Face Recognition Using Artificial Neural Networks

Thesis · March 2004

CITATION
1

READS
7,096

2 authors, including:

S M Kamrul Hasan
RPM Global
18 PUBLICATIONS **246** CITATIONS

SEE PROFILE

FACE RECOGNITION USING ARTIFICIAL NEURAL NETWORKS

Prepared By

1. *S. M. Kamrul Hasan*
   Department of Computer Science & Engineering
   Rajshahi University of Engineering & Technology (RUET)

2. *Tanvir Ahmed Chowdhury*
   Department of Computer Science & Engineering
   Rajshahi University of Engineering & Technology (RUET)

ABSTRACT

This paper represents the development of a system which can identify the person with the help of a face using artificial neural network technique. This system is implemented in two stages. They are the learning stage and the testing stage. Image acquisition, preprocessing, image filtering, feature extraction and learning are included in the learning stage. At first the system takes the image of a person. The input image is then converted to a gray scale image and the position of the face is detected from the image after highpass filtering and edge detection. The features, gray levels of the image are extracted which can be represented as a matrix and this feature matrix is given as input for the Kohonen self organizing map and fed to this network. The unsupervised learning network is trained and creates a knowledge base for future use.

In the testing stage the system takes the face of the image of a person for recognition. Image acquisition, preprocessing, image filtering, feature extraction are similar to the learning stage. For classification the features are fed to the network. The network will classify the face image from the knowledge base and recognizes it.

ACKNOWLEDGEMENT

First of all, I would like to thank my supervisor, ***Md. Nazrul Islam Mondal***, Lecturer, Department of Computer Science & Engineering, Rajshahi University of Engineering & Technology (RUET), Rajshahi, for his proper guidance, suggestions, co-operations, constant encouragement and supervision at all stages throughout this study.

We are grateful to our respective teacher ***Kaushik Roy***, Lecturer Department of Computer Science & Engineering, Rajshahi University of Engineering & Technology, Rajshahi. We are much benefited from his image processing lectures which are very useful for the first step of our research work. Thanks to him for providing us a strong knowledge about image processing.

Special thanks to our teacher ***A. S. M. Sohail, Lecturer***, Department of Computer Science & Engineering, Rajshahi University of Engineering & Technology, Rajshahi, for his valuable suggestion about Artificial Neural Network which is the main part of our research. His neural network lectures inspired us for better development and made us doubtless about our success.

A great deal of thanks goes to our honorable teacher ***Dr. Md. Mortuza Ali***, Professor and Head of the Department, Department of Computer Science & Engineering, Rajshahi University of Engineering & Technology, Rajshahi, for his endless support. We are grateful to him for providing us an unlimited facility of Internet and departmental laboratories.

We are very obliged to the laboratory staff of Computer Science & Engineering Department, RUET, for their kind co-operation and assistance throughout this study.

Above all, we bow down to the almighty Allah for the successful completion of the research.

RUET, Rajshahi
November 8, 2013 ***Authors***

List of Contents

List of Tables

List of Figures

Chapter 1 : Introduction

1.1 Background

The human ability to recognize faces is quite remarkable and to infer intelligence or character from facial appearance is suspect. The face is the focus of first attention to recognize a person. We can recognize thousands of faces learned throughout our lifetime and recognizing familiar faces at a glance even after years of separation. But this skill is not useful despite if there is a change in the faces due to viewing conditions, expression, aging, and distractions such as glasses, beards or changes in hair style.

Face recognition has become an important issue in many applications such as security systems, credit card verification and criminal identification. Although it is clear that people are good at face recognition, it is not at all obvious how faces are encoded or decoded by the human brain.

Researchers are working for a long time to develop a computer system which can act as a human being. The system will perform visual sensing like identifying objects such home, office, motor car, faces of people, handwriting and printed words and aural sensing like voices recognition. But the computers can perform repetitive actions like calculating the sum of few hundred digit numbers and poorer at the task of processing the vast quantities of different types of data a vision system requires. Instead of these difficulties the researchers are still working for systems that are artificially intelligent in any general sense that would recognize.

Human face recognition by computer system has been studied for more than twenty years. Unfortunately developing a computational model of face recognition is quite difficult, because faces are complex, multi-dimensional visual stimuli. Therefore, face recognition is a very high level computer vision task, in which many early vision techniques can be involved. So, we develop an automatic system to find neutral faces in images for face recognition. A neutral face is a relaxed face without contraction of facial muscles and facial movements. It is the states of a person’s face most of the time, i.e. it is the facial appearance without any expression.

1.2 Research About Face Recognition

The subject of face recognition is as old as computer vision, both because of the practical importance of the topic and theoretical interest from cognitive scientists. Despite the fact that other methods of identification (such as fingerprints, or iris scans) can be more accurate, face recognition has always remains a major focus of research because of its non-invasive nature and because it is people's primary method of person identification. Much of the work in computer recognition of faces has focused on detecting individual features such as the eyes, nose, mouth, and head outline, and defining a face model by the position, size, and relationships among these features. Such approaches have proven difficult to extend to multiple views and have often been quite fragile, requiring a good initial guess to guide them. Research in human strategies of face recognition, moreover, has shown that individual features and their immediate relationships comprise an insufficient representation to account for the performance of adult human face identification [1]. Nonetheless, this approach to face recognition remains the most popular one in the computer vision literature.

Bledsoe [2], [3] was the first to attempt semi-automated face recognition with a hybrid human-computer system that classified faces on the basis of fiducially marks entered on photographs by hand. Parameters for the classification were normalized distances and ratios among points such as eye corners, mouth corners, nose tip, and chin point. Later work at Bell Labs developed a vector of up to 21 features, and recognized faces using standard pattern classification techniques.

Fischler and Elschlager[4] attempted to measure similar features automatically. They described a linear embedding algorithm that used local feature template matching and a global measure of fit to find and measure facial features. This template matching approach has been continued and improved by the recent work of Yuille and Cohen [5]. Their strategy is based on deformable templates, which are parameterized models of the face and its features in which the parameter values are determined by interactions with the face image.

Connectionist approaches to face identification seek to capture the configurationally nature of the task. Kohonen [6] and Kohonen and Lehtio [7] describe an associative network with a simple learning algorithm that can recognize face images and recall a face image from an incomplete or noisy version input to the network. Fleming and Cottrell [8] extend these ideas using nonlinear units, training the system by backpropagation.

Others have approached automated face recognition by characterizing a face by a set of geometric parameters and performing pattern recognition based on the parameters. Kanade's [9] face identification system was the first system in which all steps of the recognition process were automated, using a top-down control strategy directed by a generic model of expected feature characteristics. His system calculated a set of facial parameters from a single face image and used a pattern classification technique to match the face from a known set, a purely statistical approach depending primarily on local histogram analysis and absolute gray-scale values.

Recent work by Burt [10] uses a smart sensing approach based on multi resolution template matching. This coarse to fine strategy uses a special purpose computer built to calculate multi resolution pyramid images quickly, and has been demonstrated identifying people in near real time.

1.3 Commercial Systems for Face Recognition

Currently, several face-recognition products are commercially available. Algorithms developed by the top contenders of the FacE REcognition Technology (FERET) competition are the basis of some of the available systems; others were developed outside of the FERET testing framework. While it is extremely difficult to judge, three systems -- Visionics, Viisage, and Miros -- seem to be the current market leaders in face recognition.

Visionics' FaceIt face recognition software is based on the Local Feature Analysis algorithm developed at Rockefeller University. FaceIt is now being incorporated into a Close Circuit Television (CCTV) anti-crime system called `Mandrake' in United Kingdom. This system searches for known criminals in video acquired from 144 CCTV camera locations. When a match occurs a security officer in the control room is notified [11].

Viisage, another leading face-recognition company, uses the eigenface-based recognition algorithm developed at the MIT Media Laboratory. Their system is used in conjunction with identification cards (e.g., driver's licenses and similar government ID cards) in many US states and several developing nations [11].

Miros uses neural network technology for their TrueFace face recognition software. TrueFace is used by Mr. Payroll for their check cashing system, and has been deployed at casinos and similar sites in many US states [11].

1.4 Application of Face Recognition System

Person Identification System has a broad range of applications. Examples include:

1.4.7 Automatic Face Recognition

Automatic face recognition has already been used in

- Personal identification (credit cards, driver's license, passports, employee ID)
- Access control, such as the access to check-cashing ATMs, buildings or rooms).

1.4.8 Video Coding, Video Databases and Teleconferencing System

It is well known that people are most sensitive to coding errors in facial features. The coder would

- Encode very precisely facial features (such as eyes, mouth, nose, etc.)
- Encode less precisely the rest of the picture.

1.4.9 Human-Computer Interaction

Currently some computer games can be played with head movement, instead of mouse or keyboard (but the player must wear headgear).

1.4.10 Security Monitoring

A face tracking system can be used as a security system in shopping malls, other public areas, or private houses.

1.4.11 Banking System

In banking system, human face detection system is much used phenomenon.

1.4.12 Information Retrieval

For information retrieval system human face detection is very important things.

1.5 Objectives

The objective of our face recognition systems may depend on the application of the system. We can identify at least two broad categories of face recognition systems:

- We want to find a person within a large database of faces (e.g. in a police database). These systems typically return a list of the most likely people in the database. Often only one image is available per person. It is usually not necessary for recognition to be done in real-time.
- We want to identify particular people in real-time (e.g. in a security monitoring system, banking system, etc.), or we want to allow access to a group of people and deny access to all others (e.g. access to a

building, computer, etc.). Multiple images per person are often available for training and real-time recognition is required.

In this paper, we are primarily interested in the second case. We are interested in recognition with varying facial detail, expression, pose, etc. We do not consider invariance to high degrees of rotation or scaling - we assume that a minimal preprocessing stage is available if required.

1.7 Outline of a Face Recognition System

In face recognition system, it needs to learn the machine about the facial image of the human being which the machine can recognize further. Any facial image is learnt in some pre-defined ways and stored in a knowledge base. Then while identify the face, previous stored values are used. Thus the system contains two phases –

1. Learning Phase
2. Recognizing Phase

1.6.1 Learning Process

The first task is to learn the machine about the feature vectors of the input pattern. We followed six basic steps to learn the machine about any pattern:

1. Acquiring an image
2. Pre – Processing
3. Filtering the image
4. Feature extraction
5. Training
6. Save the pattern to knowledge base

We wish to acquire the image pattern using two input devices i.e. digital camera and scanner. Digital camera provides live image of an object where scanner provides pattern of a previously stored image.

Pre-processing step includes the conversion of an image from color to grayscale and scale the image to a standard shape.

Filtering is essential to remove the unexpected features from the input image and detect the actual features from the image.

Feature extraction is to detect the actual features from the entire image and the features are represented as a two dimensional matrix. Then the extracted features matrix known as input pattern will be passed through a designed neural network for training. The output of the network is saved in a knowledge base. The learning process is shown in figure (a)

1.6.2 Testing Process

Six basic steps needed to test the pattern if it is recognized or not. These are:

1. Acquiring an image
2. Pre – Processing
3. Filtering the image
4. Feature extraction
5. Classification
6. Recognition

Among the six steps the first four steps are similar to the learning process. The facial image of the person is taken and processed what we did in the learn process.

In classification stage the given features are fed to the network and the network will classify the image from its knowledge base. The testing process is shown in figure 1.1 (b).

1.6.3 Block Diagram of Learning Process

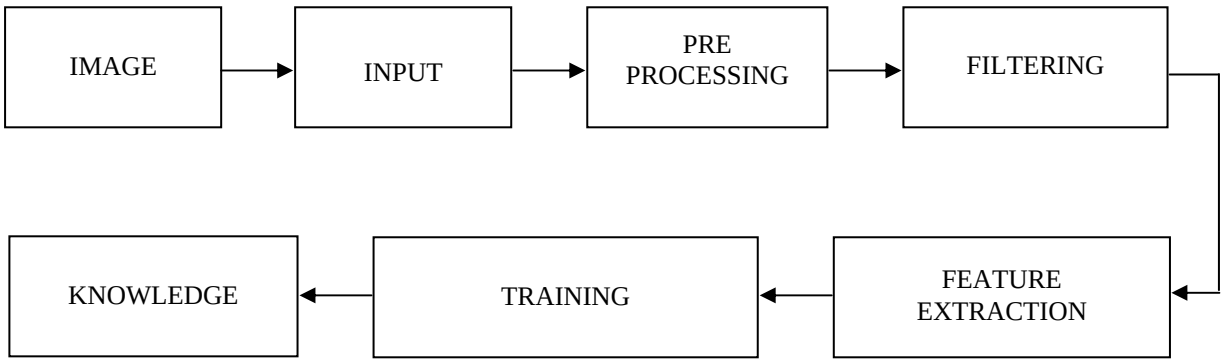


Fig 1.1 (a): Learning Process

1.6.4 Block Diagram of Testing Process

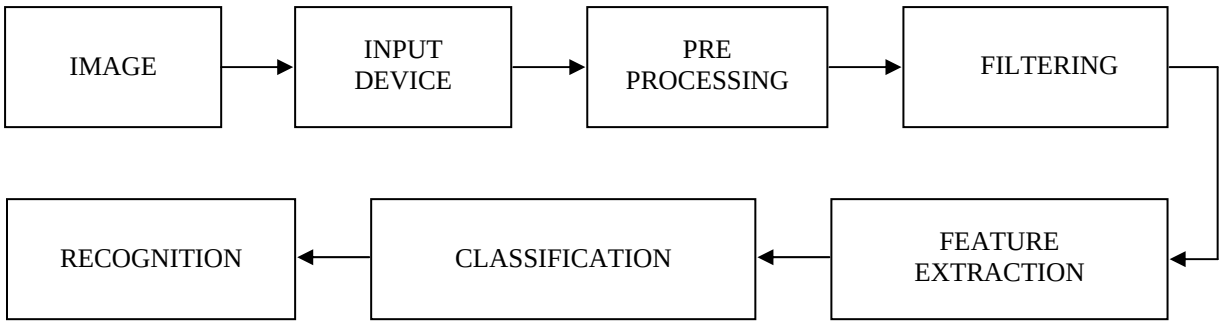


Fig 1.1 (b): Recognizing Process

Chapter 2 : Fundamentals of Image Processing

2.1 Fundamental of Digital Image

The term *monochrome image* or simply *image,* refers to a two-dimensional light intensity function $f(x, y)$, where x and y denote spatial coordinates and the value of f at any point (x, y) is proportional to the brightness (or *gray level)* of the image at that point.

A digital image is an image *f(x, y)* that has been discretized both in spatial coordinates and brightness. A digital image can be considered a matrix whose row and column indices identify a point in the image and the corresponding matrix element value identifies the gray level at that point. The elements of such a digital array are called *image elements, picture elements, pixels,* or *pels,* with the last two being commonly used abbreviations of "picture elements".

2.2 Representation of a Digital Image

To be suitable for computer processing, an image function $f(x, y)$ must be digitized both spatially and in amplitude. Digitization of the spatial coordinates (x, y) is called *image sampling,* and amplitude digitization is called *gray-level quantization.*

Suppose that a continuous image $f(x, y)$ is approximated by equally spaced samples arranged in the form of an $N \times M$ array as shown in the Eq. (2.2-1), where each elements of the array is discrete quantity:

$$f(x, y) = \begin{bmatrix} f(0,0) & f(0,1) & . & f(0, M-1) \\ f(1,0) & f(1,1) & . & f(1, M-1) \\ . & . & . & \\ f(N-1,0) & f(N-1,1) & & f(N-1, M-1) \end{bmatrix} \quad \textbf{Eq. (2.2-1)}$$

2.3 Image Acquisition

Basically the operation of the *image acquisition* is fully depends on the different types of input device (TWAIN drivers) such as digital camera, scanner, digitizer etc. But the most different devices for this purpose are an image scanner or digital camera. Practically both scanner and digital camera are used in this research. Scanner – Cannon 300F (2400 dpi) and digital camera - Prolink dual mode USB camera. During the scanning of images by scanner, the dpi is set to 100 and while acquiring by camera, resolution is set to 320×240. Output extension of the file is JPEG (*jpg, *jpeg) file format. JPEG file contains sufficient information about a color face. It is a compressed file format and enough for the research. Scanning process is simple enough that converts an image to a digital picture. Due to the conversion, image becomes available in the computer. Saved image in the disk in JPEG format can be easily be converted into a binary format. Sequential steps are required for the process of image acquisition. Those are mentioned below-

1. Select the appropriate facial photograph which is to be scanned. It is recommended to use such a picture where face is highlighted.

2. Place the photograph inside the scanner any where on the glass tray. Shut the cover of the scanner down.

3. Set the appropriate parameters of the scanner for better performance of the learning and recognition process. That means the dpi, resolution, color mode etc.

4. Operating by the driver, image is transferred to the computer. Save the image to its associated name.

And the alternate provisions to acquire image through digital camera. The steps to acquire by this are follows:

1. Power ON the camera and check the status of camera. Set the resolution of the camera to 320×240 for better performance.
2. Adjust the light of room. It is preferable to use flash while acquiring photos indoor. But is better to use it under daylight.
3. Set the face of the human at the center position of the viewing window and check on the shutter.
4. Connect the camera to any computer using Universal Serial Bus (USB) data to the USB port.
5. Check the driver status of the camera and use the download option from the driver. Files will be saved to the associated directory and rename the file to appropriate name and format.

2.4 Image Scaling

Image Scaling option includes *zooming* (enlarging) and *shrinking* (reducing) images. Image shrinking is useful for saving disk space. Fitting a large image into a small display and pasting several images into one image of the same size.

Generally two techniques for image zooming: *replication* and *interpolation*. Figure 2.1 (a) and (b) shows these techniques for a 2x enlargement. Image can be enlarged by repeating a single pixel several times in both directions. The replication method uses the simplest technique; it just copies the pixel. This is easy to do, but the enlarged image jagged.

The interpolation technique is harder to implement, but the result is smoother. Note in Figure 2.1 how the 1 in the upper left corner of original enlarges in the two techniques. Replication copies it, but interpolation smoothes the transition from 1 to 4, 6 and 9. The extra pixel values are half way between the 1 and the values of its neighbors. The 9 in the original image does not change in the final result. This will always be the case for the pixels on the bottom row and far right column of the image.

Input Image	Replication (2x)	Interpolation (2x)	Input Image
1 4	1 1 4 4	1 2 4 4	1 3 5 7
4 9	1 1 4 4	3 5 6 6	9 11 13 15
	6 6 9 9	6 7 9 9	17 19 21 23
	6 6 9 9	6 7 9 9	25 27 29 31

Corner Shrink	Average Shrink	Median Shrink
1 5	6 10	3 7
17 21	22 26	19 23

Figure 2.1 : Various Scaling of Images

2.5 Image Filtering

2.5.3 Spatial Filtering

The use of spatial masks for image processing usually is called *spatial filtering* (as opposed to *frequency domain filtering* using the Fourier transform), and the masks themselves are called *spatial filters.* In this section we consider linear and nonlinear spatial filters for image enhancement.

Linear filters state that the transfer function and the impulse or point spread function of a linear system are inverse Fourier transforms of each other) So-called *lowpass* filters attenuate or eliminate high-frequency components in the Fourier domain while leaving low frequencies untouched (that is, the filter "passes" low frequencies). High-frequency components characterize edges and other sharp details in an image, so the net effect of *lowpass* filtering is image blurring. Similarly, highpass filters attenuate or eliminate low-frequency components. Because these components are responsible for the slowly varying characteristics of an image, such as overall contrast and average intensity, the net result of highpass filtering is a reduction of these features and a correspondingly apparent sharpening of edges and other sharp details. A third type of filtering, called *bandpass filtering,* removes selected frequency regions between low and high frequencies. These filters are used for image restoration and are seldom of interest in image enhancement.

Figure 2.2 shows cross sections of circularly symmetric *lowpass, highpass,* and *bandpass* filters in the frequency domain and their corresponding spatial filters. The horizontal axes for the figures in the top row correspond to frequency, and their counterparts in the bottom row are spatial coordinates. The shapes in the bottom row are used as guidelines for specifying linear spatial filters. Regardless of the type of linear filter used, however, the basic approach is to sum products between the mask coefficients and the intensities of the pixels under the mask at a specific location in the image.

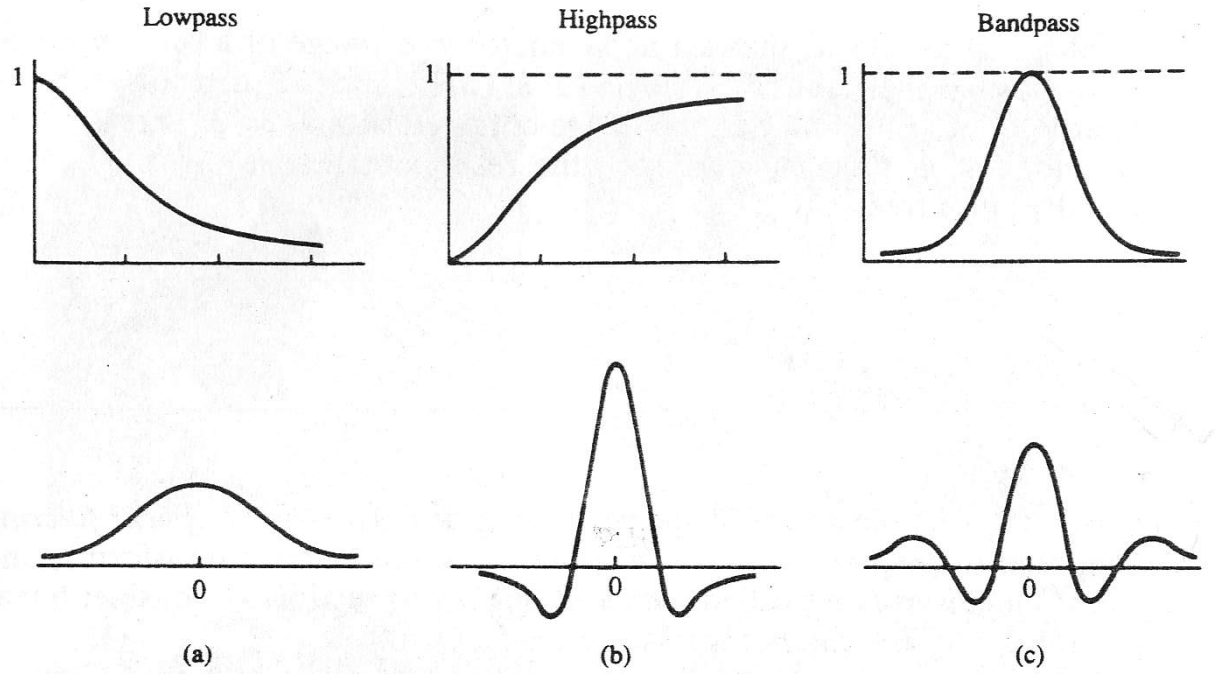


Figure 2.2 : Lowpass, Highpass and Bandpass Filters

Figure 2.3 shows a general 3×3 mask. Denoting the gray levels of pixels under the mask at any location by $z_1, z_2, z_3.....z_9$, the response of a linear mask is

$$R = w_1z_1 + w_2z_2 + ... + w_9z_9 \qquad \text{Eq. (2.5-1)}$$

With reference to Fig. 2.3, if the center of the mask is at location (x, y) in the image, the gray level of the pixel located at (x, y) is replaced by R .

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Figure 2.3 : Mask of a Spatial Filter

The mask is then moved to next pixel location in the image and the process is repeated. This continues until all pixel locations have been covered. The value of R is computed by using partial neighborhoods. The value of R is computed by using partial neighborhoods for pixels that are located in the border of the image. Also, usual practice is to create a new image to store the values of R, instead of changing pixel values in place. This practice avoids using gray levels in Eq. (2.5.-1) that have been altered as a result of an earlier application of this equation.

Nonlinear spatial filters also operate on neighborhoods. In general, however, their operation is based directly on the values of the pixels in the neighborhood under consideration, and they do not explicitly use coefficient in the manner described in Eq. (2.5-1).

2.5.4 Median filtering

Smoothing filtering method blurs edges and other sharp details. If the objective is to achieve noise reduction rather than blurring, an alternative approach is to use *median filters*. That is, the gray level of each pixel is replaced by the median of the gray levels in a neighborhood of that pixel, instead of by the average. This method is particularly effective when the noise pattern consists of strong, spike like components and the characteristic to be preserved is edge sharpness.

The median m of a set of values is such that half the values in the set are less than m and half are greater than m. In order to perform median filtering in a neighborhood of a pixel, we first sort the values of the pixel and its neighbors, determine the median, and assign this value to the pixel. For example, in a 3×3 neighborhood the median is the 5th largest value, in a 5×5 neighborhood the 13th largest value, and so on. When several values in a neighborhood are the same, all equal values have to be grouped. For example, suppose that a 3×3 neighborhood has values (10, 20, 20, 20, 15, 20, 20, 25, 100). These values are sorted as (10, 15, 20, 20, 20, 20, 20, 25, 100), which results in a median of 20. Thus the principal function of median filtering is to force points with distinct intensities to be more like their neighbors; actually eliminating intensity spikes that appear isolated in the area of the filter mask:

2.6 Fundamental of Digital Image

The term *monochrome image* or simply *image,* refers to a two-dimensional light intensity function $f(x, y)$, where x and y denote spatial coordinates and the value of f at any point (x, y) is proportional to the brightness (or *gray level)* of the image at that point.

A digital image is an image *f(x, y)* that has been discretized both in spatial coordinates and brightness. A digital image can be considered a matrix whose row and column indices identify a point in the image and the corresponding matrix element value identifies the gray level at that point. The elements of such a digital array are called *image elements, picture elements, pixels,* or *pels,* with the last two being commonly used abbreviations of "picture elements".

2.7 Representation of a Digital Image

To be suitable for computer processing, an image function $f(x, y)$ must be digitized both spatially and in amplitude. Digitization of the spatial coordinates (x, y) is called *image sampling,* and amplitude digitization is called *gray-level quantization.*

Suppose that a continuous image $f(x, y)$ is approximated by equally spaced samples arranged in the form of an $N \times M$ array as shown in the Eq. (2.2-1), where each elements of the array is discrete quantity:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & . & f(0,M-1) \\ f(1,0) & f(1,1) & . & f(1,M-1) \\ . & . & . & \\ f(N-1,0) & f(N-1,1) & & f(N-1,M-1) \end{bmatrix} \quad \textbf{Eq. (2.2-1)}$$

2.8 Image Acquisition

Basically the operation of the *image acquisition* is fully depends on the different types of input device (TWAIN drivers) such as digital camera, scanner, digitizer etc. But the most different devices for this purpose are an image scanner or digital camera. Practically both scanner and digital camera are used in this research. Scanner – Cannon 300F (2400 dpi) and digital camera - Prolink dual mode USB camera. During the scanning of images by scanner, the dpi is set to 100 and while acquiring by camera, resolution is set to 320×240. Output extension of the file is JPEG (*jpg, *jpeg) file format. JPEG file contains sufficient information about a color face. It is a compressed file format and enough for the research. Scanning process is simple enough that converts an image to a digital picture. Due to the conversion, image becomes available in the computer. Saved image in the disk in JPEG format can be easily be converted into a binary format. Sequential steps are required for the process of image acquisition. Those are mentioned below-

1. Select the appropriate facial photograph which is to be scanned. It is recommended to use such a picture where face is highlighted.

2. Place the photograph inside the scanner any where on the glass tray. Shut the cover of the scanner down.

3. Set the appropriate parameters of the scanner for better performance of the learning and recognition process. That means the dpi, resolution, color mode etc.

4. Operating by the driver, image is transferred to the computer. Save the image to its associated name.

And the alternate provisions to acquire image through digital camera. The steps to acquire by this are follows:

6. Power ON the camera and check the status of camera. Set the resolution of the camera to 320×240 for better performance.
7. Adjust the light of room. It is preferable to use flash while acquiring photos indoor. But is better to use it under daylight.
8. Set the face of the human at the center position of the viewing window and check on the shutter.
9. Connect the camera to any computer using Universal Serial Bus (USB) data to the USB port.
10. Check the driver status of the camera and use the download option from the driver. Files will be saved to the associated directory and rename the file to appropriate name and format.

2.9 Image Scaling

Image Scaling option includes *zooming* (enlarging) and *shrinking* (reducing) images. Image shrinking is useful for saving disk space. Fitting a large image into a small display and pasting several images into one image of the same size.

Generally two techniques for image zooming: *replication* and *interpolation.* Figure 2.1 (a) and (b) shows these techniques for a 2x enlargement. Image can be enlarged by repeating a single pixel several times in both directions. The replication method uses the simplest technique; it just copies the pixel. This is easy to do, but the enlarged image jagged.

The interpolation technique is harder to implement, but the result is smoother. Note in Figure 2.1 how the 1 in the upper left corner of original enlarges in the two techniques. Replication copies it, but interpolation smoothes the transition from 1 to 4, 6 and 9. The extra pixel values are half way between the 1 and the values of its neighbors. The 9 in the original image does not change in the final result. This will always be the case for the pixels on the bottom row and far right column of the image.

Input Image	Replication (2x)	Interpolation (2x)	Input Image
1 4	1 1 4 4	1 2 4 4	1 3 5 7
4 9	1 1 4 4	3 5 6 6	9 11 13 15
	6 6 9 9	6 7 9 9	17 19 21 23
	6 6 9 9	6 7 9 9	25 27 29 31

Corner Shrink	Average Shrink	Median Shrink
1 5	6 10	3 7
17 21	22 26	19 23

Figure 2.1 : Various Scaling of Images

2.10 Image Filtering

2.5.5 Spatial Filtering

The use of spatial masks for image processing usually is called *spatial filtering* (as opposed to *frequency domain filtering* using the Fourier transform), and the masks themselves are called *spatial filters.* In this section we consider linear and nonlinear spatial filters for image enhancement.

Linear filters state that the transfer function and the impulse or point spread function of a linear system are inverse Fourier transforms of each other) So-called *lowpass* filters attenuate or eliminate high-frequency components in the Fourier domain while leaving low frequencies untouched (that is, the filter "passes" low frequencies). High-frequency components characterize edges and other sharp details in an image, so the net effect of *lowpass* filtering is image blurring. Similarly, highpass filters attenuate or eliminate low-frequency components. Because these components are responsible for the slowly varying characteristics of an image, such as overall contrast and average intensity, the net result of highpass filtering is a reduction of these features and a correspondingly apparent sharpening of edges and other sharp details. A third type of filtering, called *bandpass filtering,* removes selected frequency regions between low and high frequencies. These filters are used for image restoration and are seldom of interest in image enhancement.

Figure 2.2 shows cross sections of circularly symmetric *lowpass, highpass,* and *bandpass* filters in the frequency domain and their corresponding spatial filters. The horizontal axes for the figures in the top row correspond to frequency, and their counterparts in the bottom row are spatial coordinates. The shapes in the bottom row are used as guidelines for specifying linear spatial filters. Regardless of the type of linear filter used, however, the basic approach is to sum products between the mask coefficients and the intensities of the pixels under the mask at a specific location in the image.

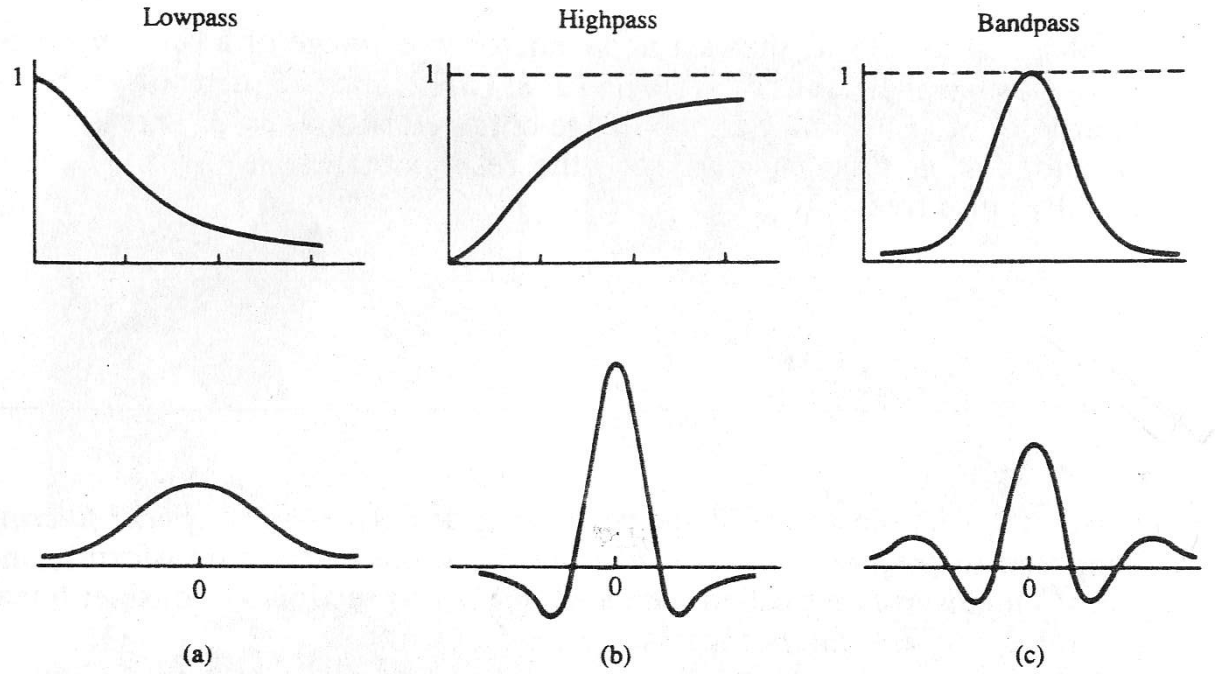


Figure 2.2 : Lowpass, Highpass and Bandpass Filters

Figure 2.3 shows a general 3×3 mask. Denoting the gray levels of pixels under the mask at any location by $z_1, z_2, z_3.....z_9$, the response of a linear mask is

$$R = w_1z_1 + w_2z_2 + ... + w_9z_9 \qquad \text{Eq. (2.5-1)}$$

With reference to Fig. 2.3, if the center of the mask is at location (x, y) in the image, the gray level of the pixel located at (x, y) is replaced by R .

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Figure 2.3 : Mask of a Spatial Filter

The mask is then moved to next pixel location in the image and the process is repeated. This continues until all pixel locations have been covered. The value of R is computed by using partial neighborhoods. The value of R is computed by using partial neighborhoods for pixels that are located in the border of the image. Also, usual practice is to create a new image to store the values of R, instead of changing pixel values in place. This practice avoids using gray levels in Eq. (2.5.-1) that have been altered as a result of an earlier application of this equation.

Nonlinear spatial filters also operate on neighborhoods. In general, however, their operation is based directly on the values of the pixels in the neighborhood under consideration, and they do not explicitly use coefficient in the manner described in Eq. (2.5-1).

2.5.6 Median filtering

Smoothing filtering method blurs edges and other sharp details. If the objective is to achieve noise reduction rather than blurring, an alternative approach is to use *median filters*. That is, the gray level of each pixel is replaced by the median of the gray levels in a neighborhood of that pixel, instead of by the average. This method is particularly effective when the noise pattern consists of strong, spike like components and the characteristic to be preserved is edge sharpness.

The median m of a set of values is such that half the values in the set are less than m and half are greater than m. In order to perform median filtering in a neighborhood of a pixel, we first sort the values of the pixel and its neighbors, determine the median, and assign this value to the pixel. For example, in a 3×3 neighborhood the median is the 5th largest value, in a 5×5 neighborhood the 13th largest value, and so on. When several values in a neighborhood are the same, all equal values have to be grouped. For example, suppose that a 3×3 neighborhood has values (10, 20, 20, 20, 15, 20, 20, 25, 100). These values are sorted as (10, 15, 20, 20, 20, 20, 20, 25, 100), which results in a median of 20. Thus the principal function of median filtering is to force points with distinct intensities to be more like their neighbors; actually eliminating intensity spikes that appear isolated in the area of the filter mask:

Chapter 3 : Artificial Neural Network

3.1 What are Artificial Neural Networks?

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANNs as well [13].

Artificial Neural Networks are being touted as the wave of the future in computing. They are indeed self learning mechanisms which don't require the traditional skills of a programmer. But unfortunately, misconceptions have arisen. Writers have hyped that these neuron-inspired processors can do almost anything. These exaggerations have created disappointments for some potential users who have tried, and failed, to solve their problems with neural networks. These application builders have often come to the conclusion that neural nets are complicated and confusing. Computers have trouble recognizing even simple patterns much less generalizing those patterns of the past into actions of the future.

Now, advances in biological research promise an initial understanding of the natural thinking mechanism. This research shows that brains store information as patterns. Some of these patterns are very complicated and allow us the ability to recognize individual faces from many different angles. This process of storing information as patterns, utilizing those patterns, and then solving problems encompasses a new field in computing. This field, as mentioned before, does not utilize traditional programming but involves the creation of massively parallel networks and the training of those networks to solve specific problems. This field also utilizes words very different from traditional computing, words like behave, react, self-organize, learn, generalize, and forget.

3.2 Why Use Neural Networks?

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyze. This expert can then be used to provide projections given new situations of interest and answer "what if" questions.

Other advantages include:

1. Adaptive learning: An ability to learn how to do tasks based on the data given for training or initial experience.
2. Self-Organization: An ANN can create its own organization or representation of the information it receives during learning time.

3. Real Time Operation: ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.
4. Fault Tolerance via Redundant Information Coding: Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.

3.3 Neural Networks Versus Conventional Computers

Neural networks take a different approach to problem solving than that of conventional computers. Conventional computers use an algorithmic approach i.e. the computer follows a set of instructions in order to solve a problem. Unless the specific steps that the computer needs to follow are known the computer cannot solve the problem. That restricts the problem solving capability of conventional computers to problems that we already understand and know how to solve. But computers would be so much more useful if they could do things that we don't exactly know how to do.

Neural networks process information in a similar way the human brain does. The network is composed of a large number of highly interconnected processing elements (neurons) working in parallel to solve a specific problem. Neural networks learn by example. They cannot be programmed to perform a specific task. The examples must be selected carefully otherwise useful time is wasted or even worse the network might be functioning incorrectly. The disadvantage is that because the network finds out how to solve the problem by itself, its operation can be unpredictable.

On the other hand, conventional computers use a cognitive approach to problem solving; the way the problem is to solved must be known and stated in small unambiguous instructions. These instructions are then converted to a high level language program and then into machine code that the computer can understand. These machines are totally predictable; if anything goes wrong is due to a software or hardware fault.

Neural networks and conventional algorithmic computers are not in competition but complement each other. There are tasks are more suited to an algorithmic approach like arithmetic operations and tasks that are more suited to neural networks. Even more, a large number of tasks, require systems that use a combination of the two approaches (normally a conventional computer is used to supervise the neural network) in order to perform at maximum efficiency.

3.4 History of Artificial Neural Network Research

Neural network simulations appear to be a recent development. However, this field was established before the advent of computers, and has survived at least one major setback and several eras.

The history of neural networks can be divided into several periods:

3.4.1 First Attempts: There were some initial simulations using formal logic. McCulloch and Pitts [14] (1943) developed models of neural networks based on their understanding of neurology. These models made several assumptions about how neurons worked. Their networks were based on simple neurons which were considered to be binary devices with fixed thresholds. The results of their model were simple logic functions such as "a or b" and "a and b". Another attempt was by using computer simulations. Two groups (Farley and Clark, 1954; Rochester, Holland, Haibit and Duda, 1956). The first group (IBM researchers) maintained closed

contact with neuroscientists at McGill University. So whenever their models did not work, they consulted the neuroscientists. This interaction established a multidisciplinary trend which continues to the present day [14].

3.4.2 Promising & Emerging Technology: Not only was neuroscience influential in the development of neural networks, but psychologists and engineers also contributed to the progress of neural network simulations. Rosenblatt (1958) stirred considerable interest and activity in the field when he designed and developed the Perceptron. The Perceptron had three layers with the middle layer known as the association layer. This system could learn to connect or associate a given input to a random output unit. Another system was the ADALINE (ADAptive LInear Element) which was developed in 1960 by Widrow and Hoff (of Stanford University). The ADALINE was an analogue electronic device made from simple components. The method used for learning was different to that of the Perceptron, it employed the Least-Mean-Squares (LMS) learning rule [14], [15].

3.4.3 Period of Frustration & Disrepute: In 1969 Minsky and Papert [14] wrote a book in which they generalized the limitations of single layer Perceptrons to multilayered systems. In the book they said: "...our intuitive judgment that the extension (to multilayer systems) is sterile". The significant result of their book was to eliminate funding for research with neural network simulations. The conclusions supported the disenchantment of researchers in the field. As a result, considerable prejudice against this field was activated.

3.4.4 Innovation: Although public interest and available funding were minimal, several researchers continued working to develop neuromorphically based computational methods for problems such as pattern recognition. During this period several paradigms were generated which modern work continues to enhance. Grossberg's (Steve Grossberg and Gail Carpenter in 1988) influence founded a school of thought which explores resonating algorithms. They developed the ART (Adaptive Resonance Theory) networks based on biologically plausible models. Anderson and Kohonen developed associative techniques independent of each other. Klopf (A. Henry Klopf) in 1972, developed a basis for learning in artificial neurons based on a biological principle for neuronal learning called heterostasis. Werbos (Paul Werbos 1974) developed and used the back-propagation learning method, however several years passed before this approach was popularized. Back-propagation nets are probably the most well known and widely applied of the neural networks today. In essence, the back-propagation net. is a Perceptron with multiple layers, a different threshold function in the artificial neuron, and a more robust and capable learning rule [15], [16].

Amari (A. Shun-Ichi 1967) was involved with theoretical developments: he published a paper which established a mathematical theory for a learning basis (error-correction method) dealing with adaptive pattern classification. While Fukushima (F. Kunihiko) developed a step wise trained multilayered neural network for interpretation of handwritten characters. The original network was published in 1975 and was called the Cognitron [14].

3.4.5 Re-Emergence: Progress during the late 1970s and early 1980s was important to the re-emergence on interest in the neural network field. Several factors influenced this movement. For example, comprehensive books and conferences provided a forum for people in diverse fields with specialized technical languages, and the response to conferences and publications was quite positive. The news media picked up on the increased activity and tutorials helped disseminate the technology. Academic programs appeared and courses were introduced at most major Universities (in US and Europe). Attention is now focused on funding levels

throughout Europe, Japan and the US and as this funding becomes available, several new commercial with applications in industry and financial institutions are emerging [16].

3.4.6 Today: Significant progress has been made in the field of neural networks-enough to attract a great deal of attention and fund further research. Advancement beyond current commercial applications appears to be possible, and research is advancing the field on many fronts. Neurally based chips are emerging and applications to complex problems developing. Clearly, today is a period of transition for neural network technology.

Year	*Research*	*Researcher*
<1940	Biological NN of living animals in Neuro-biological context	Alan Turing, Warren McCulloch,
1943	First ANN model, MP model, First electrical circuit, Perform any logical function on its inputs.	Warren McCulloch Walter Pitts
1949	Organization of Behaviour	Donald Hebb
1950	Digital computer	
1954, 56	Dartmouth Project	Farley and Clark, Rochester, Holland, Haibit and Duda
1957	Perceptron, supervised learning, reinforcement learning.	Frank Rosenblatt, R. Bellman
1959	ADALINE, MADALINE, First real world application of ANN.	Bernard Widrow, Marcian Hoff
1969	Book '*Perceptrons*', starting of "disillusioned years"	Marvin Minsky Seymour Papert
1972	Associative techniques, Heterostasis	Anderson, T.Kohonen A. Henry Klopf
1974	Introduction of Back-propagation algorithm	Paul Webros
1975	Cognitron, first multi-layered network	Kunihiko Fukushima
1973, 76	Hebb's postulate of learning	G. S. Stent, Danchin, Changuex
1976	Self Organising Map (SOM)	Willshaw, Von der Malsburg, Grossberg
1982	Self organizing feature mapping (SOFM)	T. Kohonen
1982	Limitation and ability of ANN, he showed that it can be used as an mathematical tool	John Hopfield
1985	Introduction of Radial-Basis Function networks	M. J. D. Powell
1982, 89	Introduction of energy function and temperature (*statistical mechanics*)	Hopfield, Grossberg, Amit
1985	Back-propagation algorithm	Parker, LeCun
1986	Book 'Parallel Distributed processing'	RumelHart,McClelland
1988, 91	Introduced the general notion of the 'space' of the NN, Hot Networks	Gardner, Amari
1983	Boltzman Machine	Hinton, Sejonwski
1991	Unsupervised learning	S. Becker
1989, 90	Recurrent neural network	John Hopfield, Jeff Elman

1989	Silicon ear, silicon retina	Carver Mead
1995	Computational maps in the primary visual cortex of the macaque monkey.	Erwin
Present	Hybrid technology, improving generalisation of NN, different regularization method, Neuro chips - digital, analog, and optical.	……..

Table 3.1: List of Events in the history of ANN research [14].

3.5 Human and Artificial Neurons - Investigating the Similarities

3.5.1 How the Human Brain Learns?

The brain is a collection of about 10 billion interconnected neurons. Each neuron is a cell [right] that uses biochemical reactions to receive process and transmit information.

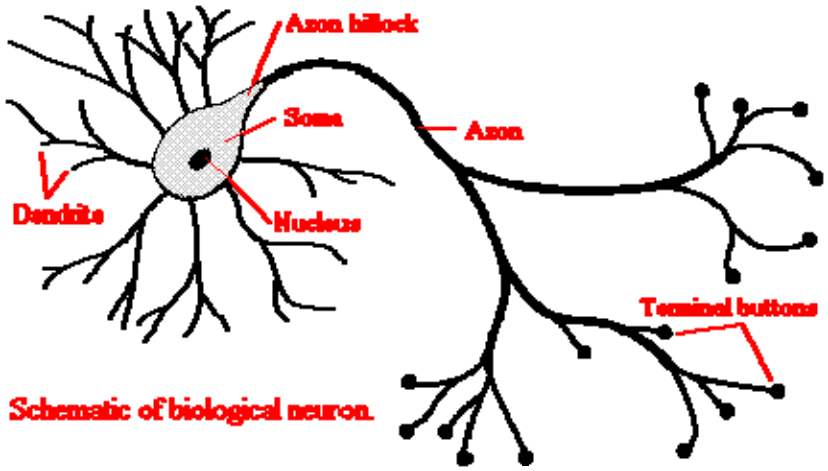


Fig 3.1: The basic features of a biological neuron

A neuron's dendrite tree is connected to a thousand neighboring neurons. When one of those neurons fire, a positive or negative charge is received by one of the dendrites. The strengths of all the received charges are added together through the processes of spatial and temporal summation. Spatial summation occurs when several weak signals are converted into a single large one, while temporal summation converts a rapid series of weak pulses from one source into one large signal. The aggregate input is then passed to the soma (cell body). The soma and the enclosed nucleus don't play a significant role in the processing of incoming and outgoing data. Their primary function is to perform the continuous maintenance required to keep the neuron functional. The part of the soma that does concern itself with the signal is the axon hillock. If the aggregate input is greater than the axon hillock's threshold value, then the neuron *fires*, and an output signal is transmitted down the axon. The strength of the output is constant, regardless of whether the input was just above the threshold, or a hundred times as great. The output strength is unaffected by the many divisions in the axon; it reaches each terminal button with the same intensity it had at the axon hillock. This uniformity is critical in an analogue device such as a brain where small errors can snowball, and where error correction is more difficult than in a digital system.

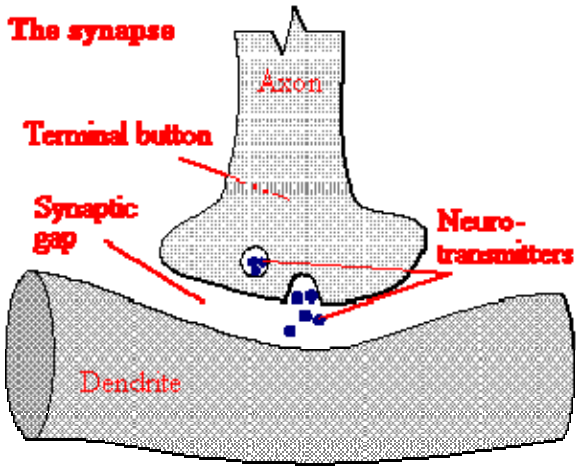


Fig 3.2: The synapse. Neurotransmitters released from the synaptic vesicles and trigger the receivers on the dendrite.

Each terminal button is connected to other neurons across a small gap called a synapse [left]. The physical and neurochemical characteristics of each synapse determine the strength and polarity of the new input signal. This is where the brain is the most flexible, and the most vulnerable. Changing the constitution of various neurotransmitter chemicals can increase or decrease the amount of stimulation that the firing axon imparts on the neighboring dendrite. Altering the neurotransmitters can also change whether the stimulation is excitatory or inhibitory. Many drugs such as alcohol and LSD have dramatic effects on the production or destruction of these critical chemicals. The infamous nerve gas sarin can kill because it neutralizes a chemical (acetyl cholinesterase) that is normally responsible for the destruction of a neurotransmitter (acetylcholine). This means that once a neuron fires, it keeps on triggering all the neurons in the vicinity. One no longer has control over muscles, and suffocation ensues.

3.6 From Human Neurons to Artificial Neurons

We conduct these neural networks by first trying to deduce the essential features of neurons and their interconnections. We then typically program a computer to simulate these features. However because our knowledge of neurons is incomplete and our computing power is limited, our models are necessarily gross idealizations of real networks of neurons.

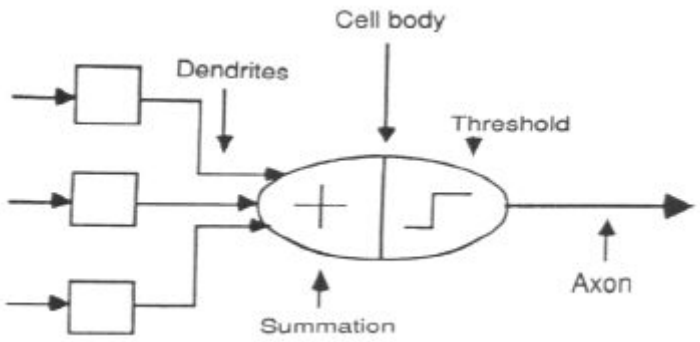


Fig 3.3: Neuron Model

Biological	*Artificial Neural Network*
Neuron	node/ unit/ cell/neurode
Synapse	connection/ edge/ link
synaptic efficiency	connection strength/ weight
firing frequency	node output

Table 3.2: Terminology

3.7 A More Complicated Neuron

The previous neuron doesn't do anything that conventional computers don't do already. A more sophisticated neuron (figure 2) is the McCulloch and Pitts model (MCP). The difference from the previous model is that the inputs are ‘weighted’; the effect that each input has at decision making is dependent on the weight of the particular input. The weight of an input is a number which when multiplied with the input gives the weighted input. These weighted inputs are then added together and if they exceed a pre-set threshold value, the neuron fires. In any other case the neuron does not fire.

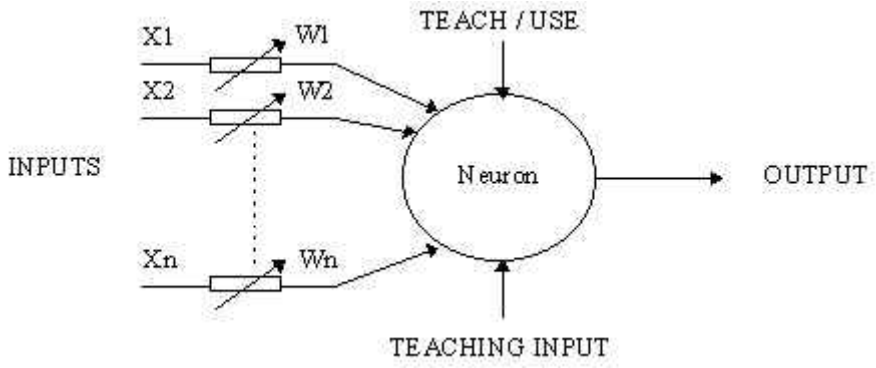


Figure 3.4: An MCP neuron

In mathematical terms, the neuron fires if and only if;

X1W1 + X2W2 + X3W3 + ... > T

The addition of input weights and of the threshold makes this neuron a very flexible and powerful one. The MCP neuron has the ability to adapt to a particular situation by changing its weights and/or threshold. Various algorithms exist that cause the neuron to 'adapt'; the most used ones are the Delta rule and the back error propagation.

3.8 Training and Memory of a Neural Network

The neural network which has not been trained is not able to solve any particular problem. This situation can be compared to a little baby, whose brain is fully developed and ready for work but who is not able to do anything because it has not experienced any stimulus. So a neural network without learning is analogous to a human without education.

Therefore, a neural network must be trained to solve some particular problem. The methodology of the training is analogous to the way you would teach a child to read or to count, that is by presenting some number or letter and by assigning the letters and numbers some values. For example you could show a child of 5 years an image of an A and you could then tell him that it's an "A'ye" with the best pronunciation you could manage.

You will teach a neural network in exactly the same way, namely you will feed our network with a set of numbers (in our library between 0 and 1), and the network will give you a result in its output layer. Since the weights of the connections in the network are initially in a random state, this result will surely at the beginning not satisfy you, so you will change the weight of some connections in order to obtain a better result. You will change the weight of the connections in fact until you get the desired result (this is the training stage).

Next you will feed the input layer of the network with other examples and continue adjusting weights, until eventually you obtain the desired output for each example.

The entire set of training examples must be shown to the network many times in order to get a satisfactory result. You would not expect a child to learn to read having seen each letter or word only once, similarly the network requires many examples.

After all of this training, your network is hopefully able to solve your problem - we say that it has learned, and its `knowledge' is stored by all the different connection weights.

3.9 Learning Method

There are two approaches to training - supervised and unsupervised.

3.9.1 Supervised Learning

The 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.

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.

3.9.2 Unsupervised Learning

Unsupervised learning is the great promise of the future. It shouts that computers could someday learn on their own in a true robotic sense. 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.

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.

At the present state of the art, unsupervised learning is not well understood and is still the subject of research. This research is currently of interest to the government because military situations often do not have a data set available to train a network until a conflict arises.

3.10 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.

3.10.1 Hebb's Rule

The first, and undoubtedly the best known, learning rule was 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.

3.10.2 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."

3.10.3 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.

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 Feed forward, 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.

3.10.4 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. This is especially important for applications where the input data is not derived from a strong underlying model.

3.10.5 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.

3.11 Applications of Neural Networks

3.11.1 Neural Networks in Practice

Given this description of neural networks and how they work, what real world applications are they suited for? Neural networks have broad applicability to real world business problems. In fact, they have already been successfully applied in many industries.

Since neural networks are best at identifying patterns or trends in data, they are well suited for prediction or forecasting needs including –

- Sales forecasting
- Industrial process control
- Customer research
- Data validation
- Risk management
- Target marketing

But to give you some more specific examples; ANN are also used in the following specific paradigms: recognition of speakers in communications; diagnosis of hepatitis; recovery of telecommunications from faulty

software; interpretation of multimeaning Chinese words; undersea mine detection; texture analysis; three-dimensional object recognition; hand-written word recognition; and facial recognition [17].

3.11.2 Neural networks in medicine

Artificial Neural Networks (ANN) are currently a 'hot' research area in medicine and it is believed that they will receive extensive application to biomedical systems in the next few years. At the moment, the research is mostly on modeling parts of the human body and recognizing diseases from various scans (e.g. cardiograms, CAT scans, ultrasonic scans, etc.).

Neural networks are ideal in recognizing diseases using scans since there is no need to provide a specific algorithm on how to identify the disease. Neural networks learn by example so the details of how to recognize the disease are not needed. What is needed is a set of examples that are representative of all the variations of the disease. The quantity of examples is not as important as the 'quantity'. The examples need to be selected very carefully if the system is to perform reliably and efficiently [18].

3.11.3 Modeling and Diagnosing the Cardiovascular System

Neural Networks are used experimentally to model the human cardiovascular system. Diagnosis can be achieved by building a model of the cardiovascular system of an individual and comparing it with the real time physiological measurements taken from the patient. If this routine is carried out regularly, potential harmful medical conditions can be detected at an early stage and thus make the process of combating the disease much easier.

A model of an individual's cardiovascular system must mimic the relationship among physiological variables (i.e., heart rate, systolic and diastolic blood pressures, and breathing rate) at different physical activity levels. If a model is adapted to an individual, then it becomes a model of the physical condition of that individual. The simulator will have to be able to adapt to the features of any individual without the supervision of an expert. This calls for a neural network.

Another reason that justifies the use of ANN technology is the ability of ANNs to provide sensor fusion which is the combining of values from several different sensors. Sensor fusion enables the ANNs to learn complex relationships among the individual sensor values, which would otherwise be lost if the values were individually analyzed. In medical modeling and diagnosis, this implies that even though each sensor in a set may be sensitive only to a specific physiological variable, ANNs are capable of detecting complex medical conditions by fusing the data from the individual biomedical sensors [19].

3.11.4 Electronic noses

ANNs are used experimentally to implement electronic noses. Electronic noses have several potential applications in telemedicine. Telemedicine is the practice of medicine over long distances via a communication link. The electronic nose would identify odors in the remote surgical environment. These identified odors would then be electronically transmitted to another site where an door generation system would recreate them. Because the sense of smell can be an important sense to the surgeon, telesmell would enhance telepresent surgery [20].

3.11.5 Instant Physician

An application developed in the mid-1980s called the "instant physician" trained an auto associative memory neural network to store a large number of medical records, each of which includes information on symptoms, diagnosis, and treatment for a particular case. After training, the net can be presented with input consisting of a set of symptoms; it will then find the full stored pattern that represents the "best" diagnosis and treatment.

3.11.6 Neural Networks in business

Business is a diverted field with several general areas of specialization such as accounting or financial analysis. Almost any neural network application would fit into one business area or financial analysis. There is some potential for using neural networks for business purposes, including resource allocation and scheduling. There is also a strong potential for using neural networks for database mining that is, searching for patterns implicit within the explicitly stored information in databases. Most of the funded work in this area is classified as proprietary. Thus, it is not possible to report on the full extent of the work going on. Most work is applying neural networks, such as the Hopfield-Tank network for optimization and scheduling.

3.11.7 Marketing

There is a marketing application which has been integrated with a neural network system. The Airline Marketing Tactician (a trademark abbreviated as AMT) is a computer system made of various intelligent technologies including expert systems. A feed forward neural network is integrated with the AMT and was trained using back-propagation to assist the marketing control of airline seat allocations. The adaptive neural approach was amenable to rule expression. Additionally, the application's environment changed rapidly and constantly, which required a continuously adaptive solution. The system is used to monitor and recommend booking advice for each departure. Such information has a direct impact on the profitability of an airline and can provide a technological advantage for users of the system. [Hutchison & Stephens, 1987]

While it is significant that neural networks have been applied to this problem, it is also important to see that this intelligent technology can be integrated with expert systems and other approaches to make a functional system. Neural networks were used to discover the influence of undefined interactions by the various variables. While these interactions were not defined, they were used by the neural system to develop useful conclusions. It is also noteworthy to see that neural networks can influence the bottom line.

3.11.8 Credit Evaluation

The HNC Company, founded by Robert Hecht-Nielsen, has developed several neural network applications. One of them is the Credit Scoring system which increases the profitability of the existing model up to 27%. The HNC neural systems were also applied to mortgage screening. A neural network automated mortgage insurance underwriting system was developed by the Nestor Company. This system was trained with 5048 applications of which 2597 were certified. The data related to property and borrower qualifications. In a conservative mode the system agreed on the underwriters on 97% of the cases. In the liberal model the system agreed 84% of the cases. This is system run on an Apollo DN3000 and used 250K memory while processing a case file in approximately 1 sec.

3.12 What the Next Developments will be

The vendors within the industry predict that migration from tools to applications will continue. In particular, the trend is to move toward hybrid systems. These systems will encompass other types of processes, such as fuzzy logic, expert systems, and kinetic algorithms. Indeed, several manufactures are working on "fuzzy neurons."

The greatest interest is on merging fuzzy logic with neural networks. Fuzzy logic incorporates the inexactness of life into mathematics. In life most pieces of data do not exactly fit into certain categories. For instance, a person is not just short or tall. He can be kinda short, pretty tall, a little above average, or very tall. Fuzzy logic takes these real-world variations into account. In potential application of neural networks, in systems which solve real problems, this fuzziness is a large part of the problem. In automating a car, to stop is not to slam on the brakes, to speed up is not to "burn rubber." To help neural networks accommodate this fuzziness of life, some researchers are developing fuzzy neurons. These neurons do not simply give yes/no answers. They provide a more fuzzy answer.

Systems built with fuzzy neurons may be initialized to what an expert thinks are the rules and the weights for a given application. This merging of expert systems and fuzzy logic with neural networks utilizes the strength of all three disciplines to provide a better system than either can provide themselves. Expert systems have the problem that most experts don't exactly understand all of the nuances of an application and, therefore, are unable to clearly state rules which define the entire problem to someone else. But the neural network doesn't care that the rules are not exact, for neural networks can then learn, and then correct, the expert's rules. It can add nodes for concepts that the expert might not understand. It can tailor the fuzzy logic which defines states like tall, short, fast, or slow. It can tweak itself until it can meet the user identified state of being a workable tool. In short, hybrid systems are the future.

Chapter 4 : Self Organizing Map

4.1 Introduction

Tuevo Kohonen, One of the leading researchers of artificial neural network and Professor of the Faculty of Electrical Engineering at the Helsinki University of Technology, has developed an unsupervised self-organizing map or network. In unsupervised learning, that no teacher is needed to define the correct output and the network is provided with inputs but not with desired outputs. The system itself must then decide what features it will use to group the input data. This is often referred to as self-organization or adaption.

It has been postulated that the brain uses spatial mapping to model complex data structure internally. Kohonen uses this idea to good advantage in his network because it allows him to perform data compression on the vectors to be stored in the network using a technique known as *vector quantization*. It also allows the network to store data in such way that spatial or topological relationships in the training data are maintained and represented in a meaningful way.

Self Organizing Map--competitive networks that provide "topological" mapping from the input space to the clusters. The SOM was inspired by the way in which various human sensory impressions are neurologically mapped into the brain such that spatial or other relations among stimuli correspond to spatial relations among the neurons. In a SOM, the neurons (clusters) are organized into a grid--usually two-dimensional or (rarely) three- or more-dimensional. The grid exists in a space that is separate from the input space. All inputs connect to every node in the network.

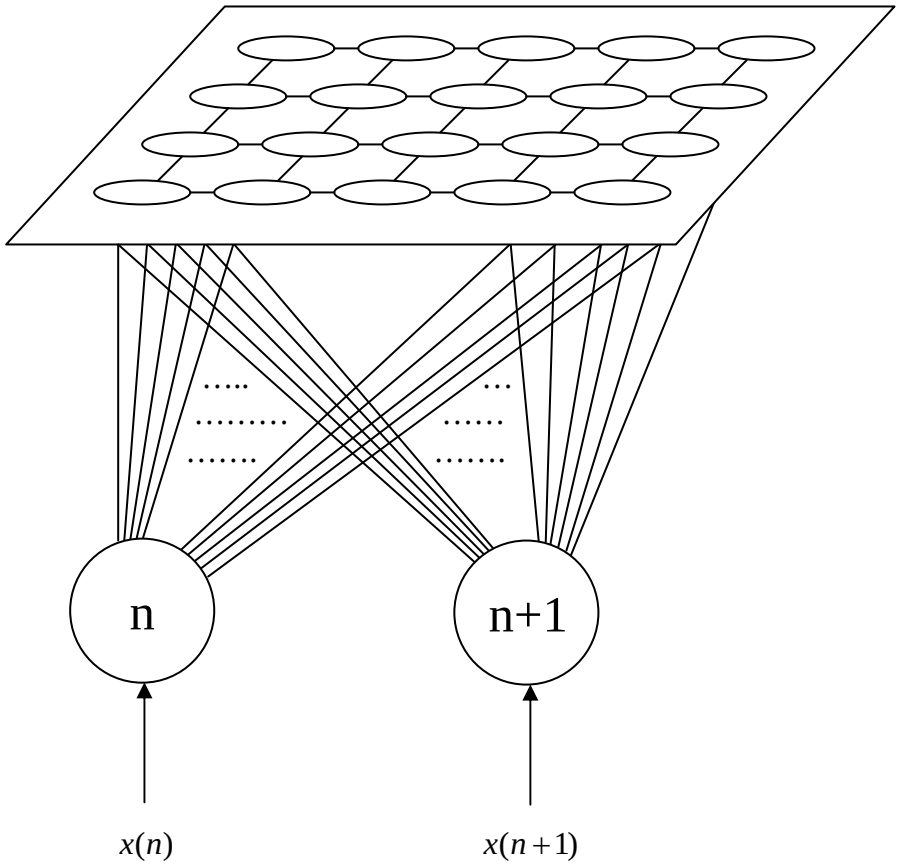


Fig 4.1 : Graphical Representation of Kohonen's Network

4.2 Kohonen Self Organizing Map Algorithm

Step 1. Initialize weights where $W_{ij}(t)\ \ (0 \le i \le n-1)$ represents weight from input i to node j at time t. Weights are initialized randomly within a fixed range and should be smaller enough..

Step 2. Determine present input $x_0(t), x_1(t), x_2(t),.....x_{n-1}(t)$, where $x_i(t)$ is the input node i at iteration t.

Step 3. Calculate distances d_j where

$$d_j = \sum_{i=0}^{n-1} \left(x_i(t) - w_{ij}(t)\right)^2$$

These distances are the differences between the input pattern and the weight vectors.

Step 4. Select the minimum distance which represents the unit with the closest match to the input from all the distances. The output node with minimum distance is represented by j^* which is the center of the neighborhood and marked it as a 'winner' (most active unit).

Step 5. Update weights for j^* The modified weights are –

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)\left(x_i(t) - w_{ij}(t)\right)$$

Where $\eta(t)$ is a gain factor which varies between 0 and 1. Above this equation ensures that the weights in the neighborhoods become more similar to the input and this procedure requires much iteration. But for the network properly converges to a solution the iterations must be limited. So, we adapt the weight according to a linear decreasing function with the number of iterations through the training set.

Step 6. Set neighbors of the winner node defined by the neighborhood size $N_{j^*}(t)$. Initially the number of neighborhoods is very large. As training process goes on the size of the neighborhood is decreased slowly to a predefined limit, thus localizing the area of maximum activity.

Step 7. If the minimum distance is not zero repeat the learning cycle (step 2–6)

The two most central issues to adaptive self-organizing learning in a Kohonen network are the weight adaption process and the concept of topological neighborhoods of nodes.

4.3 Weight Training

There is no derivative process involved in adapting the weights for the Kohonen network. Referring to the algorithm again we can see that the change in the input vector and the weight vector:

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)\left(x_i(t) - w_{ij}(t)\right)$$

Where w_{ij} is the i th component of weight vector to node j for j in the neighborhood $N_{j^*}(t)$.

The unit of proportionality is $\eta(t)$, the learning rate coefficient, where $0 < \eta(t) < 1$. This term decrease the adaption rate with the time. We can visualize the training cycle as having two stages. The first stage is creating some form of topological ordering on the map of randomly oriented nodes. The training process attempts to cluster the nodes on topological map to reflect the range of class types found in the training data. This will be a coarse mapping, where the network is discovering how many classes should lie in relation to each other on the map. These are large scale changes to the orientation of the nodes on the map, so the adaption rate is kept high to allow mapping as quickly as possible. Once a stable coarse representation is found, the nodes within the localized regions of the map are fine-tuned to the input training vectors. To achieve the fine-tuning much smaller changes must be made to the weight vectors at each node, so the adaption rate is reduced as training progresses.

Each time a new training input is applied to the network the winning node must first be located: this identifies the region of the feature map that will have its weight values updated. The winning node is categorized as the node that has the closest matching weight vector to the input vector, and the metric that is used to measure the similarity of the vectors is the Euclidean distance measure. There are, however, a few subtleties to note in implementing the technique in the Kohonen network. The Euclidean norm of a vector is a measure of its magnitude of other words; we will describe two vectors as being similar if they are pointing in the same direction, regardless of their magnitude. The only way that we can ensure that we are comparing the orientation of two vectors, using the Euclidean measure, is to first make sure that all the weight vectors are normalized. Normalizing a vector for a set of vectors in Euclidean space this means that each vector will retain its orientation but will be of a fixed length, regardless of its previous magnitude. The comparison of the weight vectors and the input vector will now be a concerned only with the orientation as required. Another useful advantage of normalizing the vector is that it reduces the training time for the network because it removes one degree of variability in the weight space. Effectively that means that the weight vectors start in an orientation that is closer to the desired state. Thus reducing some of the reorientation time during the training cycle.

4.3.1 Initializing The Weights

The network weights should be set to small, normalized random values. However, this is an over-simplification because if the weight vectors are truly randomly spread, the network may suffer non-convergent or very slow training cycles. Typically the input training vectors will fall into clusters over a limited region of the pattern space, corresponding to their class. If the weight vectors stored at the nodes in the network are randomly spread then the situation could quite easily arise where many of the weight vectors are in a very different orientation to the majority of the training inputs. These will not win any of the best match comparison and will remain unused in forming the topological map. The consequence of this is that the neighborhoods on the feature map will be very sparsely populated with trainable nodes, so much so that there may not be enough usable nodes to

adequately separate the classes. This will result in very poor classification performance due to the inability of the feature map to distinguish between the inputs.

4.4 Neighborhoods

Kohonen introduces the idea of topological neighborhoods. This is a dynamically changing boundary that defines how many nodes surrounding the winning node will be affected with weight modifications during the training process. Initially each node in the network will be assigned a large neighborhood. When a node is selected as the closest match to an input it will have its weights adapted to tune it to the input signal. However, all the nodes in the neighborhood will also be adapted by a smaller amount. As training progresses the size of the neighborhood is slowly decreased to predefined limit. To appreciate how this can force clusters of nodes that are topological related consider the sequence of diagram shown in figure 4.2 that represents the topological forming of the feature clusters during a training session. For clarity, we shall show the formation of just one cluster which is centered about highlighted node.

In A the network is shown in its initialized state with random weight vectors and large neighborhoods around each node. The arrows within each node can be thought of as a spatial representation of the orientation of each nodes weight vector. Training commences as previously described; for each training input the best match node is found, the weight change is calculated and all the nodes in the neighborhood are adjusted.

In B we can see the network after many passes through the training set. The highlighted region of the map is beginning to form a specific class orientation based around the highlighted node. The neighborhood size has also shrunk so that weight modifications now have a smaller field of influence.

The fully trained network is shown in C. The neighborhood have shrunk to a predefined limit of four nodes and the nodes within the region have all been adapted to represent an average spread of values about the training data for that class.

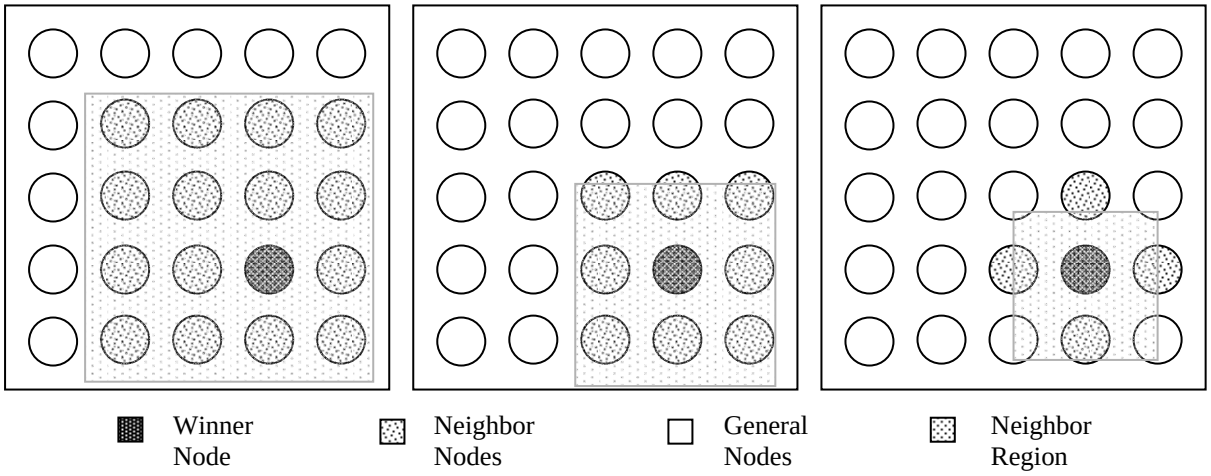


Fig 4.2: Training a localized neighborhood

The training algorithm will produce clusters for all the class types found in the training data. The ordering of the clusters on the map and the convergence times for training are dependent on the way the training data is presented to the network. Once the network has self-organized the internal representation the clusters on the feature map can be

labeled to indicate their class so that the network can be used to classify unknown inputs. Note that the network forms the internal features without supervision but the classification labeling must be done by hand once the network has been fully trained.

4.5 Reducing the Neighborhood

We have already stressed that the neighborhood size is reduced with time during the training sequence. But how quickly do we reduce it and to what final size? Unfortunately there are no hard and fast rules for adaptive training algorithms of this nature and some experimentation will be required in individual applications. However Kohonen does stress that his method is not one that is brittle that is small changes in system parameters do not reflect gross divergence of training results and also suggest some rules of thumb as a starting point for intuitive tweaking. We have explained that the adaption rate must be reduced during the training cycle so that weight changes are made more and more gradual as the map develops. This ensures that clusters form accurate internal representation of the training data as well as causing the network to convergence to a solution within a predefined time limit. In typical applications Kohonen suggests that the adaption rate be a linearly decreasing function with the number of passes through the training set.

Training is affected not only by the adaption rate and the rate at which the neighborhood is reduced but also by the shape of the neighborhood boundary. The example we used earlier in figure 5.3 only discussed the possibility of using a square neighborhood – however, that is not to say that we cannot define a circular or even a hexagonal region and these may provide optimal results in some cases. As with the adaptation rate, however, it is preferable to start with neighborhood fairly wide initially and allow them to decrease slowly with the number of training passes.

Chapter 5 : Experimentation

5.1 Introduction

This chapter concerned is with the detailed view of the overall research work on the *Recognition of Human Faces using Artificial Neural Network (ANN)*. We are going to describe a procedural detail in this chapter. The overall research work contains a number of procedural steps. Such as converting a RGB color image to grayscale and various *Image Processing* steps, learning a pattern likely as human brain's biological neuron, recognize a existing or non-existing pattern from the learned pattern using ANN technique etc.

All the steps of image processing and feature detection steps are performed and coded by the programming language *Matlab* provided by *The Mathwork Inc*. Primary operation of Matlab is to acquire an image from the storage disk which is stored in a specific format and process the image. One of the most important factors of the research work is *Feature Extraction* which is called *Face Detection* here; is implemented by Matlab. Then the processed pattern is stored in a *Binrary File* such that the file can be processed by another language further.

All other works such as Pattern Learning and Pattern Recognition is performed by Borland C++ (32-bit Programming). Program coded by this language takes the stored pattern (from Matlab) as its input pattern and makes corresponding weight set. Also the weight set and *Winner Node* of Kohonen Network is stored in a binary file. The purpose of winner node is to identify the actual pattern for recognition.

Recognition of pattern includes the testing of user input pattern with the all sets of stored weights. Program takes the user given pattern which is processed and stored by Matlab program and then tests by applying all the weights stored. Then match the resultant winner node with the currently found node. If they match or its neighbors match, then the recognition process is successful. Otherwise the program is failed to recognize the pattern; that means the system doesn't know about the given pattern.

This is the main theme of the project work. Moreover its efficiency can be increased by giving several expressions of a single person so that any further expression of the same person can be nearest of any other stored patterns. Then it will be easier more to recognize.

5.2 Why and Where Matlab is Efficient

Matlab is a complete programming language which provides several built in functions specially for filtering, curve fitting, region plotting etc. Moreover it provides various toolboxes; such as Image Processing toolbox, Neural Network toolbox etc.

Some of the provided facilities by Matlab are described below.

5.2.1 Provided Built-In Functions

Matlab provides all types of signal and image filtering as its built-in function. So it decreases the program complexities and length of the program. Suppose; a basic low pass median filter uses a mask to convolve an

image. For a 15×15 median filtering, it needs to convolve with a matrix with 265 elements. This can be done by a single Matlab function –

```
B = medfilt2(A,[m n])
```

There are such types of several functions to do several works of image processing.

5.2.2 Faster and Efficient

Matlab is specially designed for these types of operations. So it is more and more faster than any other languages. These types of operations can also be done by some other languages. But from the viewpoint of time and complexity, Matlab is faster than all others. Moreover the structures of the internal operations are easier and less complex. It makes the entire process faster than any other.

5.3 Importance of C Language

C language is one of the most powerful high level languages of the world. All types of operations can be customized by the language. This consists of all types of basic and system related operations such as sub-routine calling, memory optimizing etc.

5.3.1 Efficient Use of Memory

C language provides efficient allocation of memory. So it avoids the uncertain allocation and wastage of memory locations. C language doesn't store the previously stored dataset though the execution is completed. So there is no chance of data confliction with the previous pattern information.

5.3.2 Flexibility of Coding

C language is flexible enough. Its code to handling the memory location is easier and less complex. Though it does not provide a lot of built-in function to perform the image processing and neural network operations; but it is useful where basic operations are performed.

In our work, for learning and recognizing the face pattern; C language is used. Because, these are manual processes and need lots of looping, conditions that are flexible enough than Matlab or other low level languages. So it has a great advantage while implementing an existing algorithm and designating a non-existing algorithm.

5.4 Steps of the Research

The entire work is subdivided into two basic parts. One of them is *Image Processing* and the other is *Neural Network* part. It has described before about the languages in which the both parts have been done. Now we are going to describe about the step by step operations which is done to implement our research work.

5.4.1 Image Processing Part

Image Processing part begins with the *Image Acquisition* and ends at the *Scaling* of the processed image and saving to a file. Image acquisition process is not real time. All the images need to be stored in a storage device at any given name. Matlab program takes the image from there and then process. Neural network always works with fixed size of inputs. So at last, it needs to scale the variable size image into a fixed size image. And also, all the neural algorithms are comparatively slower. So it needs to shrink the dataset for faster processing. Minimal

size of input pattern increases the speed of learning process and finally increases the efficiency of the entire process.

Steps to process each face image are described below.

I. Acquiring Images

Acquisition of image is described in chapter 2. Best way to acquire images by a high quality *Digital Camera* or *Flatbed Scanner*. We used both ways to acquire image from the real world, then saved the images to a specific directory. Actually it is not essential to save an image to any predefined directory. But it is better practice to save in a fixed place.

Figure 1 below shows how an image can be acquired from real word –

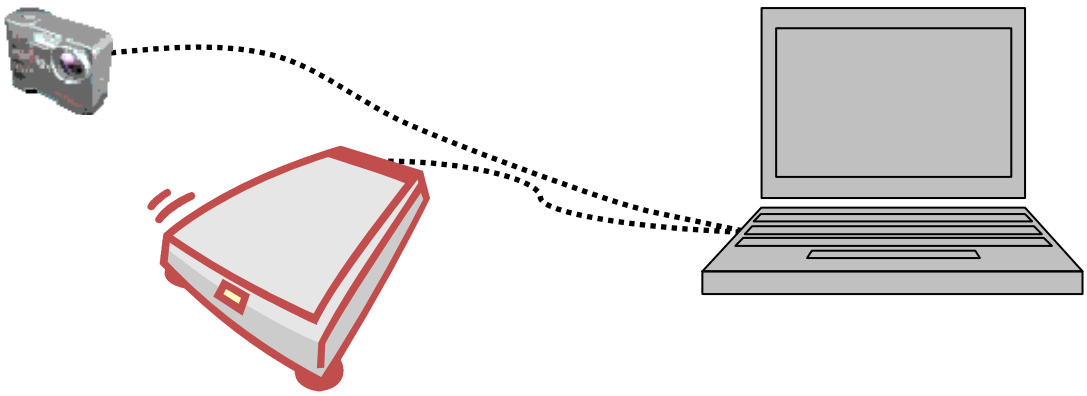


Fig 5.1 : Graphical Display of Image Acquisition Process

II. Input Image from Disk

It needs to take the image from the disk drive. Matlab has a predefined function to input an image from a disk that is

```
A = imread(filename,fmt) ----------
```
Func (5.1)

The function reads a *Grayscale* or *Truecolor* image named filename into A. If the file contains a grayscale intensity image, A is a two-dimensional array. If the file contains a truecolor (RGB) image, A is a three-dimensional ($m \times n \times 3$) array.

As for example, for a grayscale image A may be like –

255	201	211	210	158	124	…… ……	157
251	250	217	198	157	201	…… ……	147
201	147	208	144	217	205	…… ……	144
157	211	201	214	251	250	…… ……	217

198	157	201	250	217	147	…… ……	208
……	……	……	……	……	……	…… ……	……
……	……	……	……	……	……	…… ……	……
144	124	157	211	201	75	…… ……	104

Fig 5.2 : Feature Matrix of a Pattern

III. Convert a Color Image Into Grayscale

Actually, maximum types of pattern recognition need not to consider the color information of the image. Moreover, it is complex to work with a 3 dimensional matrix. So, for the face recognition system, it is better to use a color (RGB) image to grayscale.

The way to the conversion process is to average the values of every pixel points of the three matrices. The matlab function to do the process is –

```
M = median(A,dim) ----------------- Func (5.2)
```

This function returns the median values for elements along the dimension of A specified by scalar dim.

The following figures, Figure 5.3 differentiate between a *Color (RGB)* and a *Grayscale* image –

(a)

(b)

Fig 5.3 : (a) A Color (RGB) Image. (b) Grayscale Image

IV. Removing Noises by Filtering

Input images may contain lots of noises due improper acquisition i.e. scanning or getting picture through digital camera. So *dusts* and *scratches* may appear inside the acquired image. These types of dusts alter the structure of the pattern and make disturbance throughout the process; especially during *face detection.* So it is essential to remove the uncertain noises by image filtering.

We used *Lowpass Median Filter* which removes such types of noises and smooth the image pattern. The process is to take the median value from a $m \times n$ region of the image. As for example –

128	210	127

108	219	75
199	41	111

Fig 5.4 : A 3×3 Region of an Image

Above this figure, 128 is the median value of the region.

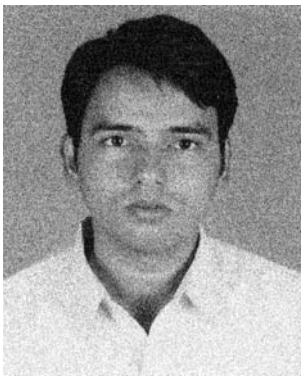
Matlab programming language also has a pre-defined function to perform the median filtering which receives the array and mask size as input. Then convolve the entire image by the mask.

The function used by Matlab is –

```
B = medfilt2(A,[m n]) ------------- Func (5.3)
```

This function performs median filtering of the matrix `A` in two dimensions. Each output pixel contains the median value in the $m \times n$ neighborhood around the corresponding pixel in the input image.

Variation of a Noisy Image and a Filtered Image is shown below in the Figure 5.5 –



(a) (b)

Fig 5.5 : (a) A Noisy Image. (b) Filtered Image by $m \times n$ Median Filter

`medfilt2(A,[m n])` function pads the image with zeros on the edges, so the median values for the points within $\lceil m \ n \rceil / 2$ of the edges may appear distorted.

V. Detecting the Edges of a Face

Edge detection is not essential for the image processing. But is most useful for detecting the feature i.e. the actual face portion from the entire image. The images do not contains only the human face; it may contain a background and blank portions by the four sides. So it is essential to extract the exact face from the image and where there is a face detection; there needs to detect the edge of the image.

Several methods may be used for edge detection purpose. As for example – prewitt, sobel, roberts, laplacian, canny, zero crossed etc.

Matlab uses a general function using which all methods of edge detection can be implemented. The function is as follows –

```
BW = edge(I,'method') ------------- Func (5.4)
```

The function takes an intensity image I as its input, and returns a binary image BW of the same size as I, with 1's where the function finds edges in I and 0's elsewhere.

edge supports six different edge-finding methods –

- The *Sobel* method finds edges using the *Sobel* approximation to the derivative. It returns edges at those points where the gradient of I is maximum.
- The *Prewitt* method finds edges using the *Prewitt* approximation to the derivative. It returns edges at those points where the gradient of I is maximum.
- The *Roberts* method finds edges using the *Roberts* approximation to the derivative. It returns edges at those points where the gradient of I is maximum.
- The *Laplacian* of Gaussian method finds edges by looking for zero crossings after filtering I with a *Laplacian* of Gaussian filter.
- The *Zero-cross* method finds edges by looking for zero crossings after filtering I with a filter you specify.
- The *Canny* method finds edges by looking for local maxima of the gradient of I . The gradient is calculated using the derivative of a Gaussian filter. We used *Prewitt* filter for the purpose of edge detection.

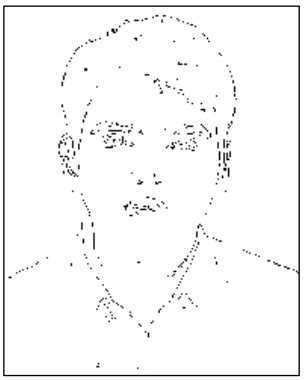
Mask used in prewitt filtering is as follows –

1	1	1
0	0	0
-1	-1	-1

Fig 5.6 : The Mask Used for Prewitt Filtering

The detected edge of a given pattern is shown in the figure 5.7 given below –

(a)



(b)

Fig 5.7 (a) An Image in Grayscale. (b) Detected Edge

5.4.2 Face Detection

The problem before face recognition is face detection which finds the face in the image. So it is necessary to determine the boundary of the face from an actual input image.

Face Detection Algorithms

Steps for determining the boundary of the face are as follows –

I. Detecting Top Boundary

The procedures of measuring the top boundary of a facial image of a human are

1. Set the first pixel's gray level as background
2. Choose appropriate threshold to ignore gray level variation.
3. Repeat until a different point whose gray level is smaller than the difference of back level and threshold is found –
    a. Go to the middle of width of matrix.
    b. Search the gray level of the point within the range.
    c. If found successive gray level, store the point and break.
    d. Go to step 3 until entire height of the image is searched
4. Remove all the points on the top of the selected point

II. Detecting Left Boundary

This procedure is very similar to the previous one. But it can be applied from the left side of the image:

1. Repeat until a different point whose gray level is smaller than the difference of back level and threshold is found –
    a. Go to the middle of height of the picture matrix
    b. Search the gray level of the point. If out of range then go to the next column.
    c. If found successive gray level, store the point and break.
    d. Go to step 1 until entire height of the image is searched
2. Remove all the points on the left of the selected point

III. Detecting Right Boundary

This procedure needs to search the image matrix from right to left to find out the right boundary of the face-

1. Repeat until a different point whose gray level is smaller than the difference of back level and threshold is found –
    a. Go to the middle of height of the picture matrix

b. Search the gray level of the point. If out of range then go to the previous column.

c. If found successive gray level, store the point and break.

d. Go to step 1 until entire height of the image is searched

2. Remove all the points on the right of the selected point

IV. Assumption and Extraction

This algorithm is how the length of a face is assumed and extracted from the image –

1. Measure face width from the difference of right and left boundary
2. Assume height is 125% of width
3. Identify the bottom point of the face from its height
4. Remove all points below the point.
5. Take a two-dimensional array of length that's the height and width of the extracted face

Copy the entire array to the newly allocated array replaces the previous dimensions with the new dimensions.

5.4.3 Graphical View of Face Detection

Fig 5.8.1

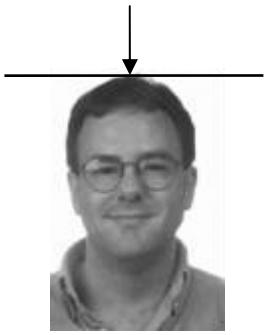


Fig 5.8.2

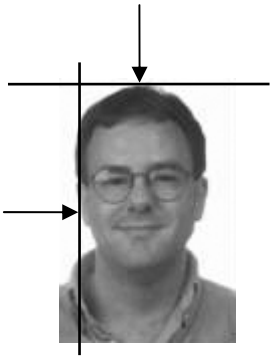


Fig 5.8.3

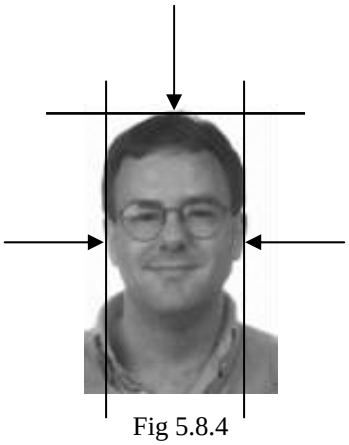


Fig 5.8.4

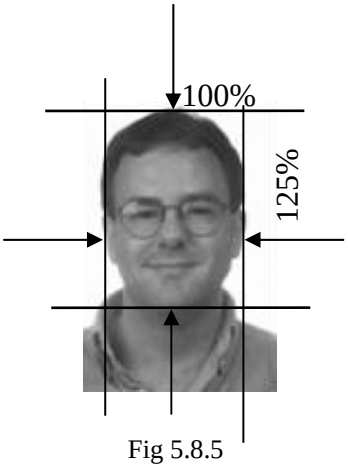


Fig 5.8.5

Fig 5.8.6

5.5 Graphical User Interface

We made a simple *visual interface* using Matlab. It contains some menus and a *picture box* which is called *axis control* in the language. Any loaded image is shown inside the axis control box. Then the processing operations may be done easily by clicking on several submenus. The main purpose of the visual interface is make the operations user friendly.

The figures and short notes contain information about the visual interface of the program.

5.5.1 File Menu

This menu contains two submenu named *Acquire* and *Exit*.

- Purpose of the *Acquire* submenu is to load an image stored in JPEG format from the disk. It displays the *open box* and the image can be selected from anywhere of the disk. It automatically places the image inside the axis control and resize according to the size of the control.

- Exit normally work as the cross button of the main window. It releaser the control from the program and return to the original workspace.

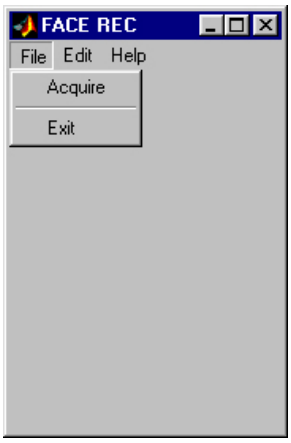


Fig 5.9 : Visual Interface (File Menu Highlighted)

5.5.2 Edit Menu

Edit menu contains three submenus. These submenus perform the main operations of the program concerned with the image processing tasks.

- *Pre-Processing* submenu takes the image loaded on the axis control as its input and process gradually those are described above. As for example; converting to grayscale, filtering, shrinking etc. And at last save the image to a binary file with a specific name so that another application can process is automatically as required. This submenu remains inactive when no image is loaded.

- *Auto Processing* submenu performs the automatic processing of all stored images into the specific folder. It searches the entire directory and lists all the files. Then takes each file, then process and save it into the predefined directory.

- *Reset* submenu is needed to initialize the process for starting the process from the root. There is a counter to count how many files are needed. It removes the entire stored and learned patterns and also set the counter to zero. Then user may start the process from the beginning.

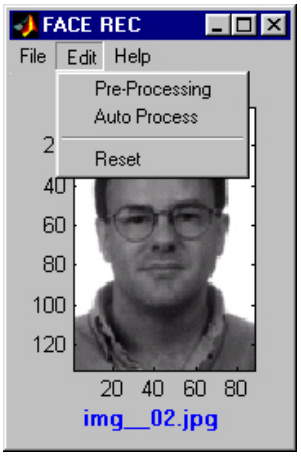


Fig 5.10 : Visual Interface (Edit Menu Highlighted)

5.5.3 Help Menu

Instruction to operate the program has not developed yet. Only a about dialog box is added which contains the information about the developers.

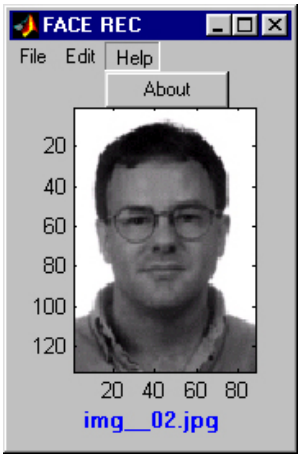


Fig 5.11 : Visual Interface (Help Menu Highlighted)

5.5.4 Additional Message Boxes

We used some other message box to inform user about the current status and completion of some specific tasks such as pre-processing and auto process.

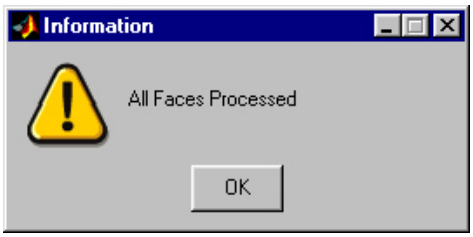


Fig 5.12 : Message Box After Processing All the Faces

5.6 Learning the Detected Feature

This operation is coded using C language due its flexibility. Learning starts from taking input from the binary file and fit to the *Kohonen Network*. Then the network is adjusted by adopting the weights.

- At first it needs to construct the network. That means the number of input and output nodes will be selected here. Efficiency and execution time depends on these parameters. It is also essential to consider the size of input so that the original pattern does not lose any of its information. As we took the pattern size 56×40, so the number of input node is 2240. The set of input is processed throughout the network and produce a set of weights and a selected node called *winner node*. We set the grid size 10×10 that means 100. Each of the input nodes is connected to each of the output nodes. That means the network is fully connected network.

- A set of weights w_{ij} should be initialized where $(0 \le i \le n-1)$ represents weight from input i to node j at time t. Weights are initialized randomly within a fixed range and should be smaller enough. Because face recognition needs to operate on a huge amount of input data. So it is difficult to maintain a long numbers. The amount of weights is the product of input and output nodes. That means; in our work we needed 2240×100 random weights. Practically we took the set of weights ranged between zero and two.

- Present input $x_0(t), x_1(t), x_2(t).....x_{n-1}(t)$ is need to be determined, where $x_i(t)$ is the input node i at iteration t. In this input set $x_0(t), x_1(t), x_2(t).....x_{n-1}(t)$ represents the gray levels of every pixels of the input pattern. Xi(0) where I =0,1,2…….n may vary from 0 to 255. It means, the input set will be applied to the input nodes of the Kohonen's map. Practically the input set is processed and ranged within zero to three for controlling the uncertain range of weighted sum. It is better to keep the values of the weighted sum within a reasonable range.

- To measure the variation of input and generated weights; the distance set d_j need to be calculated where

$$d_j = \sum_{i=0}^{n-1} \left(x_i(t) - w_{ij}(t) \right)^2$$

We calculate the distances between the connections of every unit in the grid and the input node. These distances are a function of the differences between the input pattern and the weight vectors. Distance is the measurement unit of error; that means the efficiency to learn the pattern.

We can represent it as –

```
For i= 0 to M-1
    Sum = 0
        For j = 0 to N-1
                Sum = Sum + (xi(t) –wij)2
        End of for loop
End of for loop
```

Then the calculated distances will be sort and a set of neighborhood nodes will be selected. There are no specific rules to measure the neighborhood size. It may depend on the minimum distance. Or it may be fixed at the beginning of the execution.

Now the neighborhood set of nodes indexed by j^* is measured by the values of distances those are smaller than the neighborhood value. The set of neighborhood nodes are stored in a different array.

- Now only the weights of the selected neighborhood nodes will be adopt. One of the factors of Kohonen Network's efficiency is that; it doesn't need to adjust all the weights. The adoption is required to stable the network and to reach to the nearest of the solution.

  The modified weights will be –

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t)\left(x_i(t) - w_{ij}(t)\right)$$

  where $\eta(t)$ is a gain factor which varies between 0 and 1. Above this equation ensures that the weights in the neighborhoods become more similar to the input and this procedure requires much iteration. But for the network properly converges to a solution the iterations must be limited. So, we adapt the weight according to a linear decreasing function with the number of iterations through the training set. The purpose of the *gain term* is to optimize the adoption of weights. That means, when the difference between input and weight is very large, weight changes heavily. Otherwise it changes smoothly.

- The process will be continued until the minimum distance will reach to a very small value. That means is it goes under a given small value, the process will be stopped. We used 0.00001 as the limiting value. Main target of the weight adoption process is to convert the system such that the winner node have the distance less than or equal to the limiting value.

This is the way how a system is learned from the input pattern. Now the adopted weight set will be stored in a file for further recognition.

5.7 Basic Flow Diagram of Learning Phase

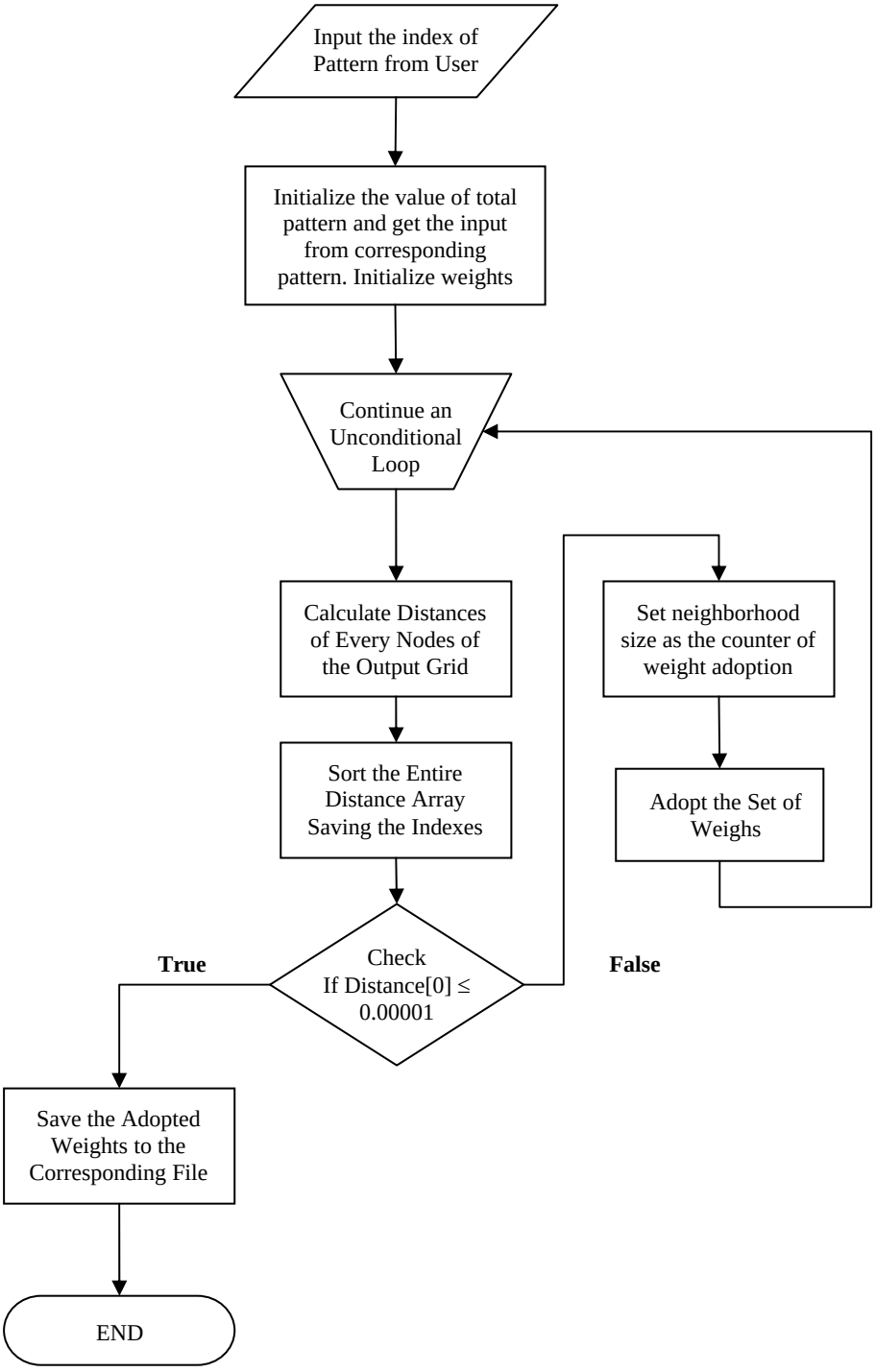


Fig 5.13 : Flow Diagram of Learning Phase

5.8 Recognition Of Face

Recognition process is simpler and faster. It just tests the user input pattern with the stored weights and verify the selected winner node. It is essential to match both the selected nodes; that is node found during learning phase is essential to be equal with the node found during recognition.

Steps to recognize a face is as follows –

- Take the user input pattern which is to be recognized. The pattern must be processed by Matlab first and need to be saved in the disk.
- The weight set and winner node index are taken from the first file from the directory. Total number of face learned can be found by counting the number of files in that directory. It is needed to test the input with all weight sets.
- Fit the weight set to a network with the input set from user. Then calculate distances of all nodes of the grid using the same formula. Then find the minimum distance among all and check the index of that node with the index of the node found from stored file.
- If the both indexes match, check the distance found. If the distances remains under a certain level; then it is proved that the face pattern is known to the system. Practically the limiting value is set to 5.00. That means distance must be lower than the value to recognize any pattern.
- If the condition does not satisfy, then take the weight set from the next file and test with the input pattern on the same way. The process will be continued until all the weight files are tested or a desired result is found.

5.9 Flow Diagram of Recognition Phase

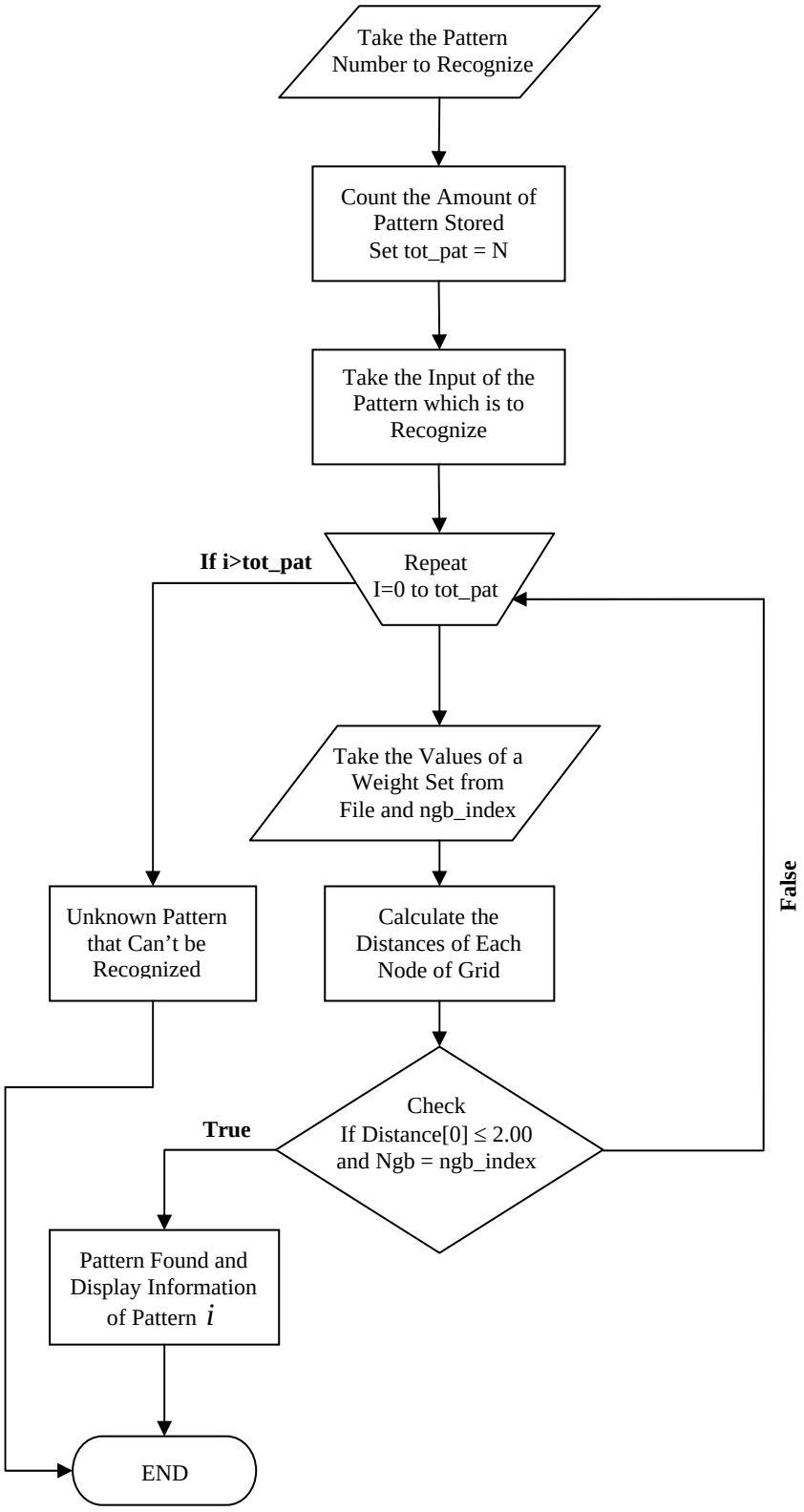


Fig 5.14 : Flow Diagram of Recognition Phase

Chapter 6 : Result and Discussion

6.1 Introduction

This chapter is concerned with the output of our testing. Performance of this system is depends on how many expressions of a single person it can recognize and the percentage of error occurred during the identification a particular learned face. We performed testing of the two types of testing –

1. Testing by individual faces
2. Testing by different expressions of a single person

During the first type of testing, we calculated the minimum distances of each pattern tested with every pattern. Then the result is plotted onto a graph. We also highlighted the winner nodes during learning and that of recognition phase. The actual possibility to match a face with the other learned faces depends on the minimum distance and winner node found during recognition. In our research, if the minimum distance is less than 2.0 and winner node (found during learning) is within the 15^{th} smallest distances, is sufficient to match a face.

On the second type of testing, we performed testing with several expressions of a single person. Suppose; there are seven facial expressions. First we learned six expressions except the first one. Then we tried to match the first pattern with the other sixes. We measured the data(s) like the previous testing and listed. From the table, we found the percentage of similarity of first pattern compared with the others. If any pattern fulfills the both conditions described before, it seems to be matched with the first pattern. And if any pattern other than the first pattern matches with the first one, it means that the system can recognize the pattern i.e. the person. Similarly we tested all other facial expressions on the same way.

6.2 Result Tested by Individual Facial Patterns

User Input Pattern	*Pattern Number*	*Minimum Distance*	*Winner Node*		*Position of the Node after Recognition*	*Result*	*Similarity (%)*
			Learning	*Recognition*			
1	1	0.0000	26	26	0	Matched	100.0
	2	6.4119	25	34	2	Unmatched	0.000
	3	10.6754	23	2	15	Unmatched	0.000
	4	10.7802	16	14	16	Unmatched	0.000
	5	8.0301	18	48	4	Unmatched	0.000
	6	8.0779	38	38	1	Unmatched	0.000
	7	8.7358	39	33	20	Unmatched	0.000

Table 6.1 : Result of Comparison with Pattern 1

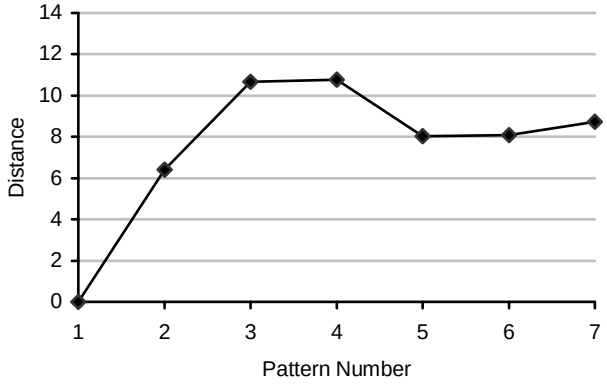


Fig 6.1 : Graph of Table 6.1

User Input Pattern	*Pattern Number*	*Minimum Distance*	*Winner Node*		*Position of the Node after Recognition*	*Result*	*Similarity (%)*
			Learning	*Recognition*			
2	1	6.4120	26	94	7	Unmatched	0.000
	2	0.0000	25	25	0	Matched	100.0
	3	9.8907	23	2	19	Unmatched	0.000
	4	12.3824	16	1	9	Unmatched	0.000
	5	7.6990	18	48	2	Unmatched	0.000
	6	8.9124	38	8	4	Unmatched	0.000
	7	9.2947	39	99	19	Unmatched	0.000

Table 6.2 : Result of Comparison with Pattern 2

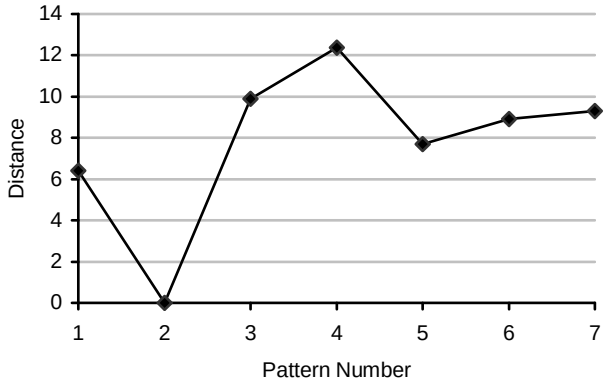


Fig 6.1 : Graph of Table 6.2

User Input Pattern	*Pattern Number*	*Minimum Distance*	*Winner Node*		*Position of the Node after Recognition*	*Result*	*Similarity (%)*
			Learning	*Recognition*			
3	1	10.6758	26	81	5	Unmatched	0.000
	2	9.8910	25	84	3	Unmatched	0.000
	3	0.0000	23	23	0	Matched	100.0
	4	8.2550	16	32	2	Unmatched	0.000
	5	7.8021	18	18	1	Unmatched	0.000
	6	6.6178	38	45	3	Unmatched	0.000
	7	9.4013	39	33	15	Unmatched	0.000

Table 6.3 : Result of Comparison with Pattern 3

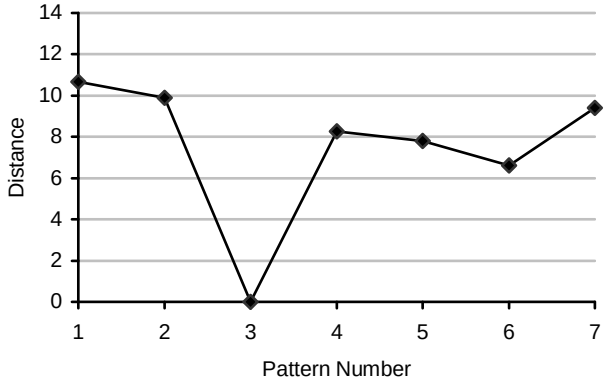


Fig 6.3 : Graph of Table 6.3

User Input Pattern	*Pattern Number*	*Minimum Distance*	*Winner Node*		*Position of the Node after Recognition*	*Result*	*Similarity (%)*
			Learning	*Recognition*			
4	1	10.7807	26	39	8	Unmatched	0.000
	2	12.3828	25	25	1	Unmatched	0.000
	3	8.2551	23	30	18	Unmatched	0.000
	4	0.0000	16	16	0	Matched	100.0
	5	7.9884	18	88	11	Unmatched	0.000
	6	6.6463	38	34	11	Unmatched	0.000
	7	9.7113	39	33	16	Unmatched	0.000

Table 6.4 : Result of Comparison with Pattern 4

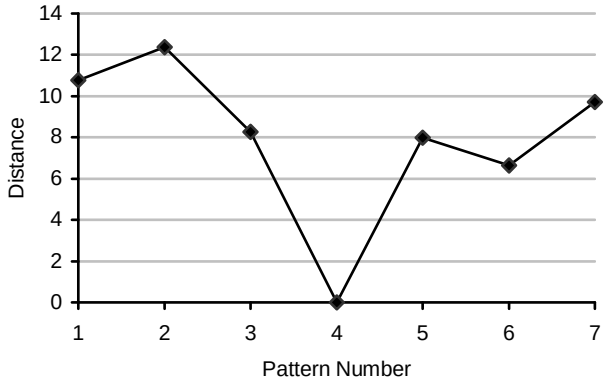


Fig 6.4 : Graph of Table 6.4

6.3 Result Tested by Different Facial Expressions

User Input Pattern	*Expr. Number*	*Minimum Distance*	*Winner Node*		*Position of the Node after Recognition*	*Result*	*Similarity (%)*
			Learning	*Recognition*			
1	2	1.5658	26	31	8	Matched	21.710
	3	0.7059	25	26	14	Matched	64.705
	4	1.6923	23	91	4	Matched	15.385
	5	4.4223	16	18	1	Unmatched	0.000
	6	4.3154	18	12	3	Unmatched	0.000
	7	4.2884	38	96	11	Unmatched	0.000

Final Result : Recognized and Best Match with Pattern 3

Table 6.5 : Result of Comparison with Expression 1

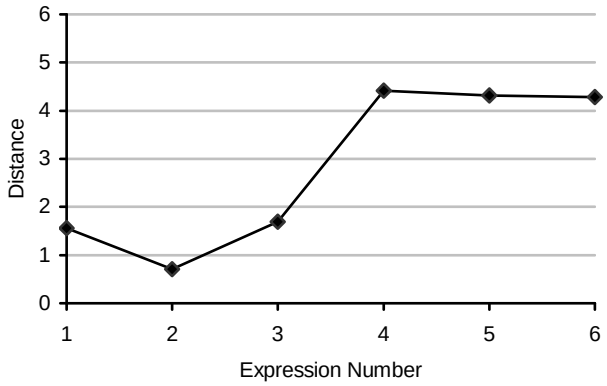


Fig 6.5 : Graph of Table 6.5

User Input Pattern	*Expr. Number*	*Minimum Distance*	*Winner Node*		*Position of the Node after Recognition*	*Result*	*Similarity (%)*
			Learning	*Recognition*			
2	1	1.5658	26	41	19	Unmatched	0.000
	3	1.3292	25	26	9	Matched	33.54
	4	1.7449	23	25	17	Unmatched	0.000
	5	3.1547	16	18	1	Unmatched	0.000
	6	4.3626	18	22	8	Unmatched	0.000
	7	3.6080	38	96	10	Unmatched	0.000

Final Result : Recognized and Best Match with Pattern 3

Table 6.6 : Result of Comparison with Expression 2

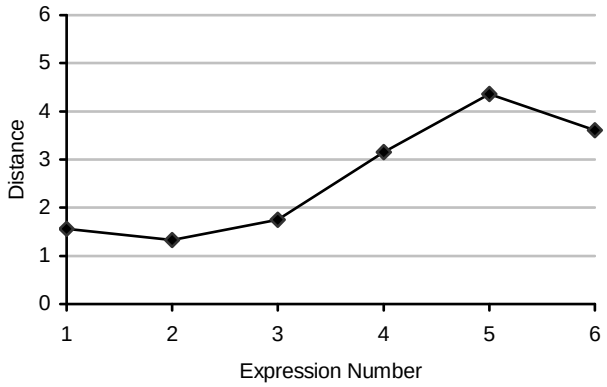


Fig 6.6 : Graph of Table 6.6

User Input Pattern	*Expr. Number*	*Minimum Distance*	*Winner Node*		*Position of the Node after Recognition*	*Result*	*Similarity (%)*
			Learning	*Recognition*			
3	1	0.7059	26	55	8	Matched	64.705
	2	1.3292	25	16	14	Matched	33.54
	4	1.6661	23	11	4	Matched	16.695
	5	4.1431	16	18	1	Unmatched	0.000
	6	4.5422	18	12	3	Unmatched	0.000
	7	4.2126	38	96	11	Unmatched	0.000

Final Result : Recognized and Best Match with Pattern 1

Table 6.7 : Result of Comparison with Expression 3

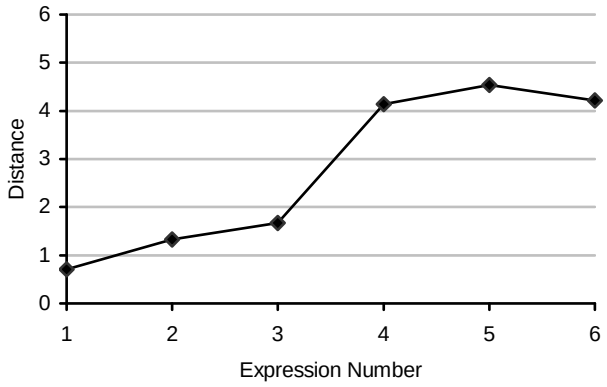


Fig 6.7 : Graph of Table 6.7

User Input Pattern	*Expr. Number*	*Minimum Distance*	*Winner Node*		*Position of the Node after Recognition*	*Result*	*Similarity (%)*
			Learning	*Recognition*			
4	1	1.6923	26	24	11	Matched	15.385
	2	1.7499	25	79	6	Matched	12.505
	3	1.6661	23	33	6	Matched	16.695
	5	4.9874	16	18	1	Unmatched	0.000
	6	5.1213	18	45	7	Unmatched	0.000
	7	4.2319	38	96	3	Unmatched	0.000

Final Result : Recognized and Best Match with Pattern 3

Table 6.8 : Result of Comparison with Expression 4

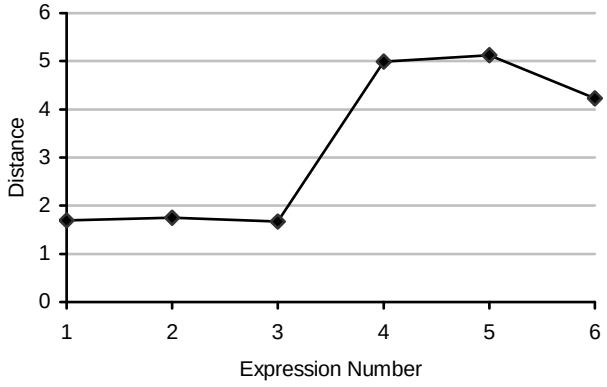


Fig 6.8 : Graph of Table 6.8

User Input Pattern	*Expr. Number*	*Minimum Distance*	*Winner Node*		*Position of the Node after Recognition*	*Result*	*Similarity (%)*
			Learning	*Recognition*			
5	1	4.4223	26	55	11	Unmatched	0.000
	2	3.1547	25	85	6	Unmatched	0.000
	3	4.1431	23	65	6	Unmatched	0.000
	4	4.9873	16	9	1	Unmatched	0.000
	6	1.7661	18	91	7	Matched	11.695
	7	1.3171	38	27	3	Matched	34.145

Final Result : Recognized and Best Match with Pattern 7

Table 6.9 : Result of Comparison with Expression 5

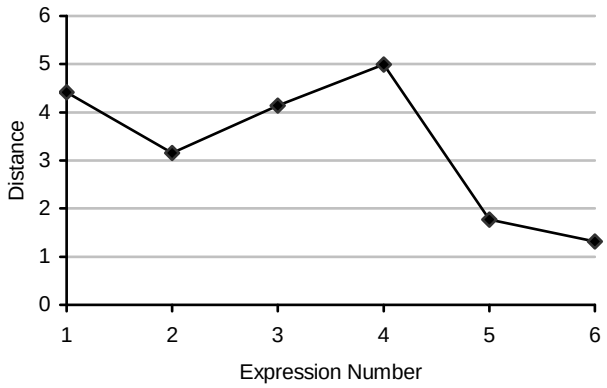


Fig 6.9 : Graph of Table 6.9

User Input Pattern	*Expr. Number*	*Minimum Distance*	*Winner Node*		*Position of the Node after Recognition*	*Result*	*Similarity (%)*
			Learning	*Recognition*			
6	1	4.3154	26	41	15	Unmatched	0.000
	2	4.3626	25	85	3	Unmatched	0.000
	3	4.5422	23	29	15	Unmatched	0.000
	4	5.1212	16	56	6	Unmatched	0.000
	5	1.7661	18	49	5	Matched	11.695
	7	1.5022	38	57	12	Matched	24.89

Final Result : Recognized and Best Match with Pattern 7

Table 6.10 : Result of Comparison with Expression 6

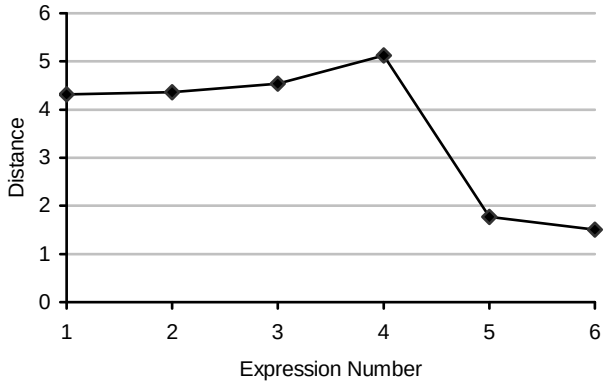


Fig 6.10 : Graph of Table 6.10

User Input Pattern	*Expr. Number*	*Minimum Distance*	*Winner Node*		*Position of the Node after Recognition*	*Result*	*Similarity (%)*
			Learning	*Recognition*			
7	1	4.2884	26	2	12	Unmatched	0.000
	2	3.6080	25	85	5	Unmatched	0.000
	3	4.2120	23	25	19	Unmatched	0.000
	4	4.2319	16	56	12	Unmatched	0.000
	5	1.3171	18	18	0	Matched	34.145
	6	1.5023	38	61	15	Matched	24.885

Final Result : Recognized and Best Match with Pattern 5

Table 6.11 : Result of Comparison with Expression 7

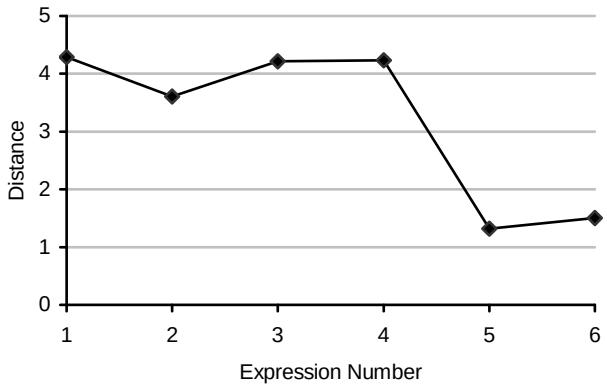


Fig 6.11 : Graph of Table 6.11

6.4 Discussion

From table 6.1 to table 6.4, we found that, if we learn the network with only a single facial expression of different persons and also try to recognize with a pattern within those faces, then the 100% accuracy can be achieved. So, in this case, our system is completely efficient.

But it differs, when we try to recognize the face of a person with various facial expressions those are not learned. We used the maximum limit of distances to two, which is the scale of our performance measurement. The formula to measure the similarity of an expression with any other is –

$$Similarity = \frac{2.0 - \text{Minimum Distance}}{2.0} \times 100\%$$

We learned six different facial expressions of one person. And then tested with another expression which was not learned. We found that, this expression matched with at least one expression other than that. Similarly we performed the same processes six times and observed that, the tested expressions matched with at least one expression. From the table 6.5 - 6.11, we found that the percentage of similarity ranges between 11.695 and 64.705. Comparing with any system accuracy we found that our system is very inefficient. But if we tested the facial expression of another person what was not learned yet then we found the percentage of similarity was 0%. So, in this respect we can tell that our system is completely efficient.

6.5 Limitations

No system is out of limitations. Though from the result, we found that our system is very efficient; but it has some bindings. Such types of limitations are enlisted below –

1. The *face detection algorithm* is not so much efficient. This algorithm considers hairs and other minor parts of an image which are not essential for the work. As for example, any face pattern with rippled hair or wearing hat may not effective for face recognition.
2. The image scaling algorithm we used; is not so powerful. If the compression ratio is high, it is difficult to detect feature from the face pattern. It hides some essential features from the pattern.

3. In our system, it is essential to consider the patterns with light background. But it can’t be always assured that the background must be light. Due to the reason, same person can’t be identified at night if his/her face is learned with an image captured at the day.

Chapter 7 : Conclusion and Future Developments

7.1 Conclusion

Till now, no intelligent system has been developing which gives hundred percent correct outputs. Always there occur some errors due to the variation of input patterns. Face recognition is a very complicated system because human faces change depending on their age, expressions etc. There are a lot of expressions for a human being. So it is not possible to learn all types of expressions into the network. As a result there is a cause for the unrecognizing. Moreover due to the performance variations of the input device face can not be detected correctly and pattern may change extremely. In this paper we have developed and illustrated a recognition system for human faces using Kohonen self-organizing map (SOM). The main objective of our face recognition system was to obtain a model that is easy to learn i.e. minimization of learning time, react well with different facial expressions with noisy input and optimize the recognition as possible. In our system we used Kohonen self organizing map instead of Backpropagation technique for learning because Backpropagation takes several hours to learn while Kohonen self organizing map takes only a few seconds to learn. So, in this respect our system is very efficient and easy to learn

7.2 Recommendation for Future Development

Our main plan is to recover the limitations described before. We want to develop an efficient algorithm for face detection and image scaling. The system is to not real time. So it needs to acquire and process image manually. Our next plan is to develop a real time system which will acquire and process images automatically and convert it into the raw format.

The system is almost inefficient for complex expression angled or flipped (vertical or horizontal) pattern. We wish to moderate our system in such a way that it can recognize such type of expressions.

Moreover we wish to detect and recognize a facial image in motion. It is almost difficult to detect face from a moving picture. But we wish to perform the work in future.

Bibliography

[1]. Carey, S., and Diamond, R, *"From Piecemeal to Configurational Representation of Faces"*, Science 195, pp. 312 313, (1977).

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

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

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

[5]. Yuille, A. L., Cohen, D. S., and Hallinan, P. W., *"Feature Extraction from Faces Using Deformable Templates"*, Proc. of CVPR, (1989).

[6]. Kohonen, T., *"Self-Organization and Associative Memory"*, Berlin: Springer-Verlag, (1989).

[7]. Kohonen, T., and Lehtio, P., *"Storage and Processing of Information in Distributed Associative Memory Systems"*, (1981).

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

[9]. Kanade, T., *"Picture Processing System by Computer Complex and Recognition of Human Faces"*, Dept. of Information Science, Kyoto University, (1973).

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

[11]. Commercial Systems and Applications Wed Nov 27 18:01:36 EST 1996

<URL: http://www-white.media.mit.edu/tech-reports/TR-516/node10.html>

[12]. Rafael C. Gonzalez and Richard E. Woods, *"Digital Image Processing"*, First Edition, 1992.

[13]. *Artificial Neural Networks Technology*

<URL: http://www.dacs.dtic.mil/ neural_nets.html>

[14]. Simon Haykin, *"Neural Networks, a Comprehensive Foundation"* (1999), Pearson education, Inc, Singapore.Chapter 1.9: Historical Notes.

[15]. J G Taylor, *"Hand Book for Neural Computation"*, IOP publishing Ltd. Oxford university press, release1997, pp 1-7.

[16]. COGANN-92, *"International Workshop on Combination of GA and NN"*, Baltimore, Maryland, USA-1992.

[17]. *Industrial Applications of Neural Networks* (research reports Esprit, I.F.Croall, J.P.Mason)

[18]. *Artificial Neural Networks in Medicine*, 2 Jul 03

<URL:http://www.emsl.pnl.gov/proj/neuron/workshops/WEEANN95/talks/burke.weeann95.abs.htm>

[19]. *A Novel Approach to Modeling and Diagnosing the Cardiovascular System*, 02 Jul 03

<URL: http://www.emsl.pnl.gov/proj/neuron/papers/keller.wcnn95.abs.html>

[20]. *Electronic Noses for Telemedicine*, 03 Mar 98

<URL: http://www.emsl.pnl.gov/proj/neuron/workshops/TAC95-nn/talks/ keller.tac95.html>

[21]. Christos Stergiou and Dimitrios Siganos, "*NEURALNETWORKS"*.

<URL:http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report.html

[22]. R. Beale and T. Jackson, "*Neural Computing: An Introduction*", First Edition, MIT press, 1990.