, S., Giles, C., Tsoi, C., Back, A., "Face Recognition: A Convolutional Neural Network Approach ", IEEE Trans. on NN, Vol. 8, No. 1, January 1997.

[16] Zhang, M., Fulcher, J., "Face Recognition Using Artificial Neural Network Group-Based Adaptive Tolerance (GAT) Trees", IEEE Trans. on NN, Vol. 7, No. 3, May, 1996.

[17] Bouttour, H., "Neural Nets for Human Face Recognition" in Proc. Int. Joint Conf. Neural Networks, Baltimore, MD, Vol. III, pp 700-704, 1992.

[18] Flocchini, M., "Combining Image Processing Operators and Neural Networks in A Face Recognition System", Pattern Recognition Artificial Intell., Vol. 6 No. 2 pp. 447-467, 1992.

[19] Sanker, A., Mammone, R. J., "Growing and Purning Neural Network Tree Network", IEEE Trans. Computers, Vol 42, No. 3, pp 291-299, 1993.

[20] Kerin, M. A., Stonham, T. J., "Face Recognition Using A Digital Neural Network With Self-Organising Capabilities", IEEE Trans., 1990.

[21] Khashman, A., M.Sc. course handout. NEU. Northern Cyprus.

[22] Demuth, H. B., Raele, M. H., Neural Network Toolbox User's Guide for Use with Matlab. the Mathworks Inc., 1998.

[23] Hagan, M. T., Menhaj M., "Training Feedforward Neural Networks with the Marquardt Algorithm", IEEE Trans. on Neural Networks, Vol. 5, No. 6, pp 989-993, 1994.

[24] Harmon, L. D., and Hunt, W. F., "Automatic Recognition of Human Face Profiles", Computer Graphics and Image Processing, Vol. 6, pp. 135-156, 1977.

[25] Kaufman, G. J., and Breeding, K. J, "The Automatic Recognition of Human Faces From Profile Silhouettes", IEEE Trans. Syst. Man Cybern., Vol. 6, pp. 113-120, 1976.

[26] Kirby, M., and Sirovich, L., "Application Of The Karhunen-Loeve Procedure for The Characterization Of Human Faces", IEEE PAMI, Vol. 12, pp. 103-108, 1990.

[27] Sirovich, L., and Kirby, M., "Low-Dimensional Procedure for The Characterization Of Human Faces", J. Opt. Soc. Am. A, 4, 3, pp. 519-524, 1987.

[28] Turk, M., and Pentland, A., "Eigenfaces for Recognition", Journal of Cognitive Neuroscience, Vol. 3, pp. 71-86, 1991.

[29] Atalay, I., and Ballikaya, F.," Face Recognition Using Eigenfaces", M.Sc. Thesis, ITU, 1996.

## Appendix A

- **Program Source Code**

```
%*****************************************************************************************%
%*****************************************************************************************%
%*********                                                                       ***********%
%*********   FACE RECOGNITION USING EIGENFACES AND NEURAL NETWORKS               ***********%
%*********                        MASTER THESIS                                  ***********%
%*********                        ALAA ELEYAN                                    ***********%
%*********                        JULY,2004                                      ***********%
%*********                        NICOSIA                                        ***********%
%*****************************************************************************************%
%*****************************************************************************************%


%*****************************************************************************************%
%*****************************************************************************************%
%*******                                                             *************%
%*******        PREPEARING THE IMAGES FOR NEURAL NETWORK INPUT       *************%
%*******                                                             *************%
%*****************************************************************************************%
%*****************************************************************************************%

clear all;

%*********************************average**********************

cd('faces')
[X,MAP]=imread('face1.bmp');
[a,b]=size(X);
n=5;
nump=15;
I = zeros(a,b);
disp('Reading the 15 Person Faces')
disp('')
for i=1:nump
   for j=1:n
      file = ['face' int2str((i-1)*10+j) '.bmp'];
      [X,MAP]=imread(file);
      X=ind2gray(X,MAP);
      I = I + double(X);
   end
  disp(strcat(int2str(i),'th Person'))
end
AV = I/(nump*n);

figure
imagesc(AV)
colormap(gray);
title('average image of 200 images from database');
```

```
%**********************************calculate A**************************

%initialize matrix to hold difference vectors
A=zeros(a*b,nump*n);
disp('Differences')
cnt=0;
for i=1:nump
   for j=1:n
      im = ['face' int2str((i-1)*10+j) '.bmp'];
      [X1,MAP1]=imread(im);
      X1=ind2gray(X1,MAP1);
      V = double(X1) - AV;
      VEC=reshape(V,1,a*b);
      VECT = transpose( VEC);     % column vector


      cnt=cnt+1;
      A(:,cnt)=VECT;              %covariance matrix holder


  end
  disp(strcat(int2str(i),'th Person'))
end

%**********************************Eigenfaces**************************

disp('Eigenfaces calculations')
  AT=transpose(A);
 [V,D] = eig(AT*A);


U= A*V;

figure;

for i=1:20
   UTEMP=U(:,i);
   utemp =reshape(UTEMP,a,b);
   subplot(4,5,i), imagesc(utemp);
end
title('First 20 eigenfaces');
colormap(gray);
figure;

for i=40*n:-1:40*n-19
   UTEMP=U(:,i);
   utemp =reshape(UTEMP,a,b);
   subplot(4,5,i-(40*n-20)), imagesc(utemp);
end
```

```
title('Last 20 eigenfaces');
colormap(gray);

%************************** Projection ********************************

disp('--')
disp('Projections')
disp('--')

for i=1:nump
   for j=1:5
      [ArIm,Mp2]=imread(strcat(strcat('face',int2str((i-1)*10+j),'.bmp')));
      ArIm=ind2gray(ArIm,Mp2);
      DiffIm=double(ArIm)-AV;

      GAMMA=reshape(DiffIm,a*b,1);

      m=nump*5;
      omega=zeros(m,1);

      for k=nump*n:-1:nump*n-m+1
         W=transpose(U(:,k))*GAMMA;
         omega(nump*n-k+1)=W;
      end

      fnlfcs(:,i,j)=omega(:);%fnl(:);
   end
   disp(strcat(int2str(i),'th Person'));
end


%*** PREPEARED DATA NORMALIZATION FOR USING AS NN INPUT  ****%

disp('--')
disp('Normalization')
disp('--')

mx=max(max(reshape(fnlfcs,1,10*nump*nump*10)));
mn=min(min(reshape(fnlfcs,1,10*nump*nump*10)));
fnlfcs=((fnlfcs-mn)/(mx-mn))*1;
```

```
%*****************************************************************************************%
%*****************************************************************************************%
%**********                                                              ***************%
%**********              NEURAL NETWORK TRAINING PHASE                   ***************%
%**********                                                              ***************%
%*****************************************************************************************%
%*****************************************************************************************%

disp('--')
disp('------ Neural Network Training');
disp('--')
disp('Init');
disp('--')

% input layer
numin=50;
%hidden layer and output
numo=[15 nump];
%transfer functions
tfns={'tansig' 'purelin'};
%iteration for the first phase
nepochs=50;
trfn='trainlm';
learnfn='learngdm';
perf='mse';
%lp.lr = 0.15;
%lp.mc = 0.6;

mnx=[0 1];
for i=1:numin
   Mnx(i,:)=mnx;
end
net=newff(Mnx,numo,tfns,trfn,learnfn,perf);
net.trainParam.epochs=nepochs;
net.trainParam.show = 1;
net.trainParam.goal = 1e-5;
%*******************
%*** training using the average images
%disp('Preparing NN Inputs and Target');
%disp('-')
%disp('--')
%for i=1:nump
%    P(i,:)=reshape(sum(fnlfcs(1:numin,i,1:5),3)/5,numin,1)';
%end
%T=ones(nump)*-1+eye(nump)*2;
```

```
%************************
%********* training using all the images


disp('Preparing NN Inputs and Target');
   disp('.')
   disp('..')
k=1;
T=ones(nump*5,nump)*-1;
for i=1:nump
for j=1:5
      P(k,:)=reshape(fnlfcs(1:numin,i,j),numin,1)';
      T(k,:)=-1;
      T(k,i)=1;
      k=k+1;
   end

end

[net,tr,Y,E]=train(net,P',T');

disp('Network Training Finished')


%*****************************************************************************************%
%*****************************************************************************************%
%**********                                                             **********%
%**********              NEURAL NETWORK TRAINING OUTPUT                 **********%
%**********                                                             **********%
%*****************************************************************************************%
%*****************************************************************************************%


figure;
dd=0;
for d=1:5:75
dd=dd+1;
          subplot(3,5,dd);
      plot(Y(d,:))
      title(strcat(int2str(dd),'th Person Output'));
   end
end
```

```
%*****************************************************************************************%
%*****************************************************************************************%
%**********                                                              ***************%
%**********                NEURAL NETWORK TESTING OUTPUT                 ***************%
%**********                                                              ***************%
%*****************************************************************************************%
%*****************************************************************************************%

for D=1:15
  figure
for d=1:10
            subplot(2,5,d);
      A=reshape(fnlfcs(1:numin,D,d),numin,1);
            y=sim(net,A);
      plot(y)
      title(strcat(strcat('face  ',int2str(d)),strcat(' / '),strcat(int2str(D),'  th Person')));
   end
end


%************************** END OF THE PROGRAM **********************
```