NEAR EAST UNIVERSITY

1988

GRADUATE SCHOOL OF APPLIED AND SOCIAL SCIENCES

FEATURE-AVERAGE BASED FACE RECOGNITION USING NEURAL NETWORKS

Akram ABU GARAD

Master Thesis

Department of Electrical and Electronic Engineering

Nicosia - 2005

Akram ABU GARAD: Feature-Average Based Face Recognition Using Neural Networks

Approval of Director of Graduate School of Applied and Social Sciences

Prof. Dr. Fahreddin M. SADIKOĞLU

[illegible]

We certify this thesis is satisfactory for the award of the degree of Master of Science in Electrical and Electronic Engineering

Examining Committee in Charge:

Prof. Dr. Fahreddin M. SADIKOĞLU, [illegible]	Dean of Engineering Faculty, NEU
Assoc. Prof. Dr. Rahib ABIYEV, [illegible]	Vice Chairman of Computer Engineering Department, NEU
Assist. Prof. Dr. Doğan HAKTANIR, [illegible]	Electrical & Electronic Engineering Department, NEU
Assoc. Prof. Dr. Adnan KHASHMAN, [illegible]	Chairman of Electrical & Electronic Engineering Department, Supervisor, NEU

ACKNOWLEDGEMENTS

First and foremost, I would like to sincerely thank my supervisor, Assoc. Prof. Dr. Adnan KHASHMAN, for his invaluable supervision, support and encouragement, which have helped me to complete this work.

Thanks also to my family for their constant love and support throughout the year.

Finally, I would like to thank my brother Tayser Abu Jarad for his spiritual and financial support.

ABSTRACT

The technology of face recognition has become mature lately. Systems for face recognition have become true in real life applications. Face recognition relates to identifying or verifying individuals by their faces.

This thesis attempts to develop an automatic face recognition system based on the important facial features from multi-expression sequence face images.

The design of this automated face recognition system is based on feature-average based using back propagation neural networks. The face recognition system has been separated into three major phases; feature extraction, averaging and face recognition. Feature extraction phase has been implemented on assumption the locations of the essential features of the face are known. Averaging process is the most important phase in this work. The average phase has been implemented to reduce the dimensions of features matrices and to take the mean of the features from multi expression faces. The face recognition classification has been applied using back propagation neural networks. The system has been simulated using Matlab software tools. A real-life application using the developed system has been implemented using 90 image sequences obtained from 15 subjects showed an overall rate of 100% recognition and accuracy of 93%.

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

AFR: Automatic Face Recognition

ANN: Artificial Neural Network

ATM: Automatic Transfer Machine

BP: Back Propagation

HCI: Human Computer Interaction

HMM: Hidden Markov Model

ICA: Independent Component Analysis

LDA: Linear Discriminant Analysis

LLE: Locally Linear Embedding

LMS: Least Mean Square

MLP: Multilayer Perceptron

MSE: Mean Square Error

NN: Neural Network

PCA: Principle Component Analysis

PIN: Password Identification Number

PR: Pattern Recognition

SOM: Self Organization Map

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

In the modern information age, human information is valuable. It can be used for the security and important social issues. Therefore, identification and authentication methods have developed into a main technology in various areas, such as entrance control in building and access control for computers.

Face recognition is a natural and straightforward biometric method that human beings use to identify each other. Humans are able to detect and identify faces in a scene with little or no effort. Face recognition has a high identification or recognition rate of greater than 90% for huge face databases with well-controlled pose and illumination conditions. This high rate can be used for replacement of lower security requirement environment and could be successfully employed in different applications.

Automatic face recognition is a vast and modern research area of computer vision, reaching from face detection, face localization, face tracking, extraction of face orientation and facial features and facial expressions.

The objectives of the work presented within this thesis are to develop an automatic face recognition system using feature-average based method. The face recognition system using back propagation neural networks implementation on multi-expression image sequence (natural, smiley, sad, and surprised). Instead of recognizing a face from a single view, a sequence of images showing face expressions is used as the face database. The developed method recognizes faces by using the essential face features (eyes, nose, and mouth) from different face expressions (natural, smiley, sad, and surprised). The averages of these features are then determined and represented as pattern vectors. These vectors will be used as the input to the neural network classifier (Back Propagation Algorithm) for training process.

This thesis is organized into four chapters. The first 3 chapters present background information on the face recognition, face recognition methods and artificial neural networks. The final chapter describes the developed automatic face recognition system.

Chapter 1 is an introduction to face recognition system. Biometrics technologies, pattern recognition, and face recognition and their applications are also presented in this chapter.

In Chapter 2 commonly used face recognition methods are presented. Approaches such as Principle Component Analysis (PCA), Linear Discriminant Analysis (LDA), Independent Component Analysis (ICA), Locally Linear Embedding (LLE), Hidden Markov Model (HMM) and Neural Networks.

Chapter 3 is based around the classifier (Back Propagation) that is used in this research. Background about neural networks and back propagation algorithm are discussed in this chapter.

Chapter 4 is presents the suggested face recognition system that is developed by the author. In this chapter all the phases of this face recognition system from capturing the image to classifying the face (recognized or unrecognized) with the computational and mathematical forms are discussed in detail.

CHAPTER ONE

FACE RECOGNITION OVERVIEW

1.1 Overview

Face recognition is important not only because it has a lot of potential applications in research fields such as Human Computer Interaction (HCI), biometrics and security, but also because it is a typical Pattern Recognition (PR) problem whose solution would help solving other classification problems.

This chapter provides background information about biometrics technologies, pattern recognition, and face recognition system and their applications.

1.2 Biometrics

Biometrics, the science of using individual personal characteristics to verify or recover identity, is set to become the successor to the Personal Identification Number (PIN). The term biometrics refers to a range of authentication systems. Biometrics is defined as the capture and use of physiological or behavioral characteristics for personal identification and / or individual verification purposes.

Biometrics definition is: a measurable physical characteristic or personal trait used to recognize the identity, or verify the claimed identity, of a person through automated means" [1]. Biometrics represent the most secure way to identify individuals because instead of verifying identity and granting access based on the possession or knowledge of cards, passwords, or tokens, verifying an identity is established (i.e. access is granted) using a physical and unique biometric characteristic.

Passwords or PINs used alone are responsible for fraud on corporate computer networks and the Internet because they can be guessed or stolen. Plastic cards, smart cards or computer token cards used alone are also not secure because they can be forged, stolen or lost, or become corrupted or unreadable. One can lose his card or forget a password, but he/she cannot lose or forget the fingers, eyes, or face.

The technique of using biometric methods for identification can be widely applied to forensics, automatic transfer machine (ATM) banking, communication, time and attendance, and access control.

Biometric technologies include:

- Face Recognition
- Finger Print Identification
- Hand Geometry Identification
- Iris Identification
- Voice Recognition
- Signature Recognition
- Retina Identification

Among these methods, there are multiple benefits to face recognition over other biometric methods. While the other biometrics requires some voluntary action, face recognition can be used passively. This has advantages for both ease of use and for covert use such as police surveillance. Face images also allow easy audits and verification performed by human operators when logging biometrics records. Regarding data acquisition, it is also easier to acquire good face images than good fingerprints. It turns out that about 5% of all people can not provide a good enough fingerprint for a reader to use for verification. The reasons include cut skin, bandaged finger, callused finger, dry skin, dry humidity, diseased skin, old skin, oriental skin, narrow finger and smudged sensor on reader. Similar disadvantage caused by the damage of epidermis tissue happens to hand geometry identification too. Using fingerprint scanners or palm readers can also transmit germs through a hand rest. In contrast, a face recognition system is totally hygienic and requires no maintenance because the face is measured from a distance.

Iris scans can provide very high accuracy rates for person identification. However, because the iris is so small, it takes two expensive camera motion drives with high resolution to find the iris. As the camera view has to be narrow to capture the resolution of the iris, the whole process is highly sensitive to body motion and as a consequence one has to be somewhat steady in order not to get rejected. Retina readers sense the retinal vein patterns in the back of ones eye. This requires an individual to look into an eyepiece while some light is being reflected off the back of the eye to capture the vein patterns. Although retina scanning yields very high accurate identification rates, most people would still resist having intrusive measurement inside their eyes. Both iris and retina scanning have lack in failing to identify people who wear vanity contact lens which cover the iris and retina or people blinking while their picture is being taken. Glare from glasses can also prevent the scanners from finding the iris or the retina. In contrast, an automated face recognition system only requires either one or two inexpensive cameras and the cameras do not need to move because they capture a large enough field of vision to cover the range of people's heights whether they are standing or sitting. A good face recognition algorithm works even with some glare reflected from the glasses or with the eyes closed.

Voice recognition for surveillance purposes suffer also as it is not reliable in noisy environments like public places or across phone lines with variable acoustic properties. The voice recognition systems are also sensitive to hoarse throat conditions when people are sick with colds. A tape recording of the correct person's voice can fool voice recognition systems that do not have a challenge-response process. The signature is used for legally binding documents, but it usually turns out that people vary their signatures greatly from time to time and from mood to mood. There are also concerns of pen and reading surfaces wearing poorly over time. This reduces the reliability of signature identification systems.

Face recognition is thus easier to be operated indoors and outdoors by detecting and cropping the area containing suspicious face pattern from complex background [2]. One can also consider the possibility to combine different biometric techniques with face recognition in order to build multi-modal person authentication systems.

1.3 Pattern Recognition

The study of how machines can observe the environment, learn to distinguish patterns of interest and make reasonable decisions about the categories of the patterns.

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:

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix}$$

where each element x_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.

Given a pattern; its recognition or classification may be supervised classification or unsupervised classification. Supervised pattern recognition is characterized by the fact that the correct classification of every training pattern is known. In the unsupervised case however, one is faced with the problem of actually learning the pattern classes present in the given data. This problem is also known as "learning without a teacher".

A pattern recognition system can be utilized in several different applications as image preprocessing/segmentation, computer vision, speech recognition, automated target recognition, optical character recognition, man and machine diagnostics, fingerprint identification, industrial inspection, financial forecast, and medical diagnosis. Also face recognition is a pattern recognition task performed specifically on faces.

A typical pattern recognition system block diagram is shown in Figure 1.1. A pattern recognition system contains sensor, preprocessing mechanism, feature extraction mechanism (manual or automated), classification or description algorithm, and set of examples (training set) already classified or described

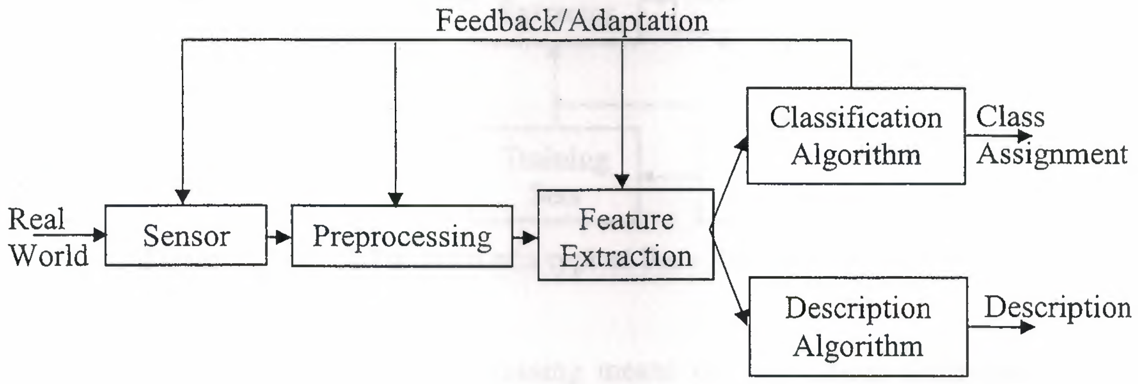


Figure 1.1 Block Diagram of a Pattern Recognition System

1.4 Face Recognition

Face recognition may seem an easy task for humans, and yet computerized face recognition system still can not achieve a completely reliable performance. The difficulties arise due to large variation in facial appearance, head size, orientation and change in environment conditions. Such difficulties make face recognition one of the fundamental problems in pattern analysis. In recent years there has been a growing interest in machine recognition of faces due to potential application.

To design a complete conventional human face recognition system should include three stages:

- Detection of an image pattern as a subject and then as a face against either uniform or complex background;
- Detection of facial landmarks for normalizing the face images to account for geometrical changes; and
- Identification and verification of face images using appropriate classification algorithms.

In Figure 1.2, the block diagram of a typical face recognition system is given.

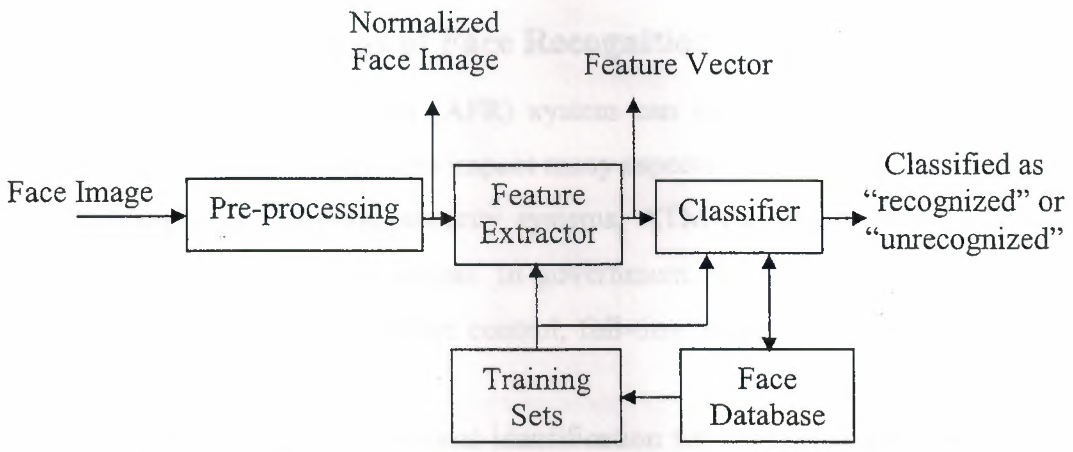


Figure 1.2 Block Diagram of a typical Face Recognition System

In the block diagram, pre-processing 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
- Histogram equalization, illumination normalization
- Median filtering
- High-pass filtering
- Background removal
- Translational and rotational normalizations

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.

Extracted features of the face image are compared with the ones stored in a face library (or face database). After doing this comparison, face image is classified as either recognized or unrecognized.

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.

After being classified as "unrecognized", 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 [3].

1.5 Real-Life Applications of Face Recognition

An Automated Face Recognition (AFR) system can be utilized in several different application domains. These domains impact many aspects of human life. In the industry, the AFR is applicable to photo-security systems, ATM banking building access, and telecommunication workstation access. In government, the AFR system can meet the needs in immigration control, border control, full-time monitoring, and airport/seaport security.

The AFR can improve criminal identification for forensic purpose and counter-terrorism techniques. This is of importance to the intelligence agencies and police departments. Defense requirements, such military troop entrance control, battlefield monitoring, and military personnel authentication, are applicable domains for this technique.

In medicine, the AFR can be useful in studies of the autonomic nervous system, the psychological reaction of patient, and intensive care monitoring by detecting and analyzing facial expressions.

Some of the applications areas of face recognition technologies have been listed in Table 1.1.

Table 1.1 Application of Face Recognition Technology [4]

Areas	**Specific Applications**
Biometrics	Driver's Licenses, Entitlement Programs Immigration, National ID, Passports, Voter Registration Welfare Fraud.
Information Security	Desktop Logon Application Security, Database Security, File Encryption Intranet Security, Internet Access, Medical Records Secure Trading Terminals
Law Enforcement and Surveillance	Advanced Video Surveillance, CCTV Control Portal Control, Post-Event Analysis Shoplifting and Suspect Tracking and Investigation
Access Control	Facility Access, Vehicular Access
Smart Cards	Stored Value Security, User Authentication
Entertainment	Video Game, Virtual Reality, Training Programs, Human-Robot-Interaction, Human-Computer-Interaction

1.6 Summary

This chapter described brief information as a background on biometric technologies, pattern recognition, and face recognition and their real life applications.

The commonly used face recognition methods will be presented in the next chapter.

CHAPTER TWO

FACE RECOGNITION METHODS

2.1 Overview

Face recognition is an example of advanced object recognition. The Face recognition is a widely explored field, and over the past 30 years, numerous algorithms have been proposed for face recognition.

This chapter presents, in detail, information about the commonly used face recognition methods that exist today such as Principle Component Analysis (PCA), Linear Discriminant Analysis (LDA), Independent Component Analysis (ICA), Locally Linear Embedding (LLE), Hidden Markov Model (HMM) and Neural Network (NN).

2.2 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) [5], also known as Karhunen-Loeve (KL) and Eigenspace projection for face recognition is based on the information theory approach. Principal component analysis is a dimensionality reduction technique which is used for compression and recognition problems. The scheme is based on an information theory approach that decomposes face images into a small set of characteristic feature images called eigenfaces, which may be thought of as the principal components of the initial training set of face images. Recognition is performed by projecting a new image onto the subspace spanned by the eigenfaces and then classifying the face by comparing its position in the face space with the positions of known individuals.

2.2.1 Eigenspace Projection

Eigenspace is calculated by identifying the eigenvectors of the covariance matrix derived from a set of training images. The eigenvectors corresponding to non-zero eigenvalues of the covariance matrix form an orthonormal basis that rotates and/or reflects the images in the N-dimensional space. Specifically, each image is stored in a vector of size N.

$$x^i = \left[x_1^i \;...... \; x_N^i\right]^T \qquad (2.1)$$

where x^i is raw training image.

The images are mean centered by subtracting the mean image from each image vector.

$$\bar{x}^i = x^i - m, \text{ where } m = \frac{1}{P}\sum_{i=1}^{P} x^i \qquad (2.2)$$

where $\bar{x}^i$, m and P are mean centered training image, mean image and number of training images respectively.

These vectors are combined, side-by-side, to create a data matrix of size NxP (where P is the number of images).

$$\bar{X} = [\ \bar{x}^1 \mid \bar{x}^2 \mid ..... \mid \bar{x}^P\ ] \qquad (2.3)$$

where $\bar{X}$ is data matrix of mean centered training images

The data matrix X is multiplied by its transpose to calculate the covariance matrix.

$$\Omega = \overline{XX}^T \qquad (2.4)$$

where Ω is covariance matrix

This covariance matrix has up to P eigenvectors associated with non-zero eigenvalues, assuming P<N. The eigenvectors are sorted, high to low, according to their associated eigenvalues. The eigenvector associated with the largest eigenvalue is the eigenvector that finds the greatest variance in the images. The eigenvector associated with the second largest eigenvalue is the eigenvector that finds the second most variance in the images. This trend continues until the smallest eigenvalue is associated with the eigenvector that finds the least variance in the images.

Identifying images through eigenspace projection takes three basic steps:

- The eigenspace must be created using training images.
- The training images are projected into the eigenspace.
- The test images are identified by projecting them into the eigenspace and comparing them to the projected training images.

2.2.1.1 Create Eigenspace

The following steps create an eigenspace:

- **Center data**: Each of the training images must be centered. Subtracting the mean image from each of the training images centers the training images as shown in equation (2.2). The mean image is a column vector such that each entry is the mean of all corresponding pixels of the training images.

- **Create data matrix**: Once the training images are centered, they are combined into a data matrix of size NxP, where P is the number of training images and each column is a single image as shown in equation (2.3).
- **Create covariance matrix**: The data matrix is multiplied by its transpose to create a covariance matrix as shown in equation (2.4). Covariance is also known as the angle measure. It calculates the angle between two normalized vectors. The covariance between images A and B is:

$$\operatorname{cov}(A,B) = \frac{A}{\|A\|} \bullet \frac{B}{\|B\|} \tag{2.5}$$

Covariance is a similarity measure. By negating the covariance value, it becomes a distance measure

- **Compute the eigenvalues and eigenvectors**: The eigenvalues and corresponding eigenvectors are computed for the covariance matrix.

$$\Omega V = \Lambda V \tag{2.6}$$

here V is the set of eigenvectors associated with the eigenvalues Λ.

- **Order eigenvectors:** Order the eigenvectors $v_i \in V$ according to their corresponding eigenvalues $\lambda_i \in \Lambda$ from high to low. Keep only the eigenvectors associated with non-zero eigenvalues. This matrix of eigenvectors is the eigenspace V, where each column of V is an eigenvector.

$$V = [\ v_1 \mid v_2 \mid \ldots\ldots \mid v_P\ ] \tag{2.7}$$

2.2.1.2 Project Training Images

Each of the centered training images $(\overline{x}^i)$ is projected into the eigenspace. To project an image into the eigenspace, calculate the dot product of the image with each of the ordered eigenvectors.

$$\widetilde{x}^i = V^T \overline{x}^i \tag{2.8}$$

where $\widetilde{x}^i$ is projected centered training image.

Therefore, the dot product of the image and the first eigenvector will be the first value in the new vector. The new vector of the projected image will contain as many values as eigenvectors.

2.2.1.3 Identify Test Images

Each test image is first mean centered by subtracting the mean image, and is then projected into the same eigenspace defined by V.

$$\bar{y}^i = y^i - m \text{, where } m = \frac{1}{P}\sum_{i=1}^{P} x^i \tag{2.9}$$

and

$$\tilde{y}^i = V^T \bar{y}^i \tag{2.10}$$

where y^i, $\bar{y}^i$, and $\tilde{y}^i$ are raw test image, mean centered test image, and projected centered test image respectively.

The projected test image is compared to every projected training image and the training image that is found to be closest to the test image is used to identify the training image. The images can be compared using any number of similarity measures; the most common is the L_2 norm.

L_2 norm: The L_2 norm is also known as the Euclidean norm or the Euclidean distance when its square root is calculated. The L_2 norm of an image A and an image B is:

$$L_2(A,B) = \sum_{i=1}^{N}(A_i - B_i)^2 \tag{2.11}$$

2.3 Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis [6] is a dimensionality reduction technique which is used for classification problems. LDA is also known as Fisher's Discriminant Analysis and it searches for those vectors in the underlying space that best discriminate among classes.

Linear Discriminant Analysis creates a linear combination of independent features which yields the largest mean differences between the desired classes. The basic idea of LDA is to find a linear transformation such that feature clusters are most separable after the transformation which can be achieved through scatter matrix analysis.

The goal of LDA is to maximize the between-class scatter matrix measure while minimizing the within-class scatter matrix measure. LDA groups images of the same class and separates images of different classes. Images are projected from N-dimensional space (where N is the number of pixels in the image) to C-1 dimensional space (where C is the number of classes of images). To identify a test image, the projected test image is compared to each projected training image, and the test image is identified as the closest training image.

The training images are projected into a subspace. The test images are projected into the same subspace and identified using a similarity measure. Following are the steps to follow to find the LDA of a set of images by first projecting the images into any orthonormal basis.

- **Compute means:** Compute the mean of the images in each class (m_i) and the total mean of all images (m).
- **Center the images in each class:** Subtract the mean of each class from the images in that class.

  $$\forall x \in X_i,\ X_i \in X,\ \acute{x} = x - m_i \tag{2.12}$$
- **Center the class means:** Subtract the total mean from the class means.

  $$\acute{m} = m_i - m \tag{2.13}$$
- **Create a data matrix:** Combine the all images, side-by-side, into one data matrix.
- **Find an orthonormal basis for this data matrix:** This can be accomplished by using a Orthogonal-triangular decomposition or by calculating the full set of eigenvectors of the covariance matrix of the training data. Let the orthonormal basis be U.
- **Project all centered images into the orthonormal basis:** Create vectors that are the dot product of the image and the vectors in the orthonormal basis.

  $$\widetilde{x} = U^T \acute{x} \tag{2.14}$$
- **Project the centered means into the orthonormal basis:**

  $$\widetilde{m} = U^T \acute{m}_i \tag{2.15}$$
- **Calculate the within class scatter matrix:** The within class scatter matrix measures the amount of scatter between items within the same class. For the i^{th}

class a scatter matrix (S_i) is calculated as the sum of the covariance matrices of the projected centered images for that class.

$$S_i = \sum_{x \in X_i} \tilde{x}\tilde{x}^T \tag{2.16}$$

The within class scatter matrix (S_W) is the sum of all the scatter matrices.

$$S_W = \sum_{i=1}^{C} S_i \tag{2.17}$$

where C is the number of classes.

- **Calculate the between class scatter matrix:** The between class scatter matrix S_B measures the amount scatter between classes. It is calculated as the sum of the covariance matrices of the projected centered means of the classes, weighted by the number of images in each class.

$$S_B = \sum_{i=1}^{C} n_i \tilde{m}_i \tilde{m}_i^T \tag{2.18}$$

where n_i is the number of images in the class.

- **Solve the generalized eigenvalue problem:** Solve for the generalized eigenvectors (V) and eigenvalues (Λ) of the within class and between class scatter matrices.

$$S_B V = \lambda S_W V \tag{2.19}$$

- **Keep the first $C-1$ eigenvectors:** Sort the eigenvectors by their associated eigenvalues from high to low and keep the first $C-1$ eigenvectors. These are the Fisher basis vectors.
- **Project images onto eigenvectors:** Project all the rotated original (i.e. Not centered) images onto the Fisher basis vectors. First project the original images into the orthonormal basis, and then project these projected images onto the Fisher basis vectors. The original rotated images are projected onto this line because these are the points that the line has been created to discriminate, not the centered images.

2.4 Independent Component Analysis (ICA)

Independent Component Analysis (ICA) [7] is a statistical method for transforming an observed multidimensional random vector into its components that are statistically as independent from each other as possible. ICA is a special case of redundancy reduction technique and it represents the data in terms of statistically independent variables. ICA of a random vector consists of searching for a linear transformation that minimizes the statistical dependence between its components.

The goal of ICA is to provide an independent image decomposition and representation. In other words, the goal is to minimize the statistical dependence between the basis vectors.

ICA is a generalization of PCA in the sense that ICA de-correlates the high-order moments of input while PCA encodes the second-order moments only. In the task of face recognition, ICA can be superior to PCA owing to its ability to represent the high-order statistics of face images. While the reconstructed face images with a few leading eigenfaces lose the details and look like low pass filtered versions, the corresponding residue images contain high-frequency components and are less sensitive to illumination variation. Since these residue images still contain rich information for the individual identities, face features are extracted from these residue faces by ICA.

The basic steps to derive the independent component analysis method are as follows:

- Collect x_i of an n dimensional data set x, i=1,2, …, m
- Mean correct all the points: Calculate mean m_x and subtract it from each data point, $x_i - m_x$
- Calculate the covariance matrix:

$$C = (x_i - m_x)(x_i - m_x)^T \tag{2.20}$$

- The ICA of x factorizes the covariance matrix C into the following form: $C = F\Delta F^T$ where Δ is a diagonal real positive matrix.
- F transforms the original data X into Z such that the components of the new data Z are independent: X = F Z. Derive the ICA transformation F by using the algorithm which consists of three operations: whitening, rotation, and normalization.

- Compare the test image's independent components with the independent components of each training image by using a similarity measure. The result is the training image which is the closest to the test image.

2.5 Locally Linear Embedding (LLE)

Dimensionality reduction is an important and necessary preprocessing of multidimensional data, such as face images. The purposes of reducing dimensionality of observation data are to compress the data to reduce storage requirements, to eliminate noise, to extract features from data for recognition, and to project data to a lower dimensional space.

For face images, recently developed Locally Linear Embedding (LLE) [8] method is used as nonlinear dimensionality reduction. The locally linear embedding (LLE) algorithm's attractive properties are:

- Only two parameters to be set
- Optimizations not involving local minima,
- Preservation of local geometry of high dimensional data in the embedded space
- A single global coordinate system of the embedded space.

The LLE algorithm is given as follows:

In LLE, the basic assumption is that the neighborhood of a given data point is locally linear. In other words, a data-point can be reconstructed as a linear combination of its neighboring points. When projecting to a low dimensional subspace, LLE preserves this locally linear structure. To enforce the linear structure, the following reconstruction error is defined:

$$e(W) = \sum_i \left| \vec{X}_i - \sum_{j \in N(i)} W_{ij} \vec{X}_j \right|^2 \qquad (2.21)$$

In this equation, $\vec{X}_i$ are the original data points and W_{ij} are the weights used for reconstruction. In contrast to a linear method, the second sum is only over the neighbors of point i, denoted N(i). From the local reconstruction in the high dimensional space, one can define a similar reconstruction error in the low dimensional space:

$$\Phi(Y) = \sum_i \left| \vec{Y}_i - \sum_{j \in N(i)} W_{ij} \vec{Y}_j \right|^2 \qquad (2.22)$$

Putting everything together, this creates the following embedding procedure:

- First, build the neighborhood map N(i).

- Second, using the neighborhood map determines the reconstruction weights W_{ij}
- Finally, using the neighborhood map and the reconstruction weights, determine the embedded values $\vec{Y}_i$.

Several key points are worth mentioning. The construction of the neighborhood map is done under the constraint $\sum_j W_{ij} = 1$. Also, a regularization term is typically used when calculating the weight matrix to prevent numerical errors and to allow embedding when the number of neighbors exceeds the number of dimensions. Finally, this entire method is computationally feasible and involves solving some linear equations and an eigenvalue problem.

2.6 Hidden Markov Models (HMM)

Hidden Markov Models (HMM) [9] have been successfully used for speech recognition and more recently in action recognition where data is essentially one dimensional over time. In order to use HMM for recognition, an observation sequence is obtained from the test signal and then the likelihood of each HMM generating this signal is computed. The HMM which has the highest likelihood then identifies the test signal. Finding the state sequence which maximizes the probability of an observation is done using the Viterbi algorithm, which is a simple dynamic programming optimization procedure.

2.6.1 One-Dimensional HMM

HMM has been extensively used for speech recognition, where data is naturally one-dimensional along the time axis. The equivalent fully-connected two-dimensional HMM would lead to a very high computational cost problem.

Samaria has proposed using the 1D continuous HMM for face recognition [10]. For a frontal face the states of the Markov model include forehead, eyes, nose, mouth, and chin, each representing a state. These states always occur in the same order, from top to bottom, even if faces undergo small rotations in the image plane. Each facial region will be assigned to a state, in a left-to-right one dimensional hidden Markov model (Figure 2.1). Only transitions between adjacent states in a top to bottom manner are allowed.

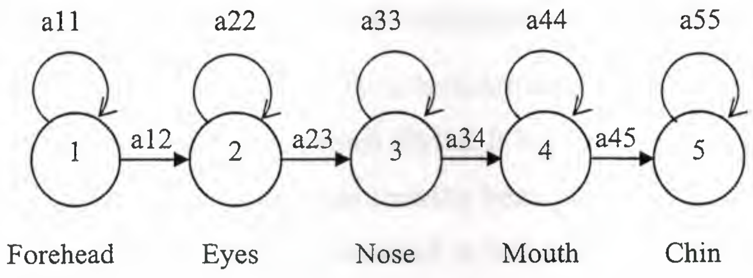


Figure 2.1 Left-to-Right States of a One-Dimensional HMM [11]

An observation sequence is generated from a face image (XxY) using a sampling window (MxL) with overlap (Figure 2.2). The observation sequence is composed of vectors that represent the consecutive horizontal strips, where each vector contains the pixel values from the associated strips. The goal of the training stage is to optimize the hidden Markov model parameters to best describe the observations. This is done by maximizing the probability of the observed sequence given a set of variable parameters. Recognition is done by matching the test image against each of the trained models. To do this the image is converted to an observation sequence and then model likelihoods for all database images are computed. The model with the highest likelihood reveals the identity of the unknown face.

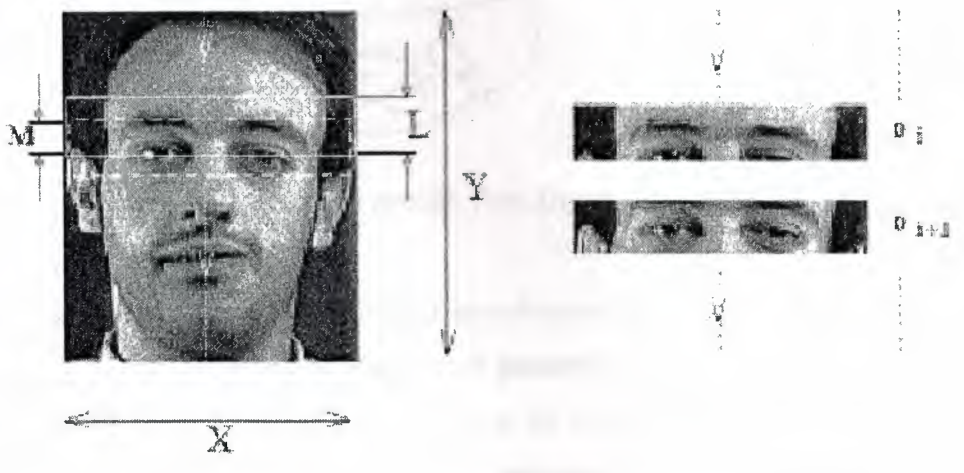


Figure 2.2 Image Sampling Technique for One-Dimensional HMM [11]

2.6.2 Pseudo and Embedded Two-Dimensional HMM

A more flexible HMM, that allows for shifts in both horizontal and vertical directions, is obtained by using a pseudo two-dimensional HMM. It has been designed specifically to deal with two-dimensional signals and has recently been proposed for face recognition applications. The structure is not fully connected in two-dimensions, hence it is pseudo two-dimensional. States are linked as in a one-dimensional HMM to form vertical superstates. Each superstate in the one-dimensional HMM is represented by an embedded one-dimensional HMM (Figure 2.3).

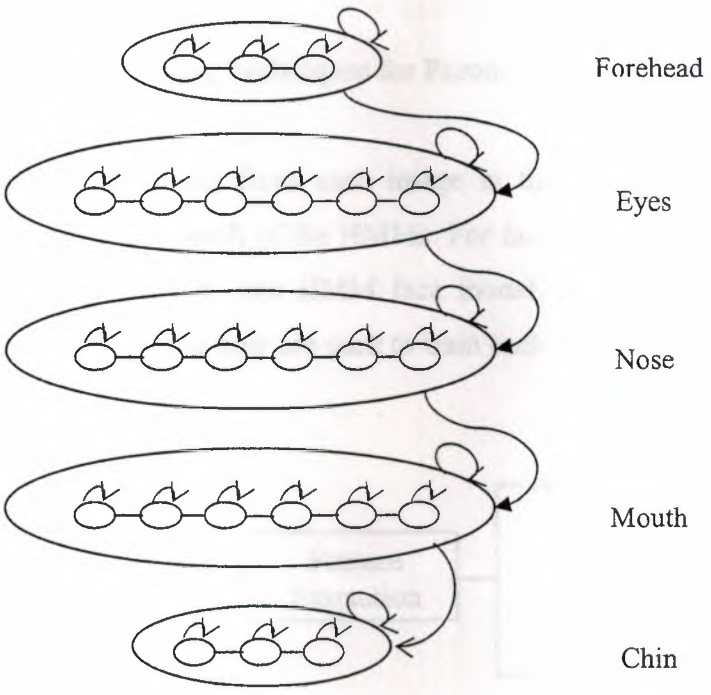


Figure 2.3 States of a pseudo Two-Dimensional HMM [11]

Samaria introduced an equivalent one-dimensional HMM and used it for face recognition [10]. The observation sequence is generated by letting a window (PxL) scan the image (XxY) from left to right, and top to bottom (Figure 2.4). Each sample overlaps other samples both in horizontal (P) and vertical (M) direction. The intensities of the pixels inside each block were used as observation vectors.

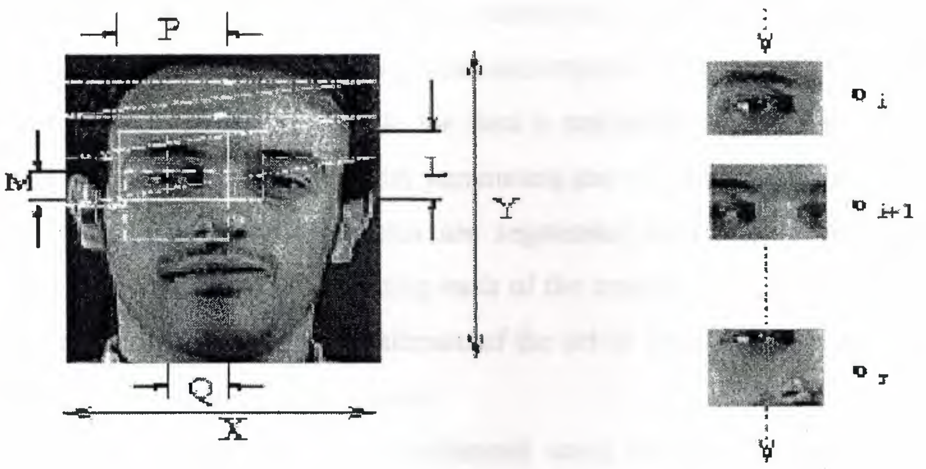


Figure 2.4 Image Sampling Techniques for Pseudo Two-Dimensional HMM [11]

After extracting blocks from each image in the training set, the observation vectors are obtained to train each of the HMMs. For face recognition each individual in the database is represented by one HMM face model. A set of images representing different instances of the same face are used to train each HMM.

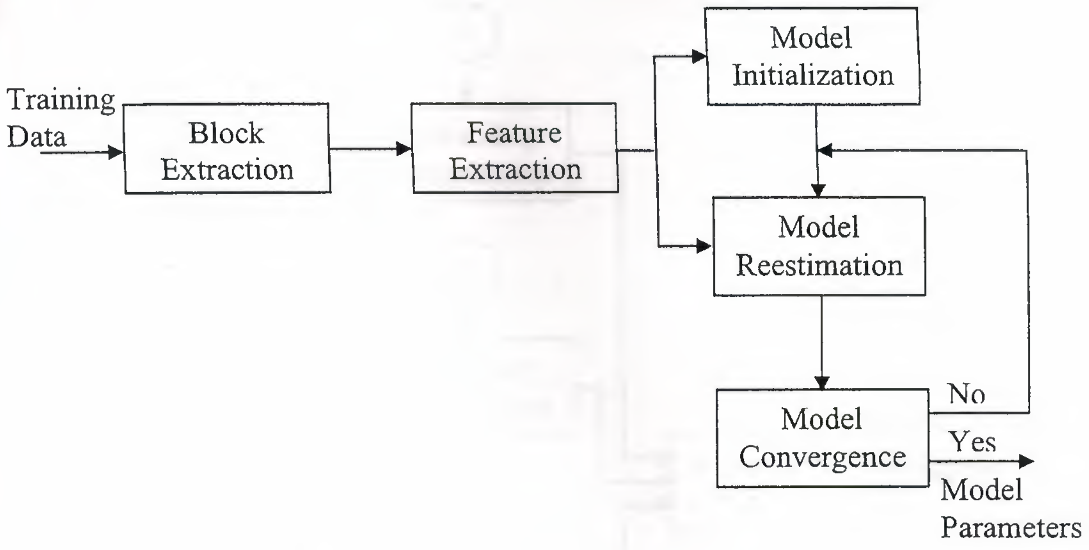


Figure 2.5 HMM Training Scheme [11]

The general HMM training scheme (Figure 2.5) is a variant of the K-means iterative procedure for clustering data. First the initial parameter values are computed iteratively using the training data and the prototype model. The goal of this stage is to find a good estimate for the observation probability. Good initial estimates of the parameters are essential for rapid and proper convergence to the global maximum of the likelihood function. On the first cycle the data is uniformly segmented, matched with each model state and the initial model parameters are extracted. On successive cycles the set of training observation cycles are segmented into states using the Viterbi algorithm [12]. The result of segmenting each of the training sequences, for each of the N states, is a maximum likelihood estimate of the set of observations that occur within each state according to the current model.

The model parameters are re-estimated using the Baum-Welch re-estimation procedure [13]. This procedure adjusts the model parameters so as to maximize the probability of observing the training data, given each corresponding model. The resulting model is then compared to the previous model by computing a distance score that reflects the statistical similarity of the HMM. If the model distance score exceeds a threshold then the old model is replaced by the new model and the training loop is repeated. If the model distance score falls below the threshold, then model convergence is assumed and the final parameters are saved. HMM recognition block diagram has been shown in Figure 2.6.

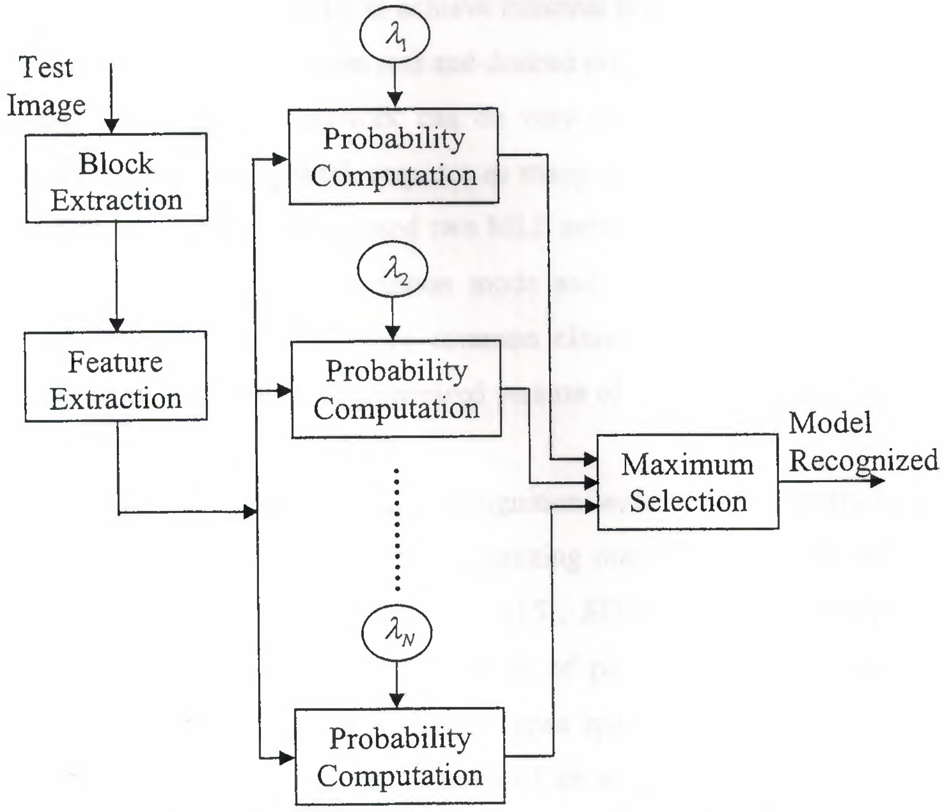


Figure 2.6 HMM Recognition Block Diagram [11]

The face recognition begins by looking within each rectangular window in the test image, extracting observation vectors. After extracting the observation vectors as in the training phase, the probability of the given observation sequence given each face model is computed using a simple Viterbi recognizer. The model with the highest likelihood is selected and this model reveals the identity of the unknown face.

2.7 Neural Networks

Recognition of visual objects is performed effortlessly in our everyday life by humans. A previously seen face is easily recognized regardless of various transformations like change in size and position. It is known that humans process a natural image in less than 150 ms. The brain thus performs these tasks at very high speed. Neural networks are attempts to create face recognition systems that are based on the way humans detect and recognize faces.

The multilayer perceptron (MLP) neural network is a good tool for classification purposes. It can approximate almost any regularity between its input and its output. The weights are adjusted by supervised training procedure called back-propagation (BP). Back-propagation is a kind of gradient descent method, which searches for an acceptable local minimum in order to achieve minimal error. Error is defined as the root mean square of differences between real and desired outputs from the neural network.

Often even a simple network can be very complex and difficult to train. A typical image recognition network requires as many input nodes as there are pixels in the image. Cottrell and Flemming used two MLP networks working together [14]. The first one operates in an auto-association mode and extracts features for the second network, which operates in the more common classification mode. In this way the hidden layer output constitutes a compressed version of the input image and can be used as input to the classification network.

One of the more successful face recognition with neural networks is a result of combining local image sampling, a self organizing map (SOM) neural network and a convolutional neural network (Figure 2.7) [15]. SOM is an unsupervised learning process which learns the distribution of a set of patterns without having any class information. A pattern is projected from an input space to a position in the map and information is thereby coded as the location of an activated node. Unlike most other classification or clustering techniques SOM preserves the topological ordering of classes. This feature makes it useful in classification of data which includes a large number of classes.

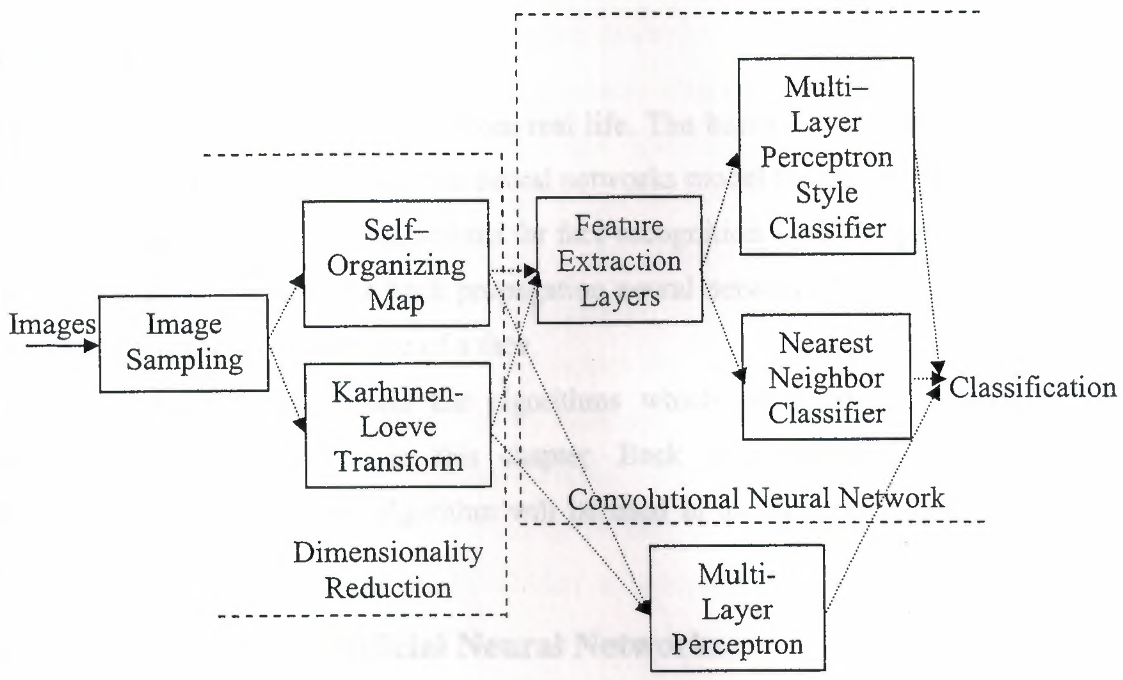


Figure 2.7 Neural Network Face Recognition System [15]

2.8 Summary

This chapter presented known face recognition methods such as Principle Component Analysis (PCA), Linear Disciminant Analysis (LDA), Independent Component Analysis (ICA), Locally Linear Embedding (LLE), Hidden Markov Model (HMM) and Neural Network (NN).

The next chapter will present in detail neural networks as classifiers. Neural networks will be used as part of the face recognition system that is developed in this thesis.

CHAPTER THREE
ARTIFICIAL NEURAL NETWORKS

3.1 Overview

The idea of face recognition comes from real life. The human brain can memorize and recognize the face of any person. The neural networks model the human brain.

Neural network (NN) algorithms for face recognition work by applying the input face after preprocessing to the back propagation neural network. The network is trained to output the presence or absence of a face.

The basic concepts and the algorithms which are used in artificial neural networks will be presented in this chapter. Back propagation algorithm will be explained in detail since the algorithm will be used in the developed face recognition system.

3.2 Introduction to Artificial Neural Networks

An artificial neural network (ANN) is a system composed of many simple processing elements operating in parallel whose function is determined by network structure, connection strengths, and the processing performed at computing element or nodes. Neural network architecture is inspired by the architecture of biological nervous systems, which use many simple processing elements operating in parallel to obtain high computation rates.

An artificial neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:

- Knowledge is acquired by the network through a learning process.
- Interneuron connection strengths known as synaptic weights are used to store the knowledge [16].

The neuron is a “many inputs one output” unit. The output can be excited or not excited, just two possible choices. The signals from other neurons are summed together and compared against a threshold to determine if the neuron shall excite. The input signals are subject to attenuation in the synapses which are junction parts of the neuron.

ANN draws much of their inspiration from the biological nervous system. It is therefore very useful to have some knowledge of the way this system is organized. Most living creatures, which have the ability to adapt to a changing environment, need a controlling unit which is able to learn. Higher developed animals and humans use very complex networks of highly specialized neurons to perform this task. The control unit - or brain - can be divided in different anatomic and functional sub-units, each having certain tasks like vision, hearing, motor and sensor control.

The brain is connected by nerves to the sensors and actors in the rest of the body. The brain consists of a very large number of neurons, about 10^{11} in average. These can be seen as the basic building bricks for the central nervous system. The neurons are interconnected at points called synapses. The complexity of the brain is due to the massive number of highly interconnected simple units working in parallel, with an individual neuron receiving input from up to 10000 others [17].

Structurally the neuron can be divided in three major parts: the cell body (soma), the dendrites, and the axon. The cell body contains the organelles of the neuron and also the `dendrites' are originating there. These are thin and widely branching fibers, reaching out in different directions to make connections to a larger number of cells within the cluster. Input connections are made from the axons of other cells to the dendrites or directly to the body of the cell. These are known as axondentrititic and axonsomatic synapses.

There is only one axon per neuron. It is a single and long fiber, which transports the output signal of the cell as electrical impulses (action potential) along its length. The end of the axon may divide in many branches, which are then connected to other cells. The branches have the function to fan out the signal to many other inputs [18],[19].

A single-input neuron artificial network is shown in Figure 3.1. The scalar input p is multiplied by the scalar weight w to form wp, one of the terms that is sent to the summer. The other input, 1, is multiplied by a bias b and then passed to the summer. The summer output net, often referred to network input, goes into an activation function *f*, which produces the scalar neuron output. This is the simplest form of the artificial neuron and is known as a perceptron.

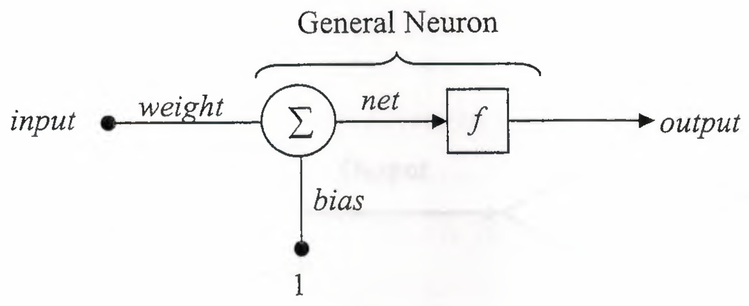


Figure 3.1 Single - Input Artificial Neuron

The neuron output is calculated by equation 3.1:

$$output = f(wp + b) \tag{3.1}$$

The simple model for artificial neuron in the Figure 3.1 can indicate the same way of the biological neuron. The weight w corresponds to the strength of a synapse, the cell body is represented by the summation and the activation function, and the neuron output represents the signal in the axon.

3.3 Teaching an Artificial Neural Network

Artificial neural networks learning algorithms can be divided into two main groups that are supervised (Associative learning) and unsupervised (Self-Organization)

3.3.1 Supervised Learning

The vast majority of artificial neural network solutions have been trained with supervision. In this mode, the actual output of a neural network is compared to the desired output. Weights, which are usually randomly set to begin with, are then adjusted by the network so that the next iteration, or cycle, will produce a closer match between the desired and the actual output. The learning method tries to minimize the current errors of all processing elements. This global error reduction is created over time by continuously modifying the input weights until acceptable network accuracy is reached.

The supervised artificial neural network needs teacher to describe what the network should have given as response. The difference between target (desired output) *and the actual output, the error is determined and back propagated through the network* to adjust the network. The basic architecture of supervised artificial neural network is shown in Figure 3.2.

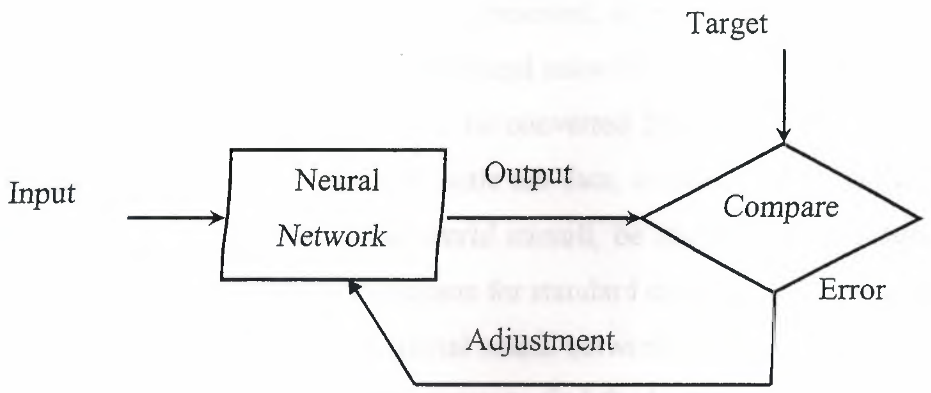


Figure 3.2 *Architecture of Supervised Artificial Neural Network*

With supervised learning, the artificial neural network must be trained before it becomes useful. Training consists of presenting input and output data to the network. This data is often referred to as the training set. That is, for each input set provided to the system, the corresponding desired output set is provided as well. In most applications, actual data must be used. This training phase can consume a lot of time. In prototype systems, with inadequate processing power, learning can take weeks. This training is considered complete when the neural network reaches a user defined performance level. This level signifies that the network has achieved the desired statistical accuracy as it produces the required outputs for a given sequence of inputs. When no further learning is necessary, the weights are typically frozen for the application. Some network types allow continual training, at a much slower rate, while in operation. This helps a network to adapt to gradually changing conditions.

Training sets need to be fairly large to contain all the needed information if the network is to learn the features and relationships that are important. Not only do the sets have to be large but the training sessions must include a wide variety of data. If the network is trained just one example at a time, all the weights set so meticulously for one fact could be drastically altered in learning the next fact. The previous facts could be forgotten in learning something new. As a result, the system has to learn everything together, finding the best weight settings for the total set of facts. For example, in teaching a system to recognize pixel patterns for the ten digits, if there were twenty examples of each digit, all the examples of the digit seven should not be presented at the same time.

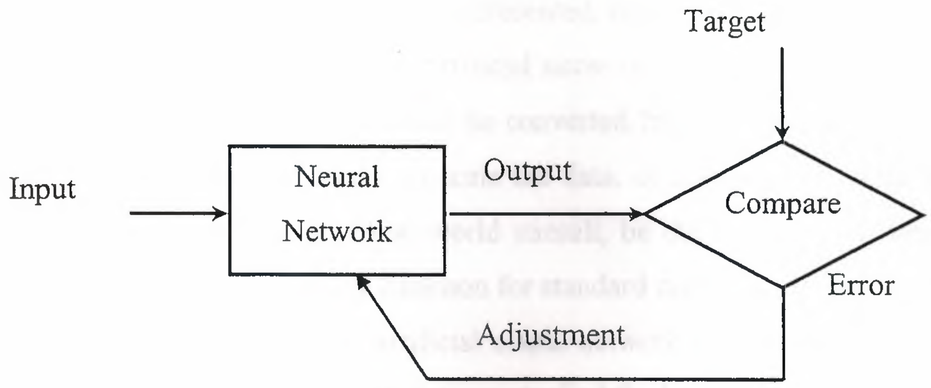


Figure 3.2 Architecture of Supervised Artificial Neural Network

With supervised learning, the artificial neural network must be trained before it becomes useful. Training consists of presenting input and output data to the network. This data is often referred to as the training set. That is, for each input set provided to the system, the corresponding desired output set is provided as well. In most applications, actual data must be used. This training phase can consume a lot of time. In prototype systems, with inadequate processing power, learning can take weeks. This training is considered complete when the neural network reaches a user defined performance level. This level signifies that the network has achieved the desired statistical accuracy as it produces the required outputs for a given sequence of inputs. When no further learning is necessary, the weights are typically frozen for the application. Some network types allow continual training, at a much slower rate, while in operation. This helps a network to adapt to gradually changing conditions.

Training sets need to be fairly large to contain all the needed information if the network is to learn the features and relationships that are important. Not only do the sets have to be large but the training sessions must include a wide variety of data. If the network is trained just one example at a time, all the weights set so meticulously for one fact could be drastically altered in learning the next fact. The previous facts could be forgotten in learning something new. As a result, the system has to learn everything together, finding the best weight settings for the total set of facts. For example, in teaching a system to recognize pixel patterns for the ten digits, if there were twenty examples of each digit, all the examples of the digit seven should not be presented at the same time.

How the input and output data is represented, or encoded, is a major component to successfully instructing a network. Artificial networks only deal with numeric input data. Therefore, the raw data must often be converted from the external environment. Additionally, it is usually necessary to scale the data, or normalize it to the network's paradigm. This pre-processing of real-world stimuli, be they cameras or sensors, into machine readable format is already common for standard computers. Many conditioning techniques which directly apply to artificial neural network implementations are readily available. It is then up to the network designer to find the best data format and matching network architecture for a given application.

After a supervised network performs well on the training data, then it is important to see what it can do with data it has not seen before. If a system does not give reasonable outputs for this test set, the training period is not over. Indeed, this testing is critical to insure that the network has not simply memorized a given set of data but has learned the general patterns involved within an application.

One of the most commonly used supervised neural network model is back propagation network that uses back propagation learning algorithm. Back propagation algorithm is one of the well-known algorithms in neural networks [20].

3.3.2 Unsupervised Learning

Unsupervised learning is the great promise of the future. Currently, this learning method is limited to networks known as self-organizing maps. These kinds of networks are not in widespread use. They are basically an academic novelty. Yet, they have shown they can provide a solution in a few instances, proving that their promise is not groundless. They have been proven to be more effective than many algorithmic techniques for numerical aerodynamic flow calculations. They are also being used in the lab where they are split into a front-end network that recognizes short, phoneme-like fragments of speech which are then passed on to a back-end network. The second artificial network recognizes these strings of fragments as words.

For an unsupervised learning rule, the training set consists of input training patterns only. Therefore, the network is trained without benefit of any teacher. The network learns to adapt based on the experiences collected through the previous training patterns. The basic architecture of an unsupervised system is shown in Figure 3.3.

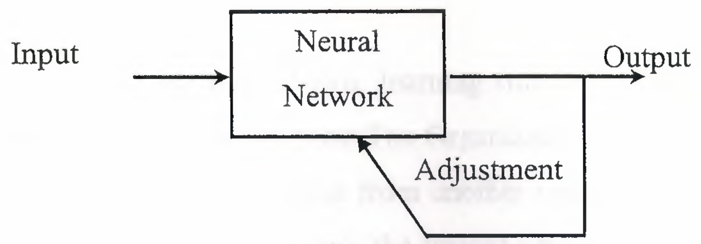


Figure 3.3 Architecture of Unsupervised Artificial Neural Network

This promising field of unsupervised learning is sometimes called self-supervised learning. These networks use no external influences to adjust their weights. Instead, they internally monitor their performance. These networks look for regularities or trends in the input signals, and makes adaptations according to the function of the network. Even without being told whether it's right or wrong, the network still must have some information about how to organize itself. This information is built into the network topology and learning rules. An unsupervised learning algorithm might emphasize cooperation among clusters of processing elements. In such a scheme, the clusters would work together. If some external input activated any node in the cluster, the cluster's activity as a whole could be increased. Likewise, if external input to nodes in the cluster was decreased, that could have an inhibitory effect on the entire cluster.

Competition between processing elements could also form a basis for learning. Training of competitive clusters could amplify the responses of specific groups to specific stimuli. As such, it would associate those groups with each other and with a specific appropriate response. Normally, when competition for learning is in effect, only the weights belonging to the winning processing element will be updated [21].

3.3.3 Learning Laws

Many learning laws are in common use. Most of these laws are some sort of variation of the best known and oldest learning law, Hebb's Rule. Research into different learning functions continues as new ideas routinely show up in trade publications. Some researchers have the modeling of biological learning as their main objective. Others are experimenting with adaptations of their perceptions of how nature handles learning. Either way, man's understanding of how neural processing actually works is very limited. Learning is certainly more complex than the simplifications represented by the learning laws currently developed. A few of the major laws are presented as examples.

Hebb's Rule:

The first, and undoubtedly the best known, learning rule were introduced by Donald Hebb. The description appeared in his book The Organization of Behavior in 1949. His basic rule is: If a neuron receives an input from another neuron and if both are highly active (mathematically have the same sign), the weight between the neurons should be strengthened [22].

Hopfield Law:

It is similar to Hebb's rule with the exception that it specifies the magnitude of the strengthening or weakening. It states, "if the desired output and the input are both active or both inactive, increment the connection weight by the learning rate, otherwise decrement the weight by the learning rate [23].

The Delta Rule:

This rule is a further variation of Hebb's Rule. It is one of the most commonly used. This rule is based on the simple idea of continuously modifying the strengths of the input connections to reduce the difference (the delta) between the desired output value and the actual output of a processing element. This rule changes the synaptic weights in the way that minimizes the mean squared error of the network. This rule is also referred to as the Widrow-Hoff Learning Rule and the Least Mean Square (LMS) Learning Rule [24]. The way that the Delta Rule works is that the delta error in the output layer is transformed by the derivative of the transfer function and is then used in the previous neural layer to adjust input connection weights. In other words, this error is back-propagated into previous layers one layer at a time. The process of back-propagating the network errors continues until the first layer is reached. The network type called Feedforward, Back-propagation derives its name from this method of computing the error term. When using the delta rule, it is important to ensure that the input data set is well randomized. Well ordered or structured presentation of the training set can lead to a network which can not converge to the desired accuracy. If that happens, then the network is incapable of learning the problem.

The Gradient Descent Rule:

This rule is similar to the Delta Rule in that the derivative of the transfer function is still used to modify the delta error before it is applied to the connection weights. Here, however, an additional proportional constant tied to the learning rate is appended to the final modifying factor acting upon the weight. This rule is commonly used, even though it converges to a point of stability very slowly. It has been shown that different learning rates for different layers of a network help the learning process converge faster. In these tests, the learning rates for those layers close to the output were set lower than those layers near the input [25].

Kohonen's Learning Law:

This procedure, developed by Teuvo Kohonen, was inspired by learning in biological systems. In this procedure, the processing elements compete for the opportunity to learn, or update their weights. The processing element with the largest output is declared the winner and has the capability of inhibiting its competitors as well as exciting its neighbors. Only the winner is permitted an output, and only the winner plus its neighbors are allowed to adjust their connection weights.

Further, the size of the neighborhood can vary during the training period. The usual paradigm is to start with a larger definition of the neighborhood, and narrow in as the training process proceeds. Because the winning element is defined as the one that has the closest match to the input pattern, Kohonen networks model the distribution of the inputs. This is good for statistical or topological modeling of the data and is sometimes referred to as self-organizing maps or self-organizing topologies [26].

3.4 Multilayer Perceptron

The multilayer perceptron (MLP) is a hierarchical structure of several perceptrons. A single perceptron is not very useful because of its limited mapping ability. No matter what activation function is used, the perceptron is only able to represent an oriented ridge-like function. The perceptrons can, however, be used as building blocks of a larger, much more practical structure. A typical multilayer perceptron (MLP) network consists of a set of source nodes forming the input layer, one or more hidden layers of computation nodes, and an output layer of nodes. The input signal propagates through the network layer-by-layer. The signal-flow of such a network with one hidden layer is shown in Figure 3.4 [16].

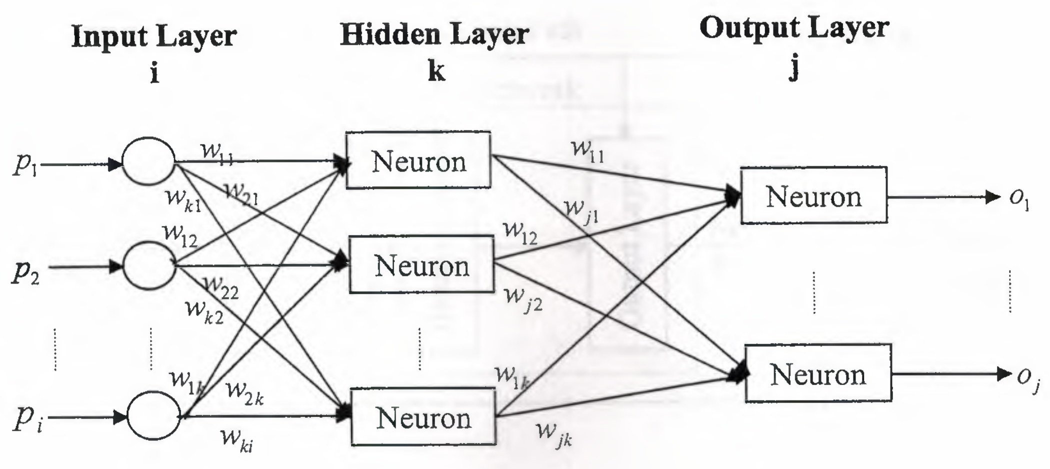


Figure 3.4 Architecture of Multilayer Perceptron

The supervised learning problem of the multilayer perceptron can be solved with the back-propagation algorithm. The algorithm consists of two steps. In the forward pass, the predicted outputs corresponding to the given inputs. In the backward pass, partial derivatives of the cost function with respect to the different parameters are propagated back through the network. The chain rule of differentiation gives very similar computational rules for the backward pass as the ones in the forward pass. The network weights can then be adapted using any gradient-based optimization algorithm. The whole process is iterated until the weights have converged [16]. The multilayer perceptron network can also be used for unsupervised learning by using the so called auto-associative structure. This is done by setting the same values for both the inputs and the outputs of the network. The extracted sources emerge from the values of the hidden neurons. This approach is computationally rather intensive. The multilayer perceptron network has to have at least three hidden layers for any reasonable representation and training such a network is a time consuming process.

3.5 Back Propagation Neural Network

In the artificial neural networks, there are several network architectures and training algorithms are available, the back-propagation algorithm is most popular algorithm. Back propagation neural network architecture is very popular because it can be applied to many different tasks.

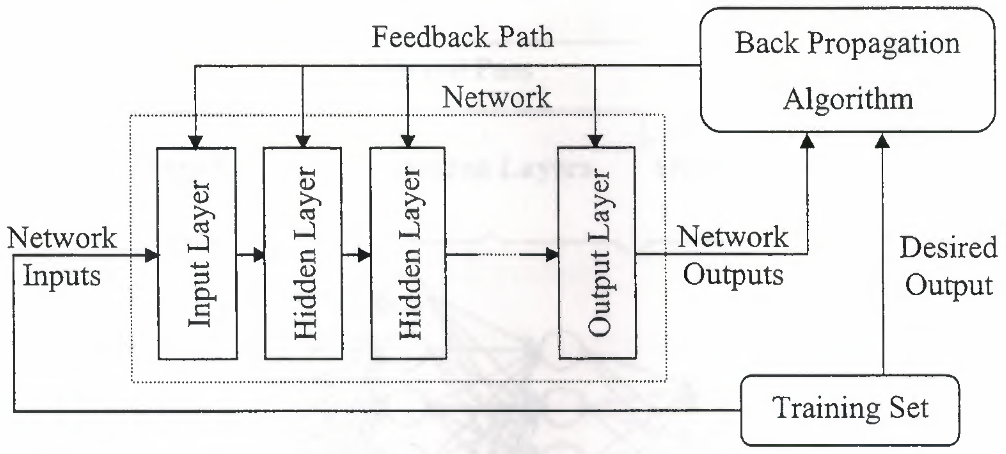


Figure 3.5 Block Diagram of Back Propagation Network

In Figure 3.5 the block diagram of back propagation network is shown. The back-propagation algorithm is a supervised learning algorithm for artificial neural networks. It extends the weight update rule used in the simple perceptron learning algorithm to multilayer feed forward artificial neural network.

The name "back-propagation" derives from the manner in which information is propagated across the network on each pass of the algorithm, the errors at the output nodes are passed back to the hidden nodes using the network connections, and the resulting information is used to update the connection weights.

3.5.1 Structure of Back propagation Network

Feed-forward neural networks trained by back propagation consist of several layers of simple processing elements called neurons, interconnections, and weights that are assigned to those interconnections. Each neuron contains the weighted sum of its inputs filtered by a sigmoid transfer function. The neurons are interconnected in such a way that information relevant to the I/O mapping is stored in the weights. The various layers of neurons in back propagation networks receive, process, and transmit information on the relationships between the input parameters and corresponding responses. Aside from the input and output layers, these networks incorporate one or more "hidden". Architecture of back propagation network is shown in Figure 3.6.

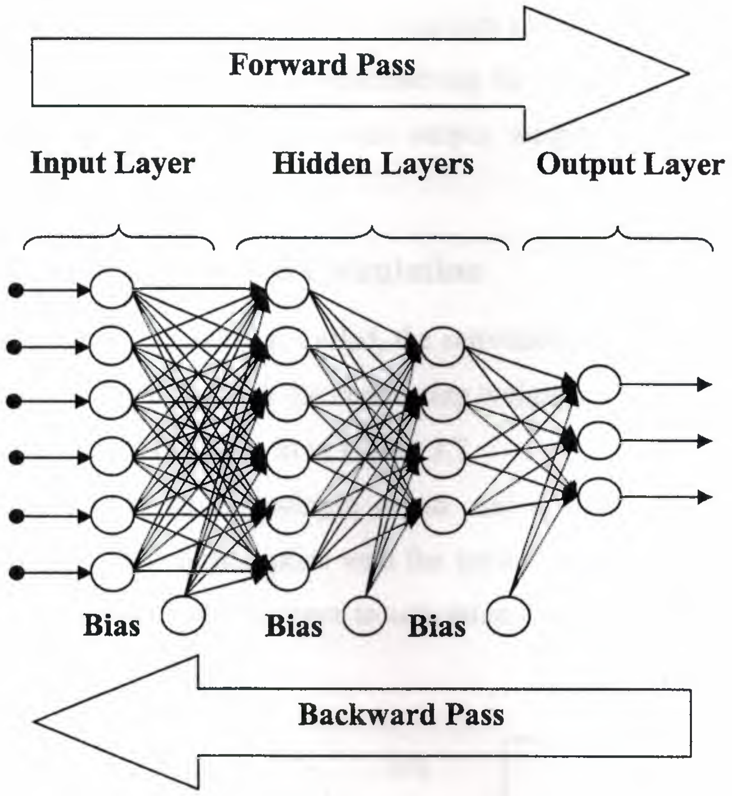


Figure 3.6 Back Propagation Network Architecture

3.5.2 Back Propagation Network Algorithm

In the back propagation learning algorithm, the network begins with a random set of weights. An input vector is fed forward through the network, and the output values are calculated using this initial weight set. Next, the calculated output is compared with the measured output data, and the squared difference between this pair of vectors determines the overall system error. The network attempts to minimize this error using the gradient descent approach, in which the network weights are adjusted in the direction of decreasing error.

The steps of back propagation algorithm can be listed as the following:

Step 1: Initialize hidden and output weights to small random values.

Step 2: Input training vector.

Step 3: Calculate outputs of hidden neurons.

Step 4: Calculate outputs of output neurons.

Step 5: Calculate the differences between the results of outputs of output neurons and targets.

Step 6: Back propagate the error to update the hidden and the output weights.

Step 7: Repeat the steps 3, 4, 5, and 6 until reaching the goal error.

Step 8: Upon conversion save hidden and output weights for use in feed forward calculations.

3.5.2.1 Feed Forward Calculation

When a back propagation network is cycled, the activations of the input units are passed forward to the output layer through the connecting weights. The starting point for most neural networks is a model neuron, as in Figure 3.7.

This neuron consists of multiple inputs and a single output. Each input is modified by a weight, which multiplies with the input value. The neuron will combine these weighted inputs and, with reference to activation function determine its output.

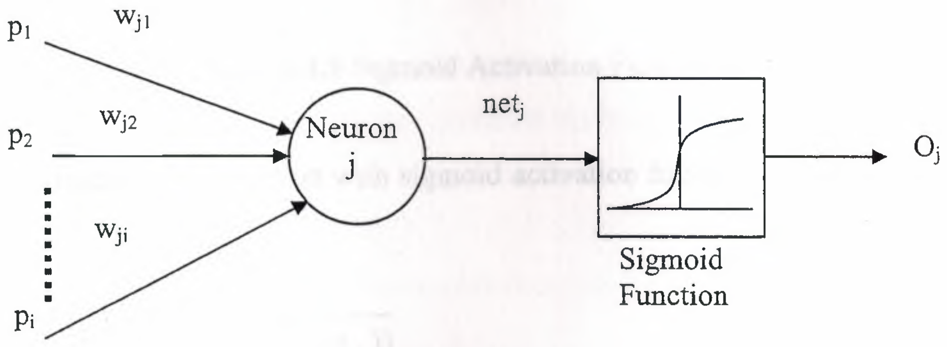


Figure 3.7 A model Neuron Structure

The main idea in feed forward calculation is passing inputs forward and all outputs are computed through sigmoid function. The output of each neuron is a function of its inputs.

In particular, the output of the ***j***th neuron in any layer is described by two sets of equations 3.2 and 3.3:

$$net_j = \sum p_i w_{ji} \tag{3.2}$$

$$O_j = f_{th}(net_j) \tag{3.3}$$

For every neuron, *j*, in a layer, each of the ***i*** inputs, p_j, to that layer is multiplied by a previously established weight, w_{ij}. These are all summed together, resulting in the internal value of this operation, net_j. This value is then sent through an activation function, f_{th}.

The activation function is usually the sigmoid function, which has an input to output mapping as shown in Figure 3.8. The resulting output, O_j, is an input to the next layer or it is a response of the neural network if it is the last layer.

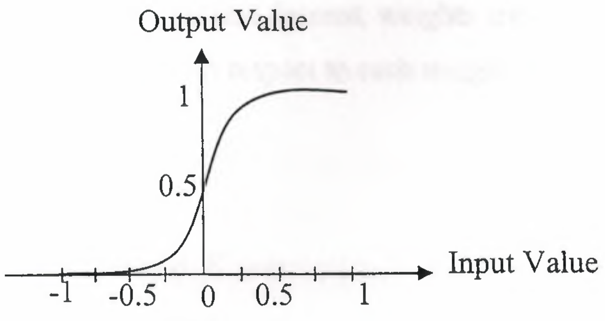


Figure 3.8 Sigmoid Activation Function

The output of the neuron with sigmoid activation function is given by equation 3.4:

$$O_j = f(net_j) = \frac{1}{(1+\exp(-net_j))} \tag{3.4}$$

The derivative of the sigmoid function can be obtained as follows equation 3.5:

$$\frac{\partial f(net_j)}{\partial net_j} = O_j * (1-O_j) = f(net_j) * (1 - f(net_j)) \tag{3.5}$$

3.5.2.2 Error Back Propagation Calculation

The error back propagation calculations are applied only during the training of the neural network. The vital elements in these calculations are the error signal, learning rate, momentum factor, and weight adjustment.

- **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 values 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 [27]. The error term represents a measure of how well a network is training on a particular training set.

A method called gradient descent is used to minimize the total error on the patterns in the training set. In gradient descent, weights are changed in proportion to the negative of an error derivative with respect to each weight given by equation 3.6:

$$\Delta w_{ji} = -\eta \left[\frac{\partial E}{\partial w_{ji}} \right] \tag{3.6}$$

where η is the learning rate and E is the average over all training instances of the sum over all output neurons (total error).

Weights move in the direction of steepest descent on the error surface defined by the total error (summed across patterns) given by equation 3.7:

$$E = \sum_p \sum_j (T_{Pj} - O_{Pj})^2 \tag{3.7}$$

where O_{pj} the actual output response to pattern p and T_{pj} is the target output value.

Figure 3.9 illustrates the concept of gradient descent using a single weight. After the error on each pattern is computed, each weight is adjusted in proportion to the calculated error gradient back propagated from the outputs to the inputs. The changes in the weights reduce the overall error in the network.

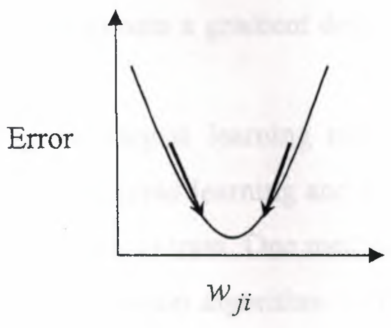


Figure 3.9 Typical Curve between Overall Error and A single Weight [26]

The aim of the training process is to minimize this error over all training patterns. From equation 3.2, it can be seen that the output of a neuron in the output layer is a function of its input, or $O_j = f_{th}(net_j)$. The first derivative of this function $O_j = f_{th}^{'}(net_j)$ is an important element in error back propagation. For output layer neurons, a quantity called the error signal is represented by Δ_{pj} which is defined in equation 3.8:

$$\Delta_{pj} = f_{th}^{'}(net_{pj}) * (T_{pj} - O_{pj}) = (T_{pj} - O_{pj}) * O_{pj} * (1 - O_{pj}) \qquad (3.8)$$

This error value 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.

- **Learning Rate and Momentum Factor**

There are two essential parameters that do affect the learning capability of the neural network. First the learning rate 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 energy minima. Both rates can have a value between 0 and 1.

The larger the learning rate η the larger the weight changes on each epoch, and the quicker the network learns. However, the size of the learning rate can also influence whether the network achieves a stable solution. If the learning rate gets too large, then the weight changes no longer approximate a gradient descent procedure. Oscillation of the weights is often the result.

The ideal case is using the largest learning rate possible without triggering oscillation. This would offer the most rapid learning and the least amount of time spent waiting at the computer for the network to train. One method that has been proposed is a slight modification of the back propagation algorithm so that it includes a momentum term.

- **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.

The weights that feed the output layer and the hidden layer are updated using equation 3.9. 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 local minima, the momentum term can be added as in equation 3.10.

$$w_{ji}(n+1) = w_{ji}(n) - \eta\Delta_{pj}O_{pi}^{T} \tag{3.9}$$

$$w_{ji}(n+1) = w_{ji}(n) - (1-\alpha)\eta\Delta_{pj}O_{pi}^{T} + \alpha[\delta w_{ji}(n)] \tag{3.10}$$

where the subscript n is the learning epoch and $\delta w_{ji}(n)$ stands for the previous weight change. The bias weights at the output and hidden layer neurons are updated, similarly [28].

3.5.3 Discussion Some Important Issues

There are some issues, which may cause some problem in neural networks, if totally ignored. For example input normalization before feeding data into a neural network is crucial. Moreover, appropriate weights initialization is needed. Also the other issues are very important in neural networks like training conversion criteria, various techniques/problems and generalization.

3.5.3.1 Input Normalization and Weights Initialization

The contribution of an input will depend heavily on its variability relative to other inputs. If for example one of the inputs has range of 0 to 1 and another has a range of 0 to 1000, then the contribution of the first input will be swamped by the second input. So it is essential to rescale the inputs so their variability reflects their importance. For lack of any prior information (regarding the importance of each input), it is common to normalize each input to the same range or the same standard deviation [29].

Typically inputs are normalized to same small ranges, like [0,1] or [-1,1]. In particular any scaling that gathers input values around zero works better. So instead of a [-1,1] scale, it might be preferable to normalize the inputs so as to have mean value of 0 and standard deviation of 1.

Weights initialization follows nearly the same path as input normalization. The main emphasis in the neural network literature on initial values has been on the avoidance of saturation, hence the desire to use small random values. Symmetry breaking in the weight space is needed in order to make neurons compute different functions. If all nodes have identical weights then they would respond identically. Therefore the gradient, which updates the weights, would be the same for each neuron. This way the weights would remain identical even after the update and this means no learning. A special case is to initialize all weights of every neuron to 0. Then in every neuron the gradient of a zero function would be zero and thus weights would remain zero until training is terminated.

Small weights (as well as small inputs) are needed to avoid immediate saturation because large weights could amplify a moderate input to produce an extremely large weighted sum at the inputs of the next layer. This would put the nodes into the flat regions of their nonlinearities and learning would be very slow because of the very small derivatives [30].

3.5.3.2 Training Conversion Criteria

Stopping of training when the back propagation network is trained has to be known. Since various "learning rate - momentum factor- number hidden neurons"-schemes are being tested to adapt the stopping criteria to each case in the network to get a good efficient learning.

Four basic termination conditions when training an artificial neural network:

- Fixed number of iterations: Iterations, also called epochs, refer to the number of times the total training set is being presented in the neural network.
- Use threshold for the error: Empirically estimate a certain value for the error, which considered being acceptable.
- Early stopping: Divide the available data into training and validation sets. Commonly use a large number of hidden units and very small initial values. Compute the validation error rate periodically during training. Finally, stop training when the validation errors rate "start to go up". However, it is important to stress that the validation error is not a good estimate of the generalization error. The most common method for getting an unbiased estimate of the

generalization error is to run the ANN on a third set of data, that is not used at all during the training process [31][32][33].

3.5.3.3 Techniques and Arising Problems

Multilayer Neural Networks have error surfaces with multiple local minima. The complexity of these surfaces increases as the number of weights (and so neurons) increases. Therefore, there is only one deepest global minimum among many shallow or deep local minimums. This means that the training procedure might get trapped into the latter "small" minima. In fact this is the case but there are two perspectives in relative bibliography that try to explain why artificial neural networks are still so much efficient and powerful tool [32].

- Many weights' means that error surfaces exist in high multidimensional spaces (one dimension for each weight). Someone would say that during back propagation one of the weights might fall in local minimum. But, other weights would not! Intuitively, the more the weights, the more dimensions exist, which provide "escape roots" from local minimums.
- Another perspective is the one, based on which sigmoid function behaves as linear when the weights are close to zero. This is the case during the first iterations of the neural network training. So in first steps the network simulates a smooth function. By the time the weights are "heavily" updated and the simulated function has much more complex error surface.

Back propagation's main problem is that it is sensitive to the so-called 'overfitting' of the training data at the cost of decreasing generalization accuracy over other unseen examples. It is said that when the overfitting case is faced the artificial neural network adopts the idiosyncrasies of the training data. This means that the performance over unseen examples decreases. Especially when the training set is not representative of the general distribution of all possible examples, the performance drops dramatically. In order to avoid overfitting, caused by the repetitive feed of the same group of training examples onto the artificial neural networks, the 'early stopping' technique is used.

It is necessary to remind that in 'early stopping' the total number of iterations of the training procedure is such that produces the lowest error over the validation set, since this is the best indicator over unseen examples. In other words, the number of iterations that yields the best performance over the validation set is needed. Another, potentially useful technique is called 'weight decay' or commonly regularization. This way, weights are kept small and the error surface smooth.

3.5.3.4 Generalization

Generalization is the ability of capturing the underlying function [31], during the training phase, and hence producing correct outputs in response to novel patterns (patterns that has not seen before). A system then is said to generalize well. If performance in new patterns is poor then poor is the generalization as well.

Minimizing the generalization error is not equivalent to selecting a model where the bias is zero. This is because the model variance penalty may be too high. This is called the bias/variance trade-off. Variance and bias are well-understood issues when it comes to regression problems (function approximation using Neural Networks). However, in classification there is a correspondence but it is surely more complex subject.

There are a few conditions that are typically necessary–although not sufficient- for good generalization:

- In order to generalize well, a system needs to be sufficiently powerful to approximate the target function. If it is too simple to fit even the training data then generalization to new data is also likely to be poor.
- The inputs contain sufficient information pertaining to the target, so that really existence a concept (unknown and complex mathematical function) that relates inputs with corrects outputs.
- In general, the training set must be a representative subset of the theoretical population. A poor set of training data may contain misleading regularities not found in the underlying function/classifier.

3.6 Summary

This chapter presented a general overview of artificial neural networks (ANN). The back propagation algorithm was also presented in detail since the algorithm is to be used in our face recognition system as a classifier.

The previous 3 chapters gave a background to understand the developed method for face recognition system which will be explained in the next chapter.

CHAPTER FOUR

FEATURE-AVERAGE BASED FACE RECOGNITION

4.1 Overview

An automatic system for the feature based face recognition must deal with three basic problems: detection of the human face in an image, extraction the essential features of the facial image; and finally the classification.

This chapter presents discussion of thesis topic about face recognition using back propagation neural networks. The method describes face recognition using back propagation neural networks approach implementation on multi-expression image sequence (natural, smiley, sad, and surprised). Instead of recognizing a face from a single view, a sequence of images showing face expressions is used as the face database. All the stages of face recognition system starting from capturing image to classifying the face (Recognized or Unrecognized) will be explained in detail in this chapter.

4.2 Image Acquisition

This stage does the capturing of the images using peripheral capturing devices. The image can be obtained from many sources. There are obvious ways to obtain digitized images: scanning a photo camera made picture, saving web-cam generated static or dynamic images, capturing with high quality video camera, or using other sources (e.g. manual scratches, painted images and results of other digital image processing procedures).

4.2.1 Capturing Device

The acquisition of the image is the first step of the image processing procedure. The input has to be a digital format picture.

The easiest and quickest way to obtain a digital image is using digital photo cameras. Good digital cameras are considered to have resolution above 3 mega-pixels. They can be used for image capturing, in a digital system, if they have an appropriate interface. This is satisfied by 99% of the cameras: they have at least one of the serial, fire-wire, analog (composite), or –the most frequent– USB interfaces. The digital photo camera that used in this work has 6.6 mega-pixels resolution.

4.2.2 Environmental Prerequisites

The environmental condition of the image acquisition has many effects on the algorithms that are used in further steps. If they are not insuring satisfactory conditions, then the captured images may not be usable

In this research, to determine a good results, some condition and constrain are considered. These conditions are:

- No physical obstruction
- The light source is the same for all face images
- The head is straight in its normal vertical position without rotation or tilting
- Camera position will be at the same distance from the face (150 to 200 cm)
- The hair doesn't cover important face features

These conditions are used in different processing steps and are considered as insured by the environment.

4.3 Database Collection

The database of images which are being used in the experiments only contains the faces and the dimension of images as a fixed dimension (100x100). The database contains 90 face images in the same resolution and the same lightning conditions. The 90 images database comprises 6 different facial expressions of 15 persons. Four out of the 6 facial expressions (natural, smiley, sad and surprised) will be reduced by averaging to one face image, thus providing 15 averaged face images for training the neural network. Generalizing the neural network will be implemented using the remaining 2 facial expressions, thus providing 30 face images for testing the trained neural network.

In addition to 90 image database, other faces will also be used for generalization. These include some images of persons with eye glasses and dark glasses. Finally, the developed system is implemented using ORL face database [34].

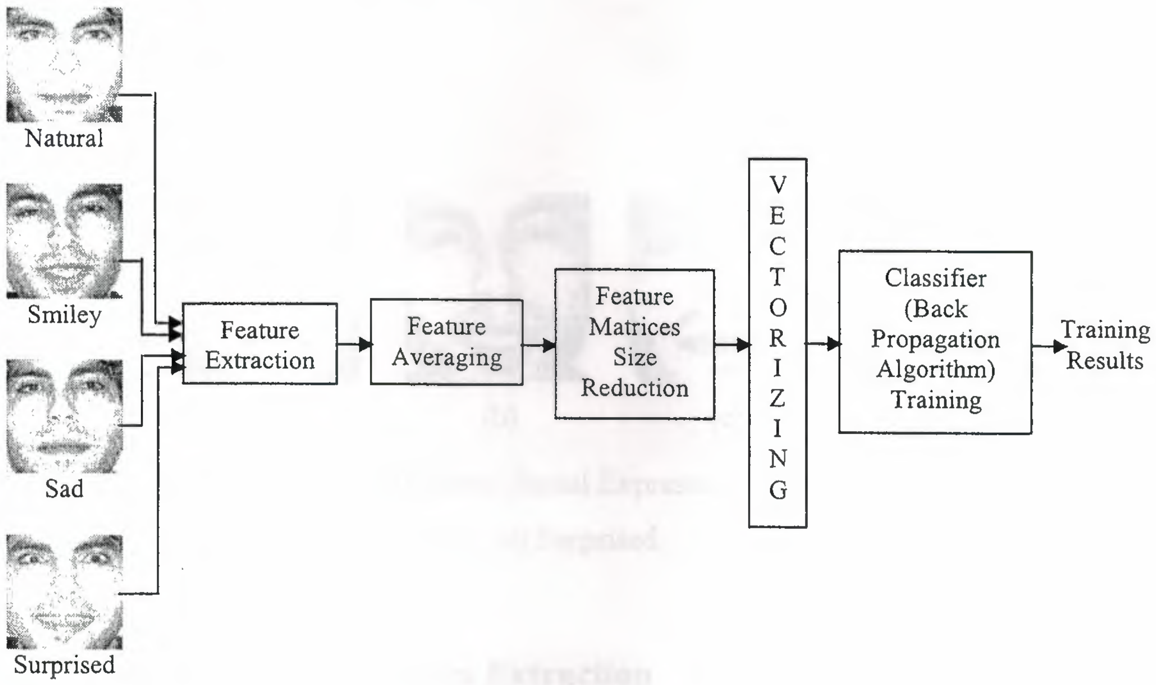
4.4 Automatic Face Recognition System

4.4.1 General Architecture

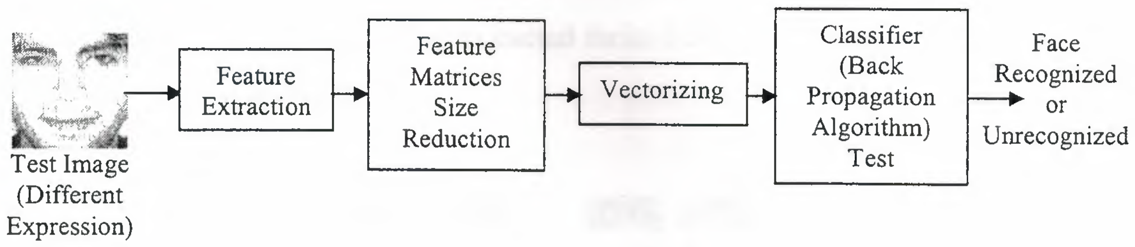
The block diagram of the proposed automatic system of face recognition is shown in Figure 4.1. The steps of the system are:

- The first step is extracting the main features of the face (eyes, nose, and mouth) for each facial expression (natural, smiley, sad and surprised) of each subject.

- The second step is taking the average of each feature for all the facial expressions of the face of the subject.
- The third step is reducing the dimensions of the matrices of each feature by averaging.
- The fourth step is representing the averaged features as vectors, instead of the matrix representation.
- The final step is classifying the subjects recognized or unrecognized by using a back propagation neural network.



(a) TRAINING


Test Image (Different Expression)
Feature Extraction
Feature Matrices Size Reduction
Vectorizing
Classifier (Back Propagation Algorithm) Test
Face Recognized or Unrecognized

(b) GENERALIZATION

Figure 4.1 Block Diagram of Feature-Average Based Face Recognition Using Back Propagation Neural Networks

4.4.2 Phases of the Automatic Face Recognition System

4.4.2.1 Preprocessing

The preprocessing in any face recognition system is an important stage. The face images sizes have been standardized by resize the images to dimension (100x100) for each subject with different facial expression such as natural, smiley, sad, and surprised. The dimension of face images has to be the same to locate the features positions. Different facial expression grayscale images (natural, smiley, sad, and surprised) are shown in Figure 4.2.

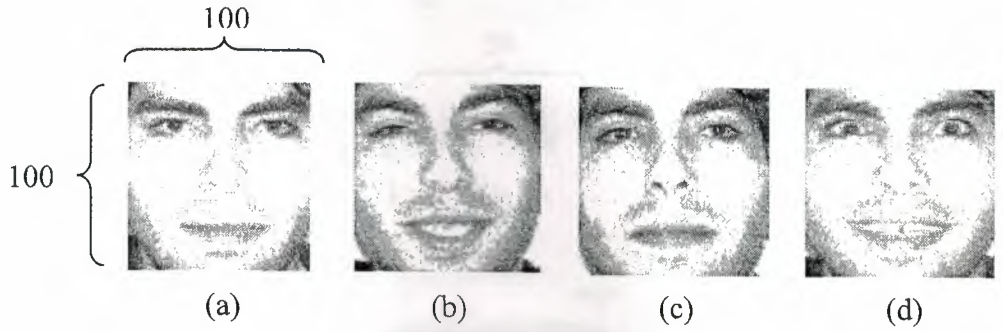


Figure 4.2 The Face Image in Different Facial Expression: (a) Natural. (b) Smiley. (c) Sad. (d) Surprised.

4.4.2.2 Facial Features Extraction

In this stage, the locations of main facial features (eyes, nose, and mouth) are assumed as known. The facial features (eyes, nose, and mouth) for each facial expression of the face images in the database have been extracted. The extracted facial features dimension for each feature has to be same. The extracted facial features of one face image from the database are shown in Figure 4.3.

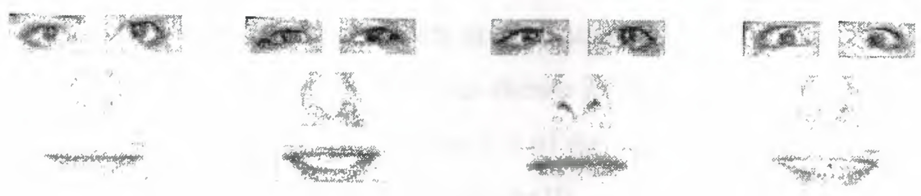


Figure 4.3 Extracted Features in Different Facial Expression.

The dimension of matrices of the extracted facial features is shown in Figure 4.4.

- Right eye dimension is 15x30 pixels.
- Left eye dimension is 15x30 pixels.
- Nose dimension is 21x30 pixels.
- Mouth dimension is 18x51pixels.

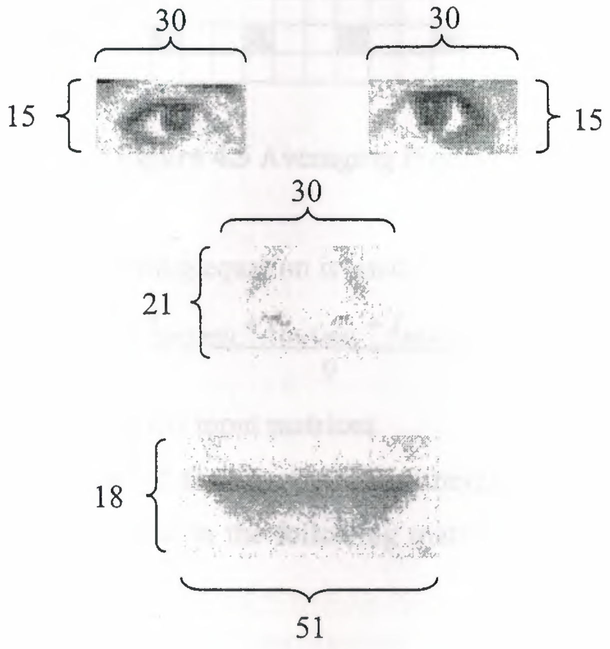


Figure 4.4 Extracted Facial Features Dimensions in Pixels

4.4.2.3 Resizing by Averaging

The size of extracted facial features dimension is very large so it is not practical to use it in the process. The larger number of input means data longer processing time. To reduce the time processing, reducing the size of the extracted facial features is a must. By using interpolation, the dimension can be reduced according to the size of the interpolation.

The reduced matrices pixel values are calculated from a weighted average of pixels in the nearest 3-by-3 neighborhood as shown in Figure 4.5. The output matrices dimension after interpolation process will be 1/3 of the input matrices. For example, the 15x30 input matrices will be after interpolation 5x10.

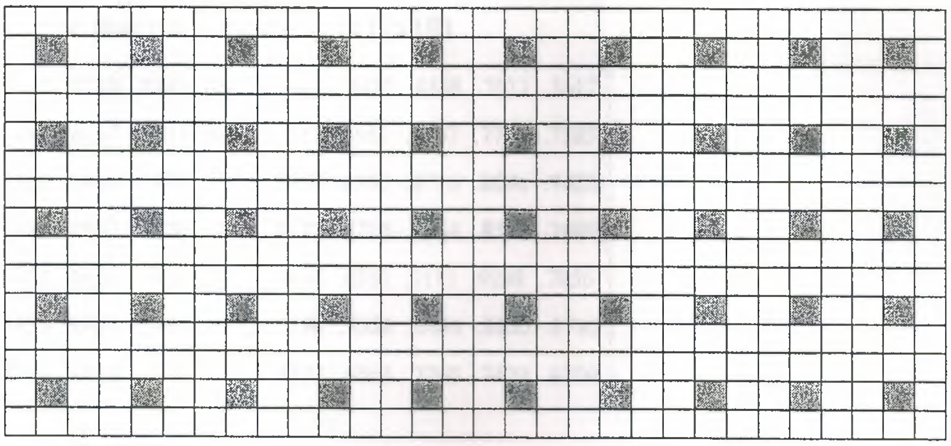


Figure 4.5 Averaging Process

For averaging the following equation is used:

$$O_{(m,n)} = \frac{I_{(m,n)} + I_{(m+1,n)} + I_{(m,n+1)} + I_{(m+1,n+1)} + I_{(m-1,n-1)} + I_{(m,n-1)} + I_{(m-1,n)} + I_{(m+1,n-1)} + I_{(m-1,n+1)}}{9}$$

where O is output matrices and I is input matrices.

By taking the average of the nearest 3-by-3 neighbor pixels the dimension of output matrices will be resized as in the following matrices. Table 4.1 summarizes the image database resizing process.

Resized right eye matrix dimension is (5x10)

$$\begin{bmatrix} .7272 & .5800 & .5464 & .5133 & .5072 & .4911 & .4636 & .4427 & .4418 & .4632 \\ .6667 & .6645 & .6645 & .6370 & .6048 & .5699 & .5412 & .4980 & .4819 & .4963 \\ .7338 & .7246 & .6253 & .5407 & .5455 & .5381 & .4941 & .4584 & .4340 & .4649 \\ .7743 & .6488 & .5490 & .5320 & .4340 & .3756 & .3638 & .3882 & .3952 & .4309 \\ .7207 & .5329 & .4824 & .4815 & .3939 & .3285 & .3246 & .3978 & .3908 & .4057 \end{bmatrix}$$

Resized left eye matrix dimension is (5x10)

$$\begin{bmatrix} .5281 & .4693 & .3991 & .3582 & .3695 & .3834 & .4052 & .4627 & .5756 & .7020 \\ .5442 & .4972 & .4723 & .4876 & .5342 & .5721 & .5956 & .6275 & .6597 & .6536 \\ .4797 & .4501 & .4336 & .4776 & .5168 & .5486 & .5582 & .6031 & .6915 & .7272 \\ .4466 & .4031 & .3778 & .3673 & .3734 & .4514 & .5660 & .6131 & .5795 & .7115 \\ .4532 & .3643 & .3813 & .3068 & .2924 & .3373 & .4819 & .5115 & .5577 & .6388 \end{bmatrix}$$

Resized nose matrix dimension is (7x10)

$$\begin{bmatrix} .6353 & .7577 & .8710 & .8349 & .8017 & .8327 & .8915 & .8588 & .7011 & .5617 \\ .7438 & .8484 & .8837 & .8431 & .8061 & .8327 & .8932 & .8667 & .7769 & .7285 \\ .8031 & .8532 & .8488 & .8479 & .8148 & .8362 & .8937 & .8776 & .8096 & .7625 \\ .7861 & .8362 & .8597 & .8370 & .7965 & .8122 & .8715 & .9264 & .8540 & .7089 \\ .8235 & .8728 & .8636 & .7913 & .7569 & .7695 & .8331 & .9111 & .9298 & .7856 \\ .8989 & .8623 & .8004 & .7407 & .7150 & .7198 & .7556 & .8444 & .8850 & .8780 \\ .7908 & .7259 & .6928 & .6706 & .6444 & .6532 & .6863 & .7203 & .7420 & .8200 \end{bmatrix}$$

Resized mouth matrix dimension is (6x17)

$$\begin{bmatrix} .5930 & .6192 & .6331 & .6458 & .6235 & .5638 & .4941 & .4497 & .4510 & .4671 & .5625 & .6745 & .7255 & .7229 & .6932 & .6427 & .5882 \\ .6296 & .7059 & .6915 & .6183 & .5486 & .5107 & .5102 & .5220 & .5281 & .5242 & .5390 & .5564 & .6305 & .7085 & .7564 & .7146 & .6122 \\ .7098 & .6514 & .5181 & .4697 & .4527 & .4484 & .4462 & .4410 & .4401 & .4501 & .4715 & .4641 & .4645 & .4780 & .5564 & .6336 & .6540 \\ .5686 & .4057 & .3895 & .4105 & .4283 & .4444 & .4340 & .4218 & .4379 & .4536 & .5002 & .5168 & .4972 & .4510 & .3817 & .4135 & .5874 \\ .4906 & .4139 & .4924 & .5861 & .6231 & .6118 & .5987 & .5895 & .5917 & .6196 & .6736 & .7159 & .7207 & .6863 & .6458 & .6780 & .6667 \\ .6240 & .6654 & .6837 & .6985 & .6427 & .5874 & .5656 & .5747 & .5839 & .5974 & .6497 & .7020 & .7368 & .7917 & .8065 & .7908 & .7590 \end{bmatrix}$$

Table 4.1 Resizing Process

Facial Feature	Extracted Feature	Averaged Feature
Right Eye	15x30 = 450 pixels	5x10 = 50 pixels
Left Eye	15x30 = 450 pixels	5x10 = 50 pixels
Nose	21x30 = 630 pixels	7x10 = 70 pixels
Mouth	18x51 = 918 pixels	6x17 = 102 pixels
	Total = 2448 pixels	**Total = 272 pixels**

4.4.2.4 Implement Averaging Method

For each facial feature (right eye, left eye, nose, and mouth) of each face image, the average for the different facial expressions (natural, smiley, sad, and surprised) of this feature will be calculated from the following equations.

Averaged Right Eye

$$A\_R\_E = \frac{1}{4}\sum_{i=1}^{4} R\_E_i, \qquad i = 1,2,3,4$$

Averaged Left Eye

$$A\_L\_E = \frac{1}{4}\sum_{i=1}^{4} L\_E_i, \qquad i = 1,2,3,4$$

Averaged Nose

$$A\_N = \frac{1}{4}\sum_{i=1}^{4} N_i, \qquad i = 1,2,3,4$$

Averaged Mouth

$$A\_M = \frac{1}{4}\sum_{i=1}^{4} M_i, \qquad i = 1,2,3,4$$

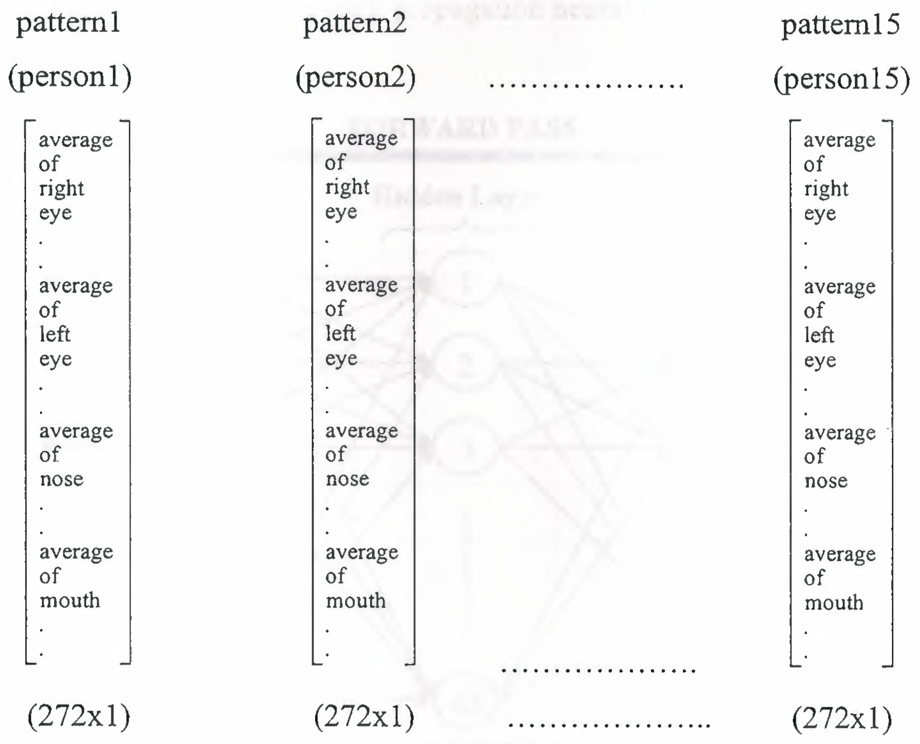
The averaged facial features are combined, side-by-side, to create a data matrices of size 272x15 as following:

$$\text{PATTERNS} = \begin{bmatrix} A\_R\_E_{p1} & A\_R\_E_{p2} & \cdot\ \cdot & A\_R\_E_{p15} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ A\_L\_E_{p1} & A\_L\_E_{p2} & \cdot\ \cdot & A\_L\_E_{p15} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ A\_N_{p1} & A\_N_{p2} & \cdot\ \cdot & A\_N_{p15} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ A\_M_{p1} & A\_M_{p2} & \cdot\ \cdot & A\_M_{p15} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \end{bmatrix}$$

4.4.2.5 Patterns Vectorizing

To prepare the training set patterns to starting the training the neural network, the matrices of the features (right eye, left eye, nose, and mouth) have been vectorized as patterns.

The dimension of each pattern from the 15 averaged faces will be a vector of 272 pixels as following.



4.4.2.6 Classification Using Back Propagation

Establishing back propagation neural network to classify the patterns (features) as recognized or not recognized.

Back Propagation algorithm is the most commonly used neural network. This is due to its simplicity and efficiency. The back propagation network is suitable for solving pattern recognition problems [16].

The Back-Propagation neural network that is used in the developed system comprises an input layer, a hidden layer and an output layer. The input layer receives input (with 272 neurons) directly that carries the values of the averaged features.

The size of the input layer should match exactly the size of the input pixels numbers (272). The number of neurons in the hidden layer is 65 neurons. The number of neurons in the output layer is 15 neurons which is the number of persons.

The desired output for each digit is 1 on the corresponding node and 0 on all other nodes. The back propagation neural network here works as classifier. The number of patterns equal to number of faces (15).

Each pattern contains 272 values representing the average of facial features of one person. Figure 4.6 shows the back propagation neural network architecture.

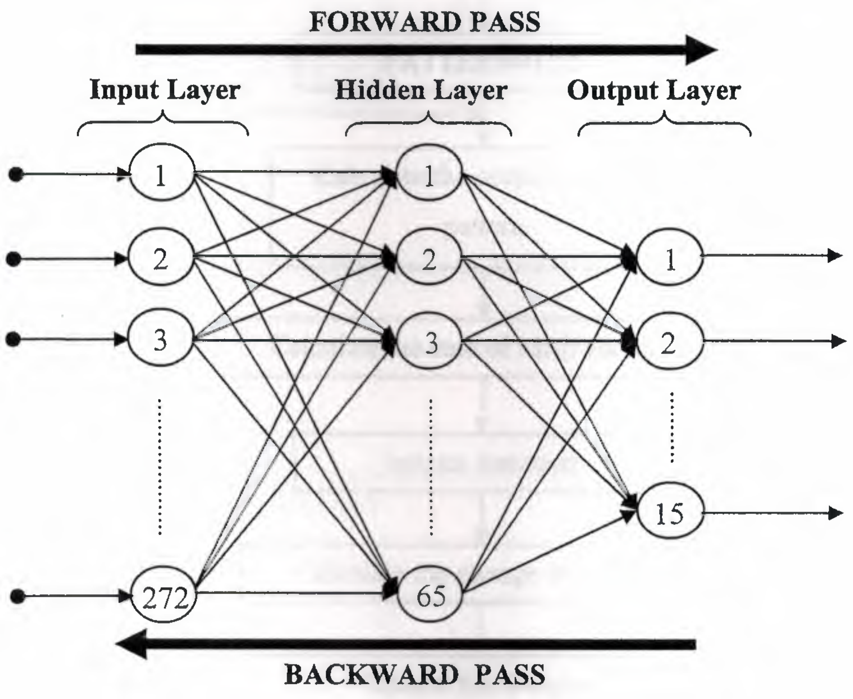


Figure 4.6 Architecture of the Back Propagation Neural Networks.

The flowchart of back propagation algorithm that used as classifier in face recognition system is shown in Figure 4.7.

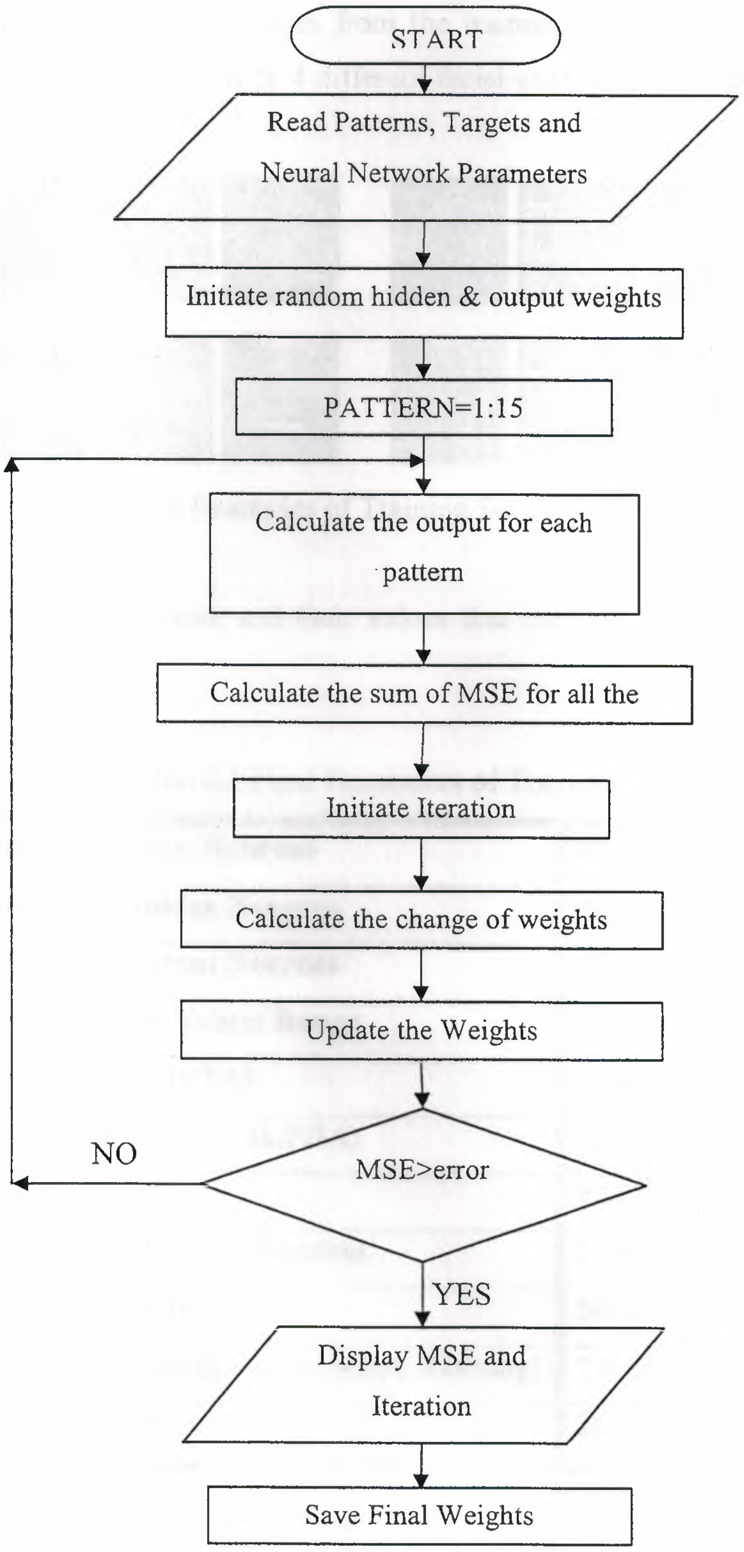

START
Read Patterns, Targets and
Neural Network Parameters
Initiate random hidden & output weights
PATTERN=1:15
Calculate the output for each pattern
Calculate the sum of MSE for all the
Initiate Iteration
Calculate the change of weights
Update the Weights
MSE>error
NO
YES
Display MSE and Iteration
Save Final Weights

Figure 4.7 Flowchart of Neural Network Training

4.5 Experimental Results

4.5.1 Training the Face Images

Figure 4.8 shows some sample images from the training set database. The database contains 15 subject face images with 4 different facial expressions. Appendix II shows all image databases.

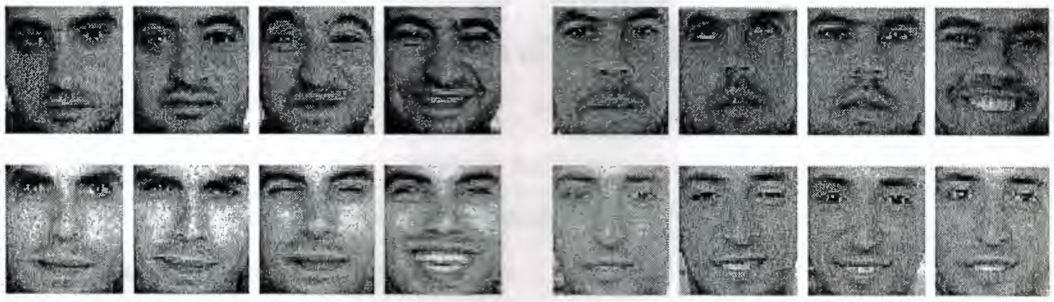


Figure 4.8 Examples of Training Set Face Images

The essential parameters and their values that used in the training have been listed in the Table 4.2.

Table 4.2 Final Parameters of Training

Number of Input Neurons	**272**
Number of Hidden Neurons	**65**
Number of Output Neurons	**15**
Initial Weights Values Range	**-0.3 to 0.3**
Learning Rate (ETA)	**0.0495**
Momentum Factor (ALPHA)	**0.41**
Error	**0.001**
Number of Training Iteration	**3188**
Maximum Iteration	**5000**
Processing Time (Extracting and Resizing)	**7.5 Seconds***
Training Time	**265 Seconds***

* Time results were obtained using a 1.6 GHz PC with 256 Mb of RAM, Windows XP and the Matlab 6.5 software.

4.5.2 Testing the Face Images

All images in the test set are tested without using the same images that used in the training set. Some example of test set database is shown in Figure 4.9.

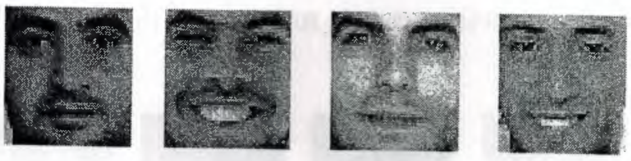


Figure 4.9 Examples of Test Set Face Images

The recognition rates, recognition accuracy and running time of the training set and test set is show in Table 4.3. The recognition accuracy is the average of the 15 person's images training or test results.

Table 4.3 Recognition Rates, Accuracy and Run Time of Training and Test Sets

Image Set	Recognition Rate	Recognition Accuracy	Run Time
Training Set	(15/15) 100%	95.6%	0.032 Second*
Testing Set	(30/30) 100%	93%	

* Time results were obtained using a 1.6 GHz PC with 256 Mb of RAM, Windows XP and the Matlab 6.5 software.

In Figure 4.10 the mean square error (MSE) vs. iteration graph with 15 averaged face images training set is shown.

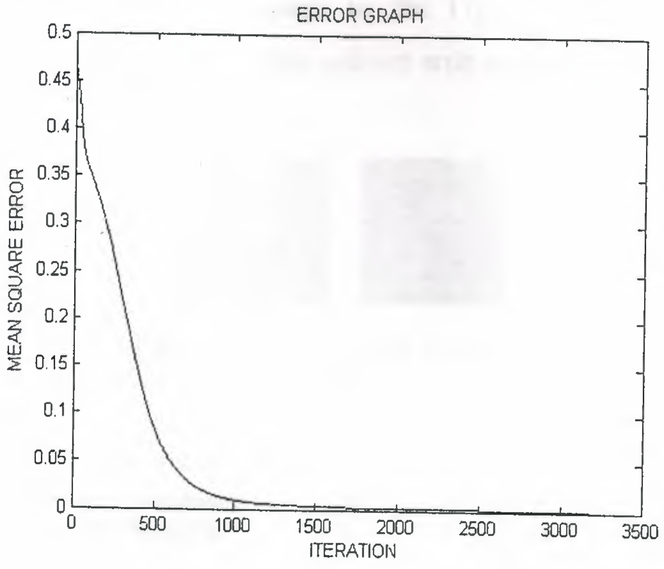


Figure 4.10 Mean Square Error vs. Iteration Graph

4.5.3 Recognition Performance with Glasses

In the experiment, the effect of the presence of facial detail such as glasses on recognition performance was tested. In Figure 4.11 the training set of the subject and the test set which contains faces with and without glasses of the subject.

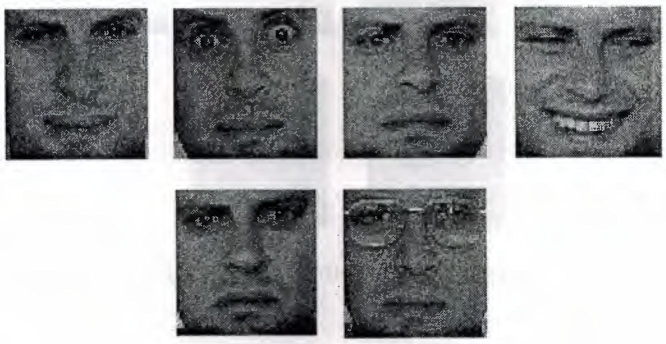


Figure 4.11 Training Set and Test Set with Eye Glasses

The recognition accuracy of the test set of this subject with and without eye glasses is shown in Table 4.4.

Table 4.4 Recognition Accuracy of Face with and without Eye Glasses

Recognition Accuracy of Face without Glasses	**96%**
Recognition Accuracy of Face with Eye Glasses	**86%**

The closed eyes face test image in the Figure 4.12 also was tested. The recognition accuracy of the test set of this subject with and without eye glasses is shown in Table 4.5.

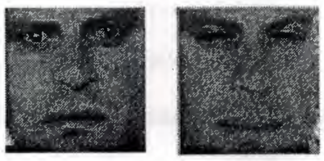


Figure 4.12 Open and Closed Eyes Test Set Face

Table 4.5 Recognition Accuracy of Face Open and Closed Eyes

Recognition Accuracy of Face without Glasses	**96%**
Recognition Accuracy Face with Closed Eyes	**80%**

In Figure 4.13, the train set and the test set of another subject with dark glasses are shown.

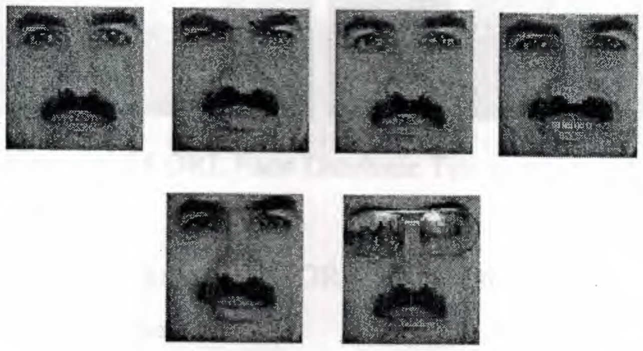


Figure 4.13 Training Set and Test Set with Dark Glasses

The recognition accuracy of the test set of this subject with and without dark glasses is shown in Table 4.6.

Table 4.6 Recognition Accuracy of Face with and without Dark Glasses

Recognition Accuracy of Face without Glasses	**90%**
Recognition Accuracy of Face with Glasses	**63%**

4.5.4 Experiments on ORL Face Database

There are 10 different images of 40 distinct subjects in the ORL face database [34]. For the subjects, the images were taken with slightly varying lightings, facial expressions (open/closed eyes, smiling/ non-smiling) and facial details (glasses/no-glasses). All the subjects are in frontal position but with some pose variation.

To implementation on developed face recognition system, 4 different images of 15 distinct subjects and one image of the same 15 subjects from ORL face database as training set and test set respectively. Some example images from ORL face database training set are shown in Figure 4.14. Appendix II shows all the images of ORL face database, which are used in training and testing.

Figure 4.14 ORL Face Database Training Set [34]

In Figure 4.15 some examples of ORL face database test set images, which are used for testing the neural network.

Figure 4.15 ORL Face Database Test Set [34]

Training and test recognition rates, accuracy rates and running time of the system by applying the ORL face database are shown in Table 4.7.

Table 4.7 Recognition Rate, Recognition Accuracy and Run Time of Training and Test Sets of ORL Database

Image Set	**Recognition Rate**	**Recognition Accuracy**	**Run Time**
Training Set	(15/15) 100%	95.6%	0.032 Second*
Testing Set	(15/15) 100%	93.5%	

* Time results were obtained using a 1.6 GHz PC with 256 Mb of RAM, Windows XP and the Matlab 6.5 software.

4.6 Comparison with Other Face Recognition Methods

A comparison has been drawn between the developed method and 2 of the most efficient methods; namely, PCA (Eigenfaces) and LDA (Fisherfaces).

The ORL face database has been used for comparison the results of the face recognition methods. The recognition rates and accuracies of the face recognition methods using ORL face database are listed in table 4.8.

Table 4.8 Results of Different Methods of Face Recognition Using ORL Database

Method	Recognition Rate	Recognition Accuracy
PCA (Eigenfaces)	64%-96% [5]	79.1% [35]
LDA (Fisherfaces)	99.4% [6]	81% [35]
Feature-Average	100%	93.5%

4.7 Analysis and Discussion

In experimental results on our database, the developed system recognized with a good recognition accuracy all the 15 person's images in the test set.

In training process, a combination of a minimum error MSE and high accuracy of recognition is an important factor in the classifying process. Minimum error and high accuracy are determined by varying the back propagation parameters (learning rate, momentum factor, and hidden layer neurons number).

Selecting the important features of the face, which are usually not changeable allowed the reduction of the dimensions of the data thus providing less processing time in the system. Compressing the selected facial features also decreased the processing time.

Averaging the facial features of the four different facial expressions (natural, smiley, sad and surprised) made the system convenient to respond correctly to other face expressions.

Experimental results have shown that, the presence of some details in the face such as eye glasses did not cause a major problem to the system, whereas the presence of dark glasses (sun glasses) which decreased the recognition accuracy, because the dark sun glasses cover 2 essential features from the 4 features of the face.

The performance of the system has been illustrated by the implementation using the other database such as ORL face database, which contains images with small variations in illumination and orientation. The result of the method has been compared with 2 other efficient face recognitions methods namely, PCA (Eigenfaces) and LDA (Fisherfaces) using ORL database.

The efficiency of the method suggested in this thesis has been shown where 100% recognition rates with high recognition accuracy have been obtained.

4.8 Software Tools (MATLAB)

This section contains a simple description of the tools that were used. The implementation has been done using Matlab version 6.5, Image Processing Toolbox, and Neural Networks Toolbox.

Matlab is a simulation environment for doing numerical computations with matrices and vectors. It handles a wide range of computing tasks in engineering and science, and has several built-in interfaces that let you quickly access and manipulate data. The Matlab environment integrates mathematical computing, visualization and a powerful technical language. A large user community spread throughout industry, government and academia makes Matlab a recognized standard worldwide for technical computing. Matlab is used in a variety of application areas including signal and image processing, neural networks, control system design, earth and life sciences, finance and economics and instrumentation. The open architecture makes it easy to use Matlab to explore data and create custom tools. One interesting feature is that Matlab applications can be converted to standalone applications using the C and C++ compiler.

In addition there are several toolboxes available to expand the capabilities of Matlab. The image processing toolbox extends the Matlab computing environment to provide functions and interactive tools for enhancing and analyzing digital images and developing image processing algorithms. The Neural Network Toolbox is one of these toolboxes. The neural network toolbox makes it easier to use neural network algorithms.

4.9 Summary

This chapter explained in detail the developed face recognition system. The experimental results were also discussed in this chapter, which demonstrated the successful implementation of the developed method.

CONCLUSION

Face recognition is a difficult problem because faces can vary substantially in their orientation, lighting, scale and facial expression. In this work, I presented an approach for feature-average based face recognition using back propagation neural networks.

As the thesis relies on feature based face recognition, the dimensionality reduction of the features matrices are applied by selecting the essential features of the faces (eyes, nose, and mouth). Also the dimension of the features matrices are reduced by implementing interpolation (averaging). Both feature selecting and feature averaging have been used to provide efficient face recognition while keeping minimal time cost.

Averaging the facial features of the four different facial expressions (natural, smiley, sad and surprised) made the system convenient to response correctly to other face expressions.

Experimental results have shown that, the presence of small details such as eye glasses do not cause a major problem to the system, but the presence of dark glasses (sun glasses) decreased the recognition accuracy.

By using the back propagation neural networks as classifier, the performance of the neural system can be monitored by changing the parameters: learning rate, momentum factor, and hidden layer neurons number.

Implementing the system on ORL face database which contains face images with small variations in illumination and orientation made the system suitable for real-life application.

Training processing time of neural network consumed 265 seconds. The processing time for testing one face consumed 0.032 seconds. The developed face recognition system achieved 100% recognition rates on our database and on ORL database. The system also has high recognition accuracy (93%) using both database.

Based on the findings and experimental results in this thesis, the following problems are suggested for further research:

- Recognition of more facial expression by adding more expressions.
- Considering large variation in illumination and orientation.
- Detecting the face and the facial features automatically to develop a complete automatic face recognition system.

REFERENCES

[1] Stephen Cobb (1996), The NCSA Guide to PC and LAN Security, New York, McGraw-Hill.

[2] Sung, K. K. and T. Poggio (1994), Example-based Learning for View-based Human Face Detection, MIT AI Lab. technical report, Dec.

[3] Atalay I. (1996), “Face Recognition using Eigenfaces”. M. Sc. thesis, Istanbul Technical University.

[4] R. Chellappa, C.L.Wilson, and S. Sirohey (1995). Human and Machine Recognition of Faces: A survey. PIEEE, 83(5):705–740, May.

[5] Turk, M., Pentland, A. (1991): Eignefaces for Recognition. Journal of Cognitive Neuroscience, Vol. 3, 72-86

[6] Belhumeur, P., Hespanha, J., Kriegman D. (1996): Eigenfaces vs. Fisherfaces: Face Recognition using class specific linear projection. In Proc. ECCV, 45-58

[7] M. A. O. Vasilescu, D. Terzopoulos: Multilinear (2002): Image Analysis for Facial Recognition, Proceedings of International Conference on Pattern Recognition

[8] S.T. Roweis and L.K. Saul (2000). Nonlinear dimensionality reduction by locally linear embedding. Science, 290, pp. 2323–2326.

[9] L. R. Rabiner (1989). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, Proc. of the IEEE, Vol.77, No.2, pp.257—286.

[10] Ferdinando Silvestro, Samaria (1994). “Face Recognition Using Hidden Markov Models”. PhD thesis, Univ. of Cambridge

[11] Ara V. Nefian (1999). “A Hidden Markov Model-Based Approach for Face Detection and Recognition”. PhD thesis, Georgia Institute Technology

[12] Andrew J. Viterbi (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Transactions on Information Theory 13(2):260–267.

[13] L. E. Baum, T. Peterie, G. Souled, and N. Weiss (1970). "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains," Ann. Math. Statist., vol. 41, no. 1, pp. 164—171.

[14] G. W. Cottrell and M. Flemming (1990). "Face Recognition Using Unsupervised Feature Extraction" in Proceedings International Neural Network Conference, pp. 322-325.

[15] Steve Lawrence, C. Lee Giles, Ah Chung Tsoi and Andrew D. Baek (1997). "Face Recognition: A Convolutional Neural Network Approach", IEEE Transactions on Neural Networks, vol. 8, no. 1, pp. 98-113.

[16] Simon Haykin (1999). *Neural Networks: A Comprehensive Foundation*, Prentice Hall Inc., New Jersey.

[17] http://www.teco.edu/~albrecht/neuro/html/node7.html

[18] Andreas Zell (1994). Simulation Neuronaler Netze. Addison-Wesley.

[19] Murray L. Barr & John A. Kiernan (1988). The Human Nervous System. An Anatomical Viewpoint. Fifth Edition. Harper International.

[20] Rumelhart, D.E., Hinton, G.E., and Williams, R.J (1985). "Learning Internal Representations by Error Propagation", Institute for Cognitive Science Report 8506, San Diego, University of California.

[21] http://www.dacs.dtic.mil/techs/dacs_reports/text/neural_nets.txt

[22] Hebb, D. O (1949). The Organization of Behavior, Wile, New York.

[23] Hopfield, John J. (1986). "Physics, Biological Computation and Complimentarity", The Lessons of Quantum Theory, Elsevier Science Publishers, B.B.

[24] Widrow, Bernard (1960). "An Adaptive 'Adaline' Neuron Using Chemical 'Memistors'", Technical Report Number 1553-2, Stanford Electronics Laboratories, October.

[25] http://www.dacs.dtic.mil/techs/dacs_reports/text/neural_nets.txt

[26] Kohonen, T. (1988). Self-Organization and Associative Memory, Second Edition, Springer-Verlag, New York.

[27] Rumelhart, D. E., & McClelland, J. L. (Eds.) (1986). Parallel distributed processing: Explorations in the microstructure of cognition (Vol. 1). Cambridge, MA: MIT Press.

[28] Martin T., Howard B., Mark Beale (1996). Neural Network Design, PWS Publishing Company, Boston.

[29] http://www.faqs.org/faqs/ai-faq/neural-nets/

[30] http://www.ee.surrey.ac.uk/Teaching/Courses/eem.pat/MScThesis.pdf

[31] Russell D. Reed, Robert J. Marks II (1999). *Neural Smithing: Supervised Learning in Feedforward Artificial Neural Networks (A Bradford Book),* The MIT Press.

[32] Thomas Mitchell (1997). *Machine Learning*, McGraw-Hill Editions International, New York.

[33] Rich Caruana, Steve Lawrence, Lee Giles (2000). Overfitting in Neural Nets: Backpropagation, Conjugate Gradient and Early Stopping, Neural Information Processing Systems, Denver, Colorado, November 28-30.

[34] Cambridge University, Olivetti Research Laboratory face database, http://www.uk.research.att.com/facedatabase.html

[35] Xiaoguang Lu, Yunhong Wang and Anil K. Jain, (2003). "Combining Classifiers for Face Recognition," IEEE International Conference on Multimedia & Expo (ICME'03), vol. III, pp. 13-16, Baltimore, MD.

APPENDICES

Appendix I Matlab Source Code

```
clear all
close all
clc
image_number = 15;
face_expression = 4;
PATTERNS = [];
t = cputime;                        % Initiate Processing time for feature extraction
cd( 'D:\TRAINSET\')                 % Change the directory to get the images in

for i = 1:image_number              % Start for loop to read the face images
r_eye_t = zeros(5,10);              % Preallocate empty matrices
l_eye_t = zeros(5,10);
nose_t = zeros(7,10);
mouth_t = zeros(6,17);

for j = 1:face_expression           % Start for loop to read the different facial expression images
train_image = strcat(['subject' int2str(i),'_' int2str(j),'.jpg']);
colour_image = imread(train_image);
face = rgb2gray(colour_image);    %Convert the colour image to grayscale image

r_eye = imcrop(face,[10 18 29 14]);          % Crop selected skin region (right eye)
r_eye=double(r_eye)/255;                     % Normalize the values of matrices between (0,1)
for m=2:3:14;                                % start for loop to read matrices
   for n=2:3:29;
reyer(m,n)=(r_eye(m,n)+r_eye(m+1,n)+r_eye(m,n+1)+r_eye(m+1,n+1)+r_eye(m-1,n-1)+r_eye(m,n-
1)+r_eye(m-1,n)+r_eye(m+1,n-1)+r_eye(m-1,n+1))/9;
% Take the average for each 9 pixel to resize the
% dimension of matrices
end
end
s = nonzeros(reyer);
r_eye_r = reshape(s,5,10);

l_eye = imcrop(face,[62 18 29 14]);        % Crop selected skin region (left eye)
l_eye=double(l_eye)/255;
for m=2:3:14;
   for n=2:3:29;
leyer(m,n)=(l_eye(m,n)+l_eye(m+1,n)+l_eye(m,n+1)+l_eye(m+1,n+1)+l_eye(m-1,n-1)+l_eye(m,n-
1)+l_eye(m-1,n)+l_eye(m+1,n-1)+l_eye(m-1,n+1))/9;
   end
end
s1 = nonzeros(leyer);
l_eye_r = reshape(s1,5,10);

nose = imcrop(face,[38 38 29 20]);           % Crop selected skin region (nose)
nose = double(nose)/255;
for m=2:3:20;
   for n=2:3:29;
rnose(m,n)=(nose(m,n)+nose(m+1,n)+nose(m,n+1)+nose(m+1,n+1)+nose(m-1,n-1)+nose(m,n-
1)+nose(m-1,n)+nose(m+1,n-1)+nose(m-1,n+1))/9;
   end
end
s2 = nonzeros(rnose);
```

```
nose_r = reshape(s2,7,10);


mouth = imcrop(face,[27 70 50 17]);              % Crop selected skin region (mouth)
mouth=double(mouth)/255;
for m=2:3:17;
   for n=2:3:50;
rmouth(m,n)=(mouth(m,n)+mouth(m+1,n)+mouth(m,n+1)+mouth(m+1,n+1)+mouth(m-1,n-
1)+mouth(m,n-1)+mouth(m-1,n)+mouth(m+1,n-1)+mouth(m-1,n+1))/9;
   end
end
s3 = nonzeros(rmouth);
mouth_r = reshape(s3,6,17);

r_eye_t = r_eye_t + r_eye_r;                     % summation of the 4 different right eye
l_eye_t = l_eye_t + l_eye_r;                     % summation of the 4 different left eye
nose_t = nose_t + nose_r;                        % summation of the 4 different nose
mouth_t = mouth_t + mouth_r;                     % summation of the 4 different mouth
end
avg_r_eye = r_eye_t/face_expression;             % aveage of right eyes of the subject
avg_l_eye = l_eye_t/face_expression;             % aveage of left eyes of the subject
avg_nose = nose_t/face_expression;               % aveage of nose of the subject
avg_mouth = mouth_t/face_expression;             % aveage of mouth of the subject

r_eye_v = reshape(avg_r_eye,[],1);
l_eye_v = reshape(avg_l_eye,[],1);
nose_v = reshape(avg_nose,[],1);
mouth_v = reshape(avg_mouth,[],1);

pattern=[r_eye_v; l_eye_v; nose_v; mouth_v];           % Vectorize the features
[row col] = size(pattern);
pattern1=pattern-0.5*ones(row,1);
PATTERNS = [PATTERNS pattern1];                  % combine vectors side-by-side to create a data matrices
end
PATTERNS
features_face_extraction_time = cputime – t             % calculate feature extraction time

disp(sprintf('WHAT IS THE TOLERANCE OF PROGRAM YOU NEED'));
TOLERANCE = input('IF LOW ENTER 1\nIF MEDIUM ENTER 2\nIF HIGH ENTER 3\nENTER
YOUR CHOICE : ' );
if TOLERANCE==1
         threshold=0.7;
elseif TOLERANCE==2
         threshold=0.8;
elseif TOLERANCE==3
         threshold=0.9;
 end

goalerr=0.001;                                   % error
maxiter=5000;                                    % Max iteration
ETA = 0.0495;                                    % Learning Rate
ALPHA = 0.41;                                    % Momentum Factor
% Desired Output %
TARGET = eye(15,15);                             % Desired output
PATTERN=15;                                      % Pattern numbers
a = -0.3; b = 0.3;
hidw = a + (b-a) * rand(65,row);                 % Selection the weights values between 0.3 and -0.3
outw = a + (b-a) * rand(15,65);
dhidw=0;                                         % Initiate the change of hidden weight as zero
```

```
doutw=0;                                    % Initiate the change of output weight as zero
hidb = a + (b-a) * rand(65,1);              % Selection the bias weights values between 0.3 and -0.3
outb = a + (b-a) * rand(15,1);
dhidb=0;                                    % Initiate the change of hidden bias weight as zero
doutb=0;                                    % Initiate the change of output bias weight as zero

PATTERNS;
TARGET;
  for k= 1 : PATTERN
out1(:,k) = PATTERNS(:,k);                  % Forward pass, compute outputs out1
neth = (hidw * out1(:,k))+hidb;
out2(:,k) = logsig( neth );                 % Forward pass, compute outputs out2
neto = (outw*out2(:,k))+outb;
out3(:,k) = logsig( neto );                 % Forward pass, compute outputs out3
out3(:,k);
end
e = TARGET - out3;                          % Calculate the error
error = 1/2*(mean(diag(e).*diag(e)));
iter=1;                                     % Initiate the iteration
t = cputime;                                % Initiate training time calculation

while  error >= goalerr & iter<maxiter      % Compare the error with goal error
   for k=1:PATTERN
dfout2 = dlogsig( neth , out2(:,k) );
dfout3 = dlogsig( neto , out3(:,k) );       % Calculate the signal error

dout = -2*diag(dfout3) * e(:,k);            % Adjustments at output layer
dhid = diag(dfout2)* outw'* dout;           % Adjustments at hidden layer

oldoutw = outw;
oldhidw = hidw;
oldoutb = outb;
oldhidb = hidb;

outw = outw - (1-ALPHA)*(ETA*dout*out2(:,k)') + ALPHA*doutw ;  % Update Weight of output layer
hidw = hidw - (1-ALPHA)*(ETA*dhid*out1(:,k)') + ALPHA*dhidw;  % Update Weight of hidden layer
outb = outb - (1-ALPHA)*(ETA*dout) + ALPHA*doutb ;       % Update bias Weight of output layer
hidb = hidb - (1-ALPHA)*(ETA*dhid) + ALPHA*dhidb;  % Update bias Weight of hidden layer

dhidw = hidw - oldhidw;
doutw = outw - oldoutw;
dhidb = hidb - oldhidb;
doutb = outb - oldoutb;

out1(:,k) = PATTERNS(:,k);
neth = (hidw * out1(:,k))+hidb;
out2(:,k) = logsig( neth );
neto = (outw*out2(:,k))+outb;
out3(:,k) = logsig( neto );
end

out3;
e = TARGET - out3;
error = 1/2*(mean(diag(e).*diag(e)));
disp(sprintf('ITERATION NO.%5d       MSE IS%12.6f%',iter,error));
mse(iter)=error;
iter=iter+1;
   end
   out3;
   out_out=diag(out3);
   thidw = hidw;
   toutw = outw;
   thidb = hidb;
   toutb = outb;
 training_time =cputime - t;                    % output training time
disp(sprintf('PROCESSING TIME IS %7.2f',training_time));
plot(mse,'k');
title('ERROR GRAPH');
xlabel('ITERATION');
ylabel('MEAN SQUARE ERROR');
```

%%%%%%%%RESULTS OF TRAINSET FACE IMAGES%%%%%%%%

```
PATTERNS = [PATTERNS pattern1];
for k=1:PATTERN
out1(:,k) = PATTERNS(:,k);                          % Calculate forward pass
neth = (hidw * out1(:,k))+hidb;
out2(:,k) = logsig( neth );
neto = (outw*out2(:,k))+outb;
out3(:,k) = logsig( neto );
end
out3;
out_out=diag(out3)
```

%%%%%%%%%RESULT OF TESTSET FACE IMAGES%%%%%%%%%

```
test_number = 30;
TEST_RESULTS=[];

cd( 'D:\TESTSET\')                    % change the directory to get the images in

for k = 1:test_number                 % Start for loop to read test faces
   test_image = ['test_subject' int2str(k) '.jpg'];

color_image = imread(test_image);
   %save rgb version, for display later

face1 = rgb2gray(color_image);

   r_eye_test = imcrop(face1,[10 18 29 14]);             % Crop selected skin region
r_eye_test=double(r_eye_test)/255;
for m=2:3:14;
   for n=2:3:29;
reyer_test(m,n)=(r_eye_test(m,n)+r_eye_test(m+1,n)+r_eye_test(m,n+1)+r_eye_test(m+1,n+1)+r_eye_te
st(m-1,n-1)+r_eye_test(m,n-1)+r_eye_test(m-1,n)+r_eye_test(m+1,n-1)+r_eye_test(m-1,n+1))/9;
   end
end
tt = nonzeros(reyer_test);
r_eye_tt = reshape(tt,5,10);

l_eye_test = imcrop(face1,[62 18 29 14]);
l_eye_test=double(l_eye_test)/255;
for m=2:3:14;
   for n=2:3:29;
leyer_test(m,n)=(l_eye_test(m,n)+l_eye_test(m+1,n)+l_eye_test(m,n+1)+l_eye_test(m+1,n+1)+l_eye_tes
t(m-1,n-1)+l_eye_test(m,n-1)+l_eye_test(m-1,n)+l_eye_test(m+1,n-1)+l_eye_test(m-1,n+1))/9;
   end
end
tt1 = nonzeros(leyer_test);
```

```
l_eye_tt = reshape(tt1,5,10);

nose_test = imcrop(face1,[38 38 29 20]);
nose_test=double(nose_test)/255;
for m=2:3:20;
   for n=2:3:29;
rnose_test(m,n)=(nose_test(m,n)+nose_test(m+1,n)+nose_test(m,n+1)+nose_test(m+1,n+1)+nose_test(m-
1,n-1)+nose_test(m,n-1)+nose_test(m-1,n)+nose_test(m+1,n-1)+nose_test(m-1,n+1))/9;
   end
end
tt2 = nonzeros(rnose_test);
nose_tt = reshape(tt2,7,10);

mouth_test = imcrop(face1,[27 70 50 17]);
mouth_test=double(mouth_test)/255;
for m=2:3:17;
   for n=2:3:50;
rmouth_test(m,n)=(mouth_test(m,n)+mouth_test(m+1,n)+mouth_test(m,n+1)+mouth_test(m+1,n+1)+mo
uth_test(m-1,n-1)+mouth_test(m,n-1)+mouth_test(m-1,n)+mouth_test(m+1,n-1)+mouth_test(m-
1,n+1))/9;
   end
end
tt3 = nonzeros(rmouth_test);
mouth_tt = reshape(tt3,6,17);

r_eye_v_t = reshape(r_eye_tt,[],1);
l_eye_v_t = reshape(l_eye_tt,[],1);
nose_v_t = reshape(nose_tt,[],1);
mouth_v_t = reshape(mouth_tt,[],1);

pattern2=[r_eye_v_t; l_eye_v_t; nose_v_t; mouth_v_t];
pattern3=pattern2-0.5*ones(row,1);

t = cputime;                    % Initiate test time calculation

test_out1 = pattern3;           % Calculate the forward pass for test images
test_neth = (thidw * test_out1)+thidb;
test_out2 = logsig( test_neth );
test_neto = (toutw * test_out2)+toutb;
test_out3 = logsig( test_neto )

  for i=1:PATTERN
    if test_out3(i,:)>=threshold & i == 1
      disp(sprintf('THE PERSON IS   ABED'));
        disp(sprintf('THE PERCENTAGE OF RECOGNITION IS %4.2f',test_out3(i,:)*100));

    elseif test_out3(i,:)>=threshold & i == 2
      disp(sprintf('THE PERSON IS   JEHAD'));
        disp(sprintf('THE PERCENTAGE OF RECOGNITION IS %4.2f',test_out3(i,:)*100));

    elseif test_out3(i,:)>=threshold & i == 3
      disp(sprintf('THE PERSON IS   ABUELFADEL'));
        disp(sprintf('THE PERCENTAGE OF RECOGNITION IS %4.2f',test_out3(i,:)*100));

    elseif test_out3(i,:)>=threshold & i == 4
      disp(sprintf('THE PERSON IS   MAJED'));
        disp(sprintf('THE PERCENTAGE OF RECOGNITION IS %4.2f',test_out3(i,:)*100));

    elseif test_out3(i,:)>=threshold & i == 5
```

```
disp(sprintf('THE PERSON IS   ABUKAREM'));
    disp(sprintf('THE PERCENTAGE OF RECOGNITION IS %4.2f',test_out3(i,:)*100));

elseif test_out3(i,:)>=threshold & i == 6
    disp(sprintf('THE PERSON IS   EYAD'));
        disp(sprintf('THE PERCENTAGE OF RECOGNITION IS %4.2f',test_out3(i,:)*100));

elseif test_out3(i,:)>=threshold & i == 7
    disp(sprintf('THE PERSON IS   MOHAMMED'));
        disp(sprintf('THE PERCENTAGE OF RECOGNITION IS %4.2f',test_out3(i,:)*100));

elseif test_out3(i,:)>=threshold & i == 8
    disp(sprintf('THE PERSON IS   ABUJAZAR'));
        disp(sprintf('THE PERCENTAGE OF RECOGNITION IS %4.2f',test_out3(i,:)*100));

elseif test_out3(i,:)>=threshold & i == 9
    disp(sprintf('THE PERSON IS   FEHMI'));
        disp(sprintf('THE PERCENTAGE OF RECOGNITION IS %4.2f',test_out3(i,:)*100));

elseif test_out3(i,:)>=threshold & i == 10
    disp(sprintf('THE PERSON IS   DAWOD'));
        disp(sprintf('THE PERCENTAGE OF RECOGNITION IS %4.2f',test_out3(i,:)*100));

elseif test_out3(i,:)>=threshold & i == 11
    disp(sprintf('THE PERSON IS   MAJDY'));
        disp(sprintf('THE PERCENTAGE OF RECOGNITION IS %4.2f',test_out3(i,:)*100));

elseif test_out3(i,:)>=threshold & i == 12
    disp(sprintf('THE PERSON IS   IBRAHIM'));
        disp(sprintf('THE PERCENTAGE OF RECOGNITION IS %4.2f',test_out3(i,:)*100));

elseif test_out3(i,:)>=threshold & i == 13
    disp(sprintf('THE PERSON IS   SAMEH'));
        disp(sprintf('THE PERCENTAGE OF RECOGNITION IS %4.2f',test_out3(i,:)*100));

elseif test_out3(i,:)>=threshold & i == 14
    disp(sprintf('THE PERSON IS   SHADY'));
        disp(sprintf('THE PERCENTAGE OF RECOGNITION IS %4.2f',test_out3(i,:)*100));

elseif test_out3(i,:)>=threshold & i == 15
    disp(sprintf('THE PERSON IS   AFEEF'));
        disp(sprintf('THE PERCENTAGE OF RECOGNITION IS %4.2f',test_out3(i,:)*100));
    end
end
    if test_out3(1:PATTERN,:)<threshold
                disp(sprintf('THE PERSON IS   UNRECOGNIZED'));
            end
  TEST_RESULTS=[TEST_RESULTS test_out3];
end
TEST_RESULTS;
test_result=diag(TEST_RESULTS)
test_time =cputime – t
```

Appendix II Database

Person 1

Person 2

Person 3

Person 4

Person 5

Person 6

Person 7

Person 8

Person 9

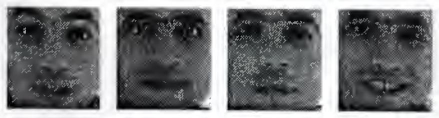
Person 10

Person 11

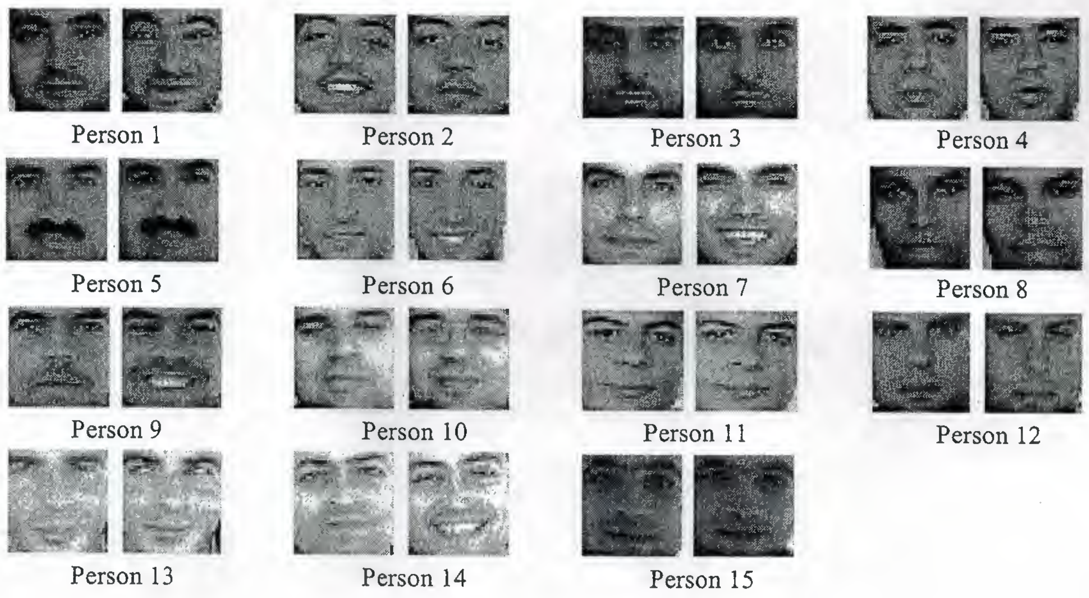
Person 12

Person 13

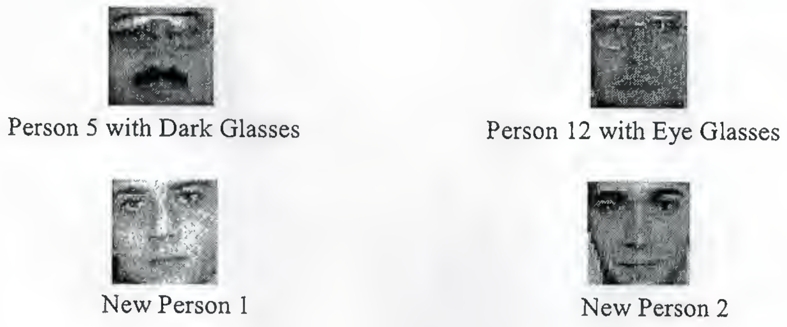
Person 14


Person 15

Training Set Database



Test Set Database



Other Test Set Database