Chapter No 1
INTRODUCTION

1.1 Introduction to Recognition

Recognition of human individuals involves physical recognition, such as visual, auditory, or behavior recognition. The combination of visual and additive recognition is even more effective and often removes any doubts.

Biometrics are broadly divided in to two major categories, behavioral biometrics and physiological biometrics. Behavioral biometrics depends upon the action of the user or individual like speaking, writing, and gait. All of these types are affected by the fatigue, time, age as well by the health factor. Whereas the physiological biometrics depends upon some physical features of individual like finger prints, iris, face recognition, hand gesture recognition and so on, which are mostly stable over the time [1,2]. So signature recognition is from the category of behavioral biometric which the writer learns and acquire over a period of time and finally that signature become the identity of an individual.[3,1]

There exist a number of biometrics methods today e.g. Signatures, Fingerprints, Iris etc. There is considerable interest in authentication based on handwritten signature verification system as it is the cheapest way to authenticate the person. Fingerprints and Iris verification require the installation of costly equipment's and hence cannot be used at day to day places like Banks etc [4].

There are two methods to recognize the signature of a person, one is Online Signature Recognition and other is Offline Signature Recognition. For the documents authentication, authorization and verification the handwritten signatures are widely used. Most of the documents like passports, official documents, and academic certificate as well the bank cheques are authorized by the handwritten signatures. In today's modern society where fraud is widespread, there is need of a strong HSV(Handwritten Signature Verification) system to complement visual verification. Offline Signature Recognition and Verification by Neural Networks (OSRVN) is a new field appeared in last decade as an important branch of image processing. It offered a new way for authenticating the signatures using Neural Nets.

1.2 Proposed System

The process of recognition is the identification of something already known or acknowledgement of something as valid the state or quality of being recognized or acknowledged. It is broadly divided into two phases, identification of object and verification of the object. In proposed system the object is signature which we will get from the scanned image

.

1.2.1 Preprocessing

The preprocessing stage includes following four steps

1. Background Elimination
2. Noise Reduction
3. Width Normalization (all signatures must have the same height and width)
4. Thinning

1.2.2 Features Extraction

Following features are extracted from every signature

1. Normalized Area of signature
2. Accepts ratio
3. Center of gravity

4. Slop of the line joining the centers of gravity
5. Crop
6. Maximum Horizontal Projection
7. Maximum Vertical Projection
8. Edge Detection
9. Texture Features

1.2.3 Verification

The gathered signatures with their features are used for the training and testing phase of the verification. With the help of neural net we will try to check the validity of signature, whether the test signature is genuine or forgery.

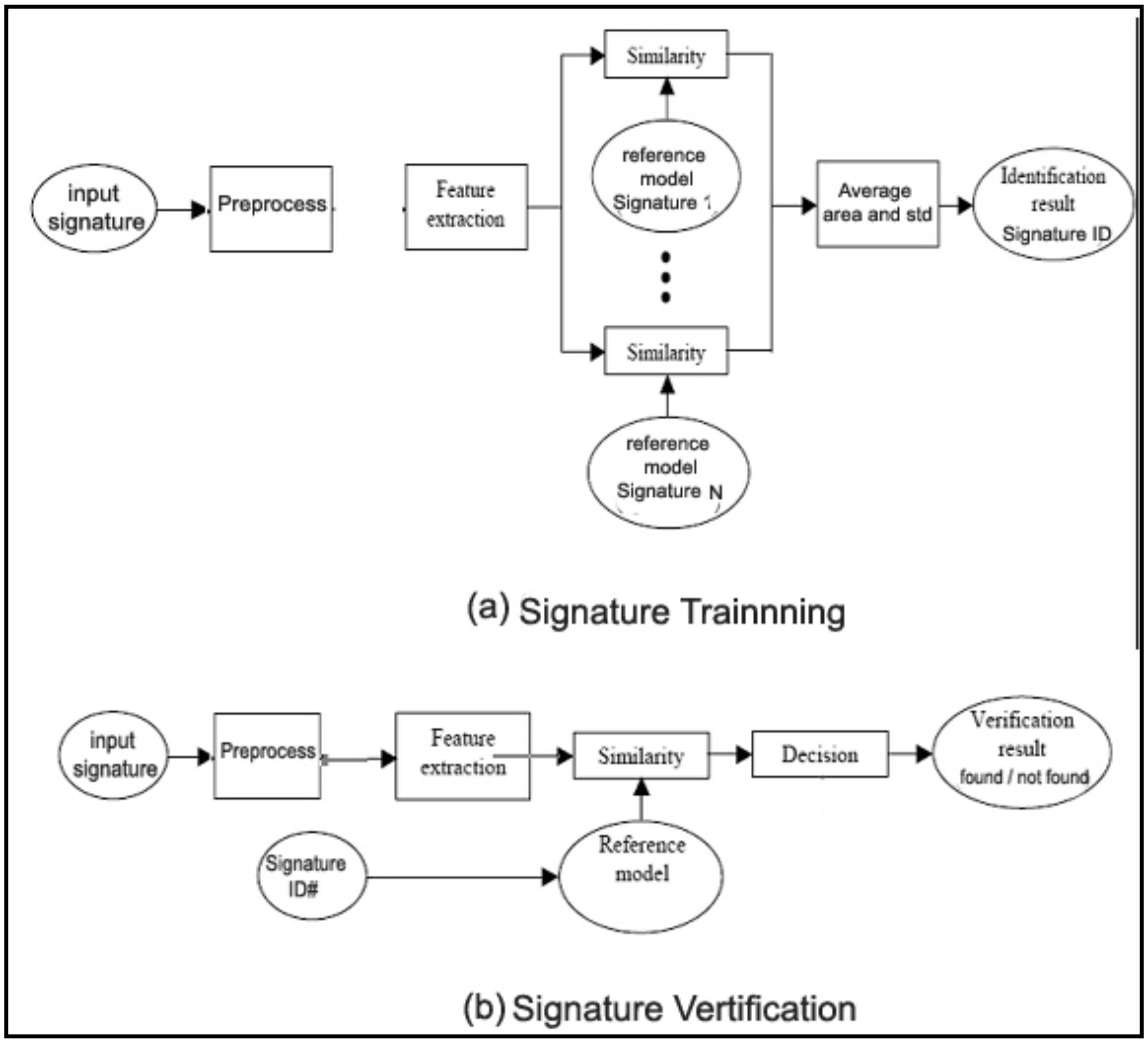


Figure (1-1) Training and Verification Process

1.3 Objectives of the thesis

1.3.1 General Objectives of the Thesis

Viewing the literature, identifying the merits and demerits of previous work, another recognition system is proposed using single layer network and multiple neurons. The proposed work focus on three types of features (Global Features, Texture Features, Geometrical Features and Grid Information Features).

1.3.2 Specific Objectives of the Thesis

The specific objectives of this research thesis are:

a) Collection of handwritten signature samples to be used as training and testing set.
b) Feature extraction of all collected signatures.
c) Testing the accuracy of verifier by using Neural Networks.

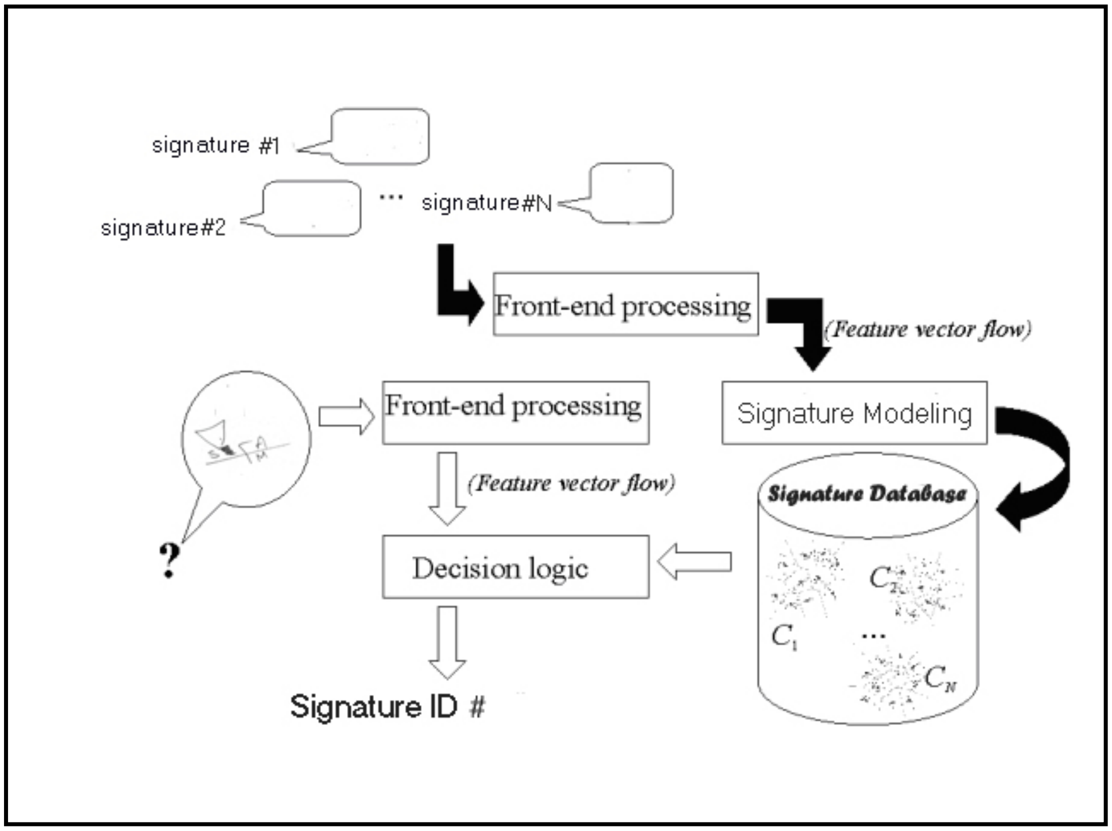


Figure (1-2) Goal of Signature Recognition

1.4 Scope

The project dealt with static images of handwritten signatures. The genuine signature samples taken from known writers and the forgery set are generated imitating the genuine set. Global, Texture and Geometrical Features are used as signature image descriptors.

1.5 Data Collection and Methodology

Both the qualitative and quantitative methodologies are going to be used during this research. Literature study, algorithm analysis, comparing the complexities of algorithms is the methods we are going to use in our study.

We can consider a block diagram for this system as follows in Figure (1-3).[11]

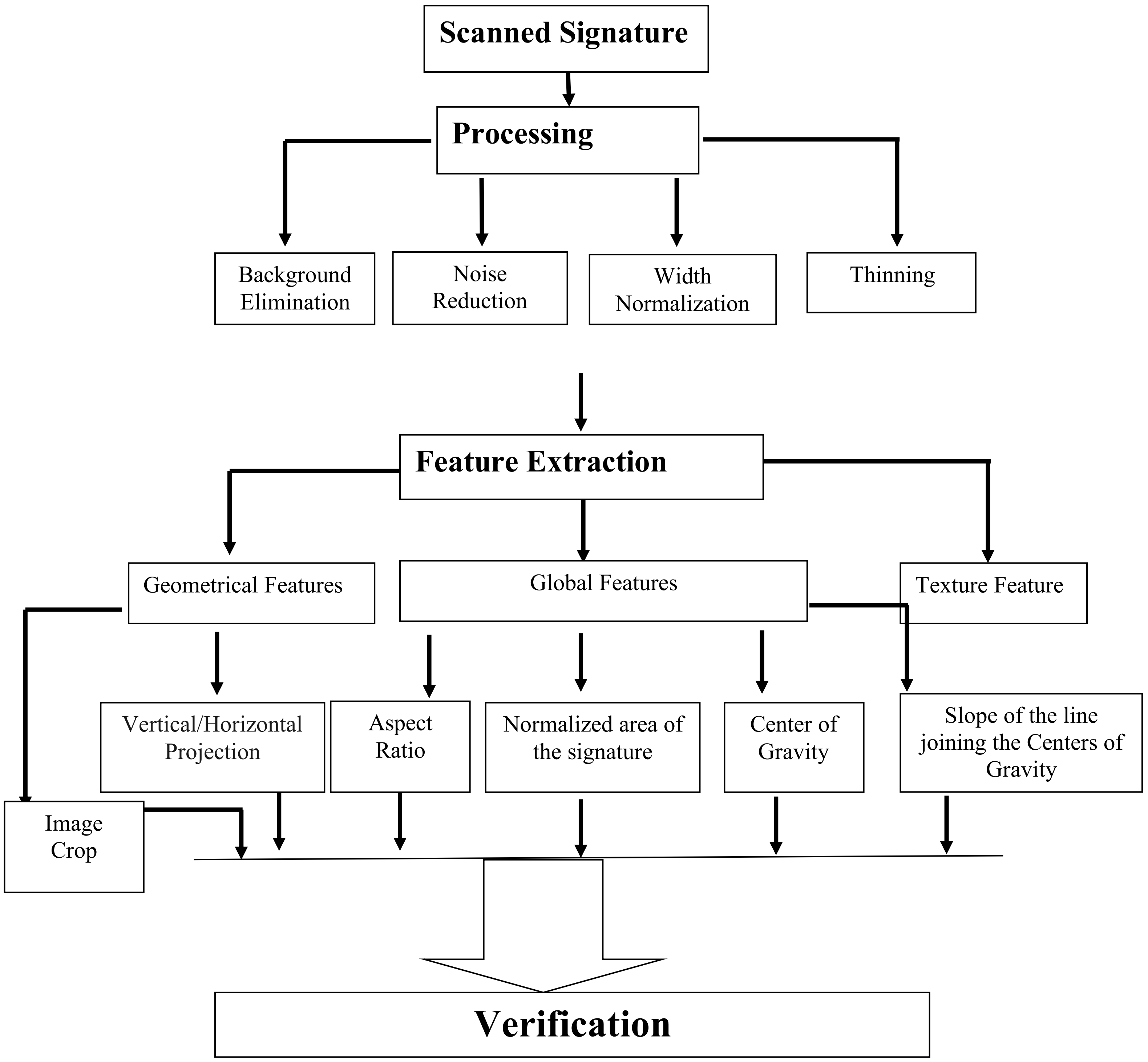


Figure 1.3: Steps for Off Line Signature Recognition

Chapter No. 2
BACKGROUND

Signature verification is one of the important fields to verify the individual. Vigorous research been done on signature verification from several decades, but still the new ideas are explored for Signature verification (SV). This section includes the review of related research papers, which represents the recent work on Handwritten Signature Verification (HSV). There many techniques for signature verification. These techniques are distributed in different categories.

1. Hidden Markov Model
2. Graph Matching
3. Support Vector Machine
4. Fuzzy Logic Method
5. Neural Network Based Model

2.1 Hidden Markov Model

In [20], a hidden markov model is introduced which uses only global features. The base of this model is to calculate a discrete randon transform for each binary signature image at range of 0-360º, which is a function of total pixel in image as well the intensity per given pixel calculation. The HMM is used to model each writer signature. The method achieves an AER of 18.4% for a set of 440 genuine signatures of 32 persons.

The idea describe in [21], of Justino et al , uses only graph metric features, that is density of pixels and pseudo dynamic features represented by axial slant. They also apply the grid features which contain the vertical and horizontal projection of signatures. A HMM is used for learning and verification process.

2.2 Graph Matching Method

Another way of signature recognition is graph matching that avoids the use of features. Abuhaiha [22] introduced graph matching method, uses only raw binary pixel intensities. A binary image is represented as a graph with a set of edges and vertices. The goal is to get the minimum cost of matching which is represented as assignment problem in graph theory. This method test 75 signatures for skilled forgery and 300 signatures for random forgery. The FAR value is 26.7% and 5.6%, whereas EER for skilled and randon forgeries is 26.7% and 5.6% respectively.

2.3 Support Vector Machine

SVMs uses high dimensional feature space and estimate differences between classes of given data to generalize unseen data. The proposed system of [23] uses global, grid and directional features of the signature and SVM for verification. The data base consists of 1320 signatures from 70 writers. 40 writers are used for training with each signing 8 signatures thus total of 320 signatures for training. For initial testing the system uses 8 original signatures and 8 forgeries achieves FRR 2% and FAR 11%.

2.4 Fuzzy Logic Approach

In [24] the authors proposed a system that extracts angle features that are modeled into fuzzy model based on Takagi-Sugeno model. This approach includes structural parameters that account for variation in writers styles and changes mood. The performance rate of this approach is 70%.

2.5 Neural Network Based

In [5], a system is proposed that uses only global and grid features of the signature. For verification and recognition an artificial Neural Network which based on back propagation algorithm is used. The system consists of a data base of 400 test signatures samples, having genuine and forgery signatures of twenty different persons. The learning rate includes the analysis of FAR and FRR. With this system the FRR of less than 0.1 and FAR of less than 0.2 are achieved.

The approach of [6] uses global, texture and grid features. The proposed system is divided into two stages which use two different types of neural network structures. The first stage uses two stages Perceptron OCON (one-class-one-network). In the first stage the classifier combines the decision results of the neural networks and the Euclidean distance obtained using the feature sets. The second phase uses the results of first stage and a radial base function (RBF) neural network structure is used to conclude the final decision. The system was tested and yielded high recognition and verification rates.

The method used in [7], performs the identification by structural features of the signature. Features are extracted by the use of novel combination of Modified Direction Features (MDF). The verification stage consists of two different neural networks. For test and train the signature the modified direction features are used in combination with additional distinguishing features. Radial Base Function (RBF) and Resilient Back Propagation (RBP) are used for neural network. For training and testing, the data base consists of 2106 signatures, which contains 936 genuine and 1170 forgeries samples. The verification rate obtained by this proposed system is 91.12%.

Another approach describe in [8], also uses neural network for the off-line signature verification. Grid, Mask and Global features are used for the identification phase of the signature. For the verification phase the Back Propagation neural network is used. The success rate is 100% only for all trained signatures. However, it has a poor performance rate with untrained signatures.

Chapter No. 3
Proposed System

3.1 Preprocessing

The basic entity for this work is signatures. The signatures were collected from different people on white paper with ordinary pen. Those collected signatures were scanned with the help of scanner. The obtained signatures contain noise and some unwanted data, therefore all the sample signatures firstly should be prepare for the further stages.To remove the noise and errors from the signatures images, preprocessing is to be performed which includes following four steps.

1. Background elimination
2. Noise reduction
3. Width normalization
4. Thinning

3.1.1 Background Elimination

Background elimination has two steps:

1. Convert the RGB image to Grey Scale
2. Convert the Grey Scale Image to Binary

3.1.1.1 Grey Scale Image

A gray scale image is also called the monochromatic image, which means only one (mono) color (chrome).A grey scale image or gray level image is simply one in which only colors are gray shades. The reason to convert the obtained original RGB image to gray scale is that less information needs to be provided for each pixel. The numerical range of gray scale image is from 0 (black) to 1(white), having different shades of gray in between the range of zero and one.

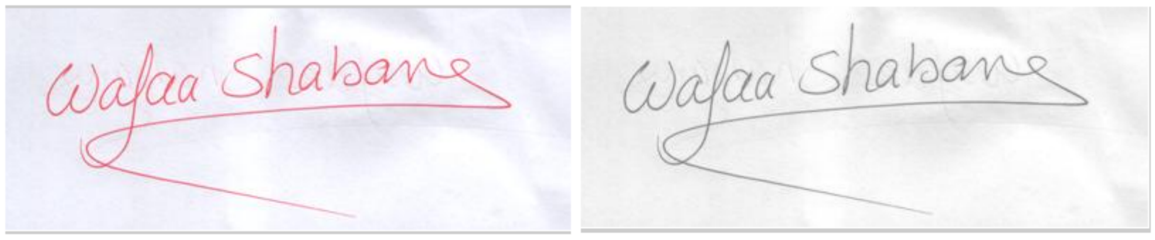


Figure (3.1)RGB to Grey Scale Image

3.1.1.2 Binary Image

A binary image is the simplest type of images. The gray scale image is further processed by converting it to binary. A binary image has been quantized to two values, usually denoted 0 and 1, but often with pixel values 0 which represents to black and 255 for white.

Binary images are used in many applications domains like identifying the objects on a conveyor, interpreting text as well to identify the orientation of objects.

Binary images are the simplest images since they are easy to process as compare to other image types because of their simple nature. These images are obtained by thresholding a grayscale image. The output image replaces all pixels in the input image which has luminance value greater than level with the value 1(white) and replaces rest of the pixels with the value 0(black). The range of levels is normally [0,1].

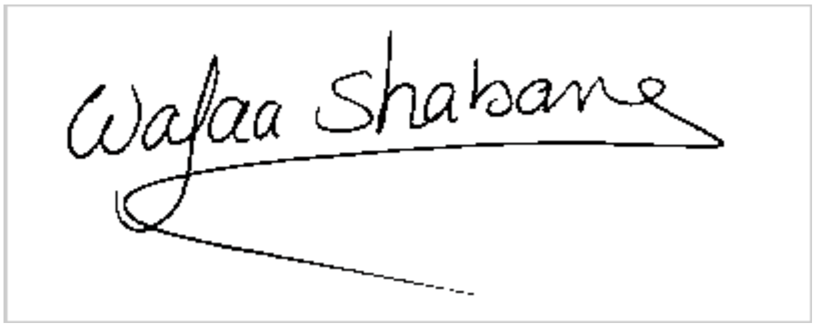


Figure (3.2) Grey Scale Image to Binary Image

3.1.2Noise Elimination

Almost all digital images consist of variety of noise. Noise is result of errors in the image acquisition process. The purpose of this stage is to remove the noise as much as possible.

In this stage a single white pixels on black background and a single black pixel on a white background is to be eliminated. In order to obtain this, we make a 3 by 3 mask to the image with a simple decision rule: if the number of the 8- neighbors of a black pixel is greater than white, the chosen pixel is considered to be black ,otherwise it will be black [6].

3.1.2.1 Median Filter

Median filtering is a nonlinear operation often used in image processingto reduce "salt and pepper" noise. A median filter is more effective thanconvolution when the goal is to simultaneously reduce noise and preserve edges.

Median filtering is a specific case of *order-statistic filtering*, also known as *rank filtering.* With median filtering, the value of an output pixel is determined by the *median* of the neighborhood pixels, rather than the mean. The median is much less sensitive than the mean to extreme values (called *outliers*). Median filtering is therefore better able to remove these outliers without reducing the sharpness of the image. The medfilt2 function implements median filtering. . [9]

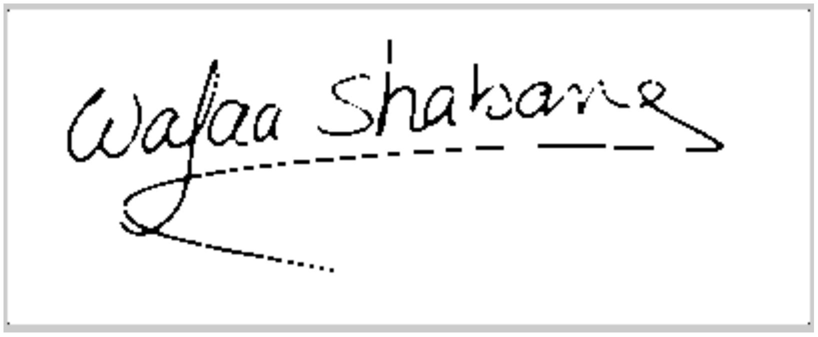


Figure (3.3) Noise Elimination – Median Filter

3.1.3Width Normalization

Almost every type of signatures have inter-personal and intra-personal differences. To remove this difference image size must be standardized to process easily in further stages. A default size is adjusted for every scanned image.

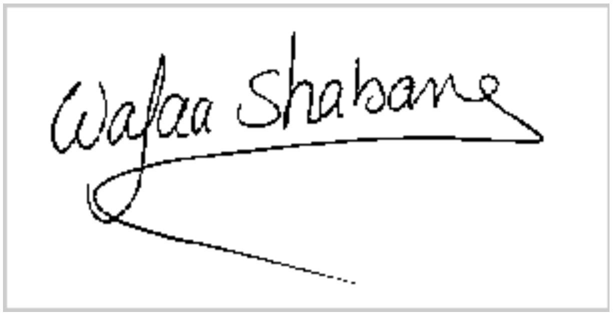


Figure (3.4) Normalized Width

3.1.4Image Thinning

The basic idea behind image thinning is to eliminate the thickness differences of pen by making the image one pixel thick. Another name for image thinning is SKELETONIZATION. It is used to reduce all objects in an image to lines, without changing the essential structure of the image.

`Bwmorph` function is used for Morphological operations on binary images. 'Thin 'operation is used in `Bwmorph function`. It removes pixels so that an object without holes shrinks to a minimally connected stroke, and an object with holes shrinks to a connected ring halfway between each hole and the outer boundary. This option preserves the Euler number.

Figure (3.5) Thinning

3.2 Features Extraction

The second important stage is the extraction of features from every enrolled signature. Every enrolled signature has its own unique features. The system works on extracting the set of eight features to uniquely identifying a candidate signature. These features are generally divided into three major categories (Global Features, Texture Features and Geometrical Features). When extracting these features we can get set of good comparisons to differentiate signatures.

Following features are extracted from every signature

1. **Geometric Features**

   Geometric features are used to extract the structure of a signature.[6]

   1. Aspect ratio
   2. Normalized Area of signature
   3. Center of gravity
   4. Slop of the line joining the centers of gravity

2. **Texture Features**

   Texture features provide information about the overall appearance of signature.[5]

3. **Global Features**

   1. Division of image into two halves.
   2. Maximum Horizontal
   3. Vertical Projection
   4. Edge Detection

3.2.1 Geometric Features

3.2.1.1 Aspect Ratio

Aspect ratio is considered to be the height of a signature. The aspect ratio (A) is the ratio of width to height of the signature. The bounding box coordinates of the signature are determined and the width (Dx) and height (Dy) are computed using these coordinates.

$$\mathbf{A = Dx/Dy} \qquad \text{…………………..}\mathbf{(1)}$$

3.2.1.2 Normalized Area

The ratio of area occupied by the foreground pixels (black pixels) of signature to the background pixels (white pixels) of bounding box of the signature is called the normalized area of signature.

NA = Δ/(DxDy) ……………………(2)

Where Δ is the area of signature pixels.

To get the area feature, first we will find the total image area by using ***bwarea***of MATLAB function and then we find the signature foreground area by using***bwprim*** function of MATLAB. BWPERIM(BW1,CONN) specifies the desired connectivity.

CONN may have the following scalar values:

4	two-dimensional four-connected neighborhood
8	two-dimensional eight-connected neighborhood
6	three-dimensional six-connected neighborhood
18	three-dimensional 18-connected neighborhood
26	three-dimensional 26-connected neighborhood

In MATLAB (***bwarea*** mean compute the area of objects in a binary image, whereas ***bwprim*** means find the perimeter pixels in binary image.)

Figure (3.6) Normalized Area

3.2.1.3 Center of Gravity (COG)

It is the center of an object's weight distribution, where the force of gravity can be considered to act. The COG calculates where the center of image lies.

To calculate the center, we have to use ***find*** function of MATLAB and apply it to given signature. It will return the X and Y Co-ordinate of a signature. Then apply the mean function on these calculate Co-ordinates.

[Cx, Cy] =find(image)

CentriodX=mean(Cx)

CentriodY=mean(Cy) **……………………(3)**

3.2.1.4 Slope of Signature

The slope is a line joining the centers of gravity of the two halves of the signature. To find the slope line, divide the image first into to equal halves; get the center of each half and drawing the line between these two points.

Matlab ***find***() function is used ,to find the indices of nonzero elements.

[YCoord1,XCoord1]= FIND(X,...) returns the row and column indices instead of linear indices into X. This syntax is especially useful when working with sparse matrices.

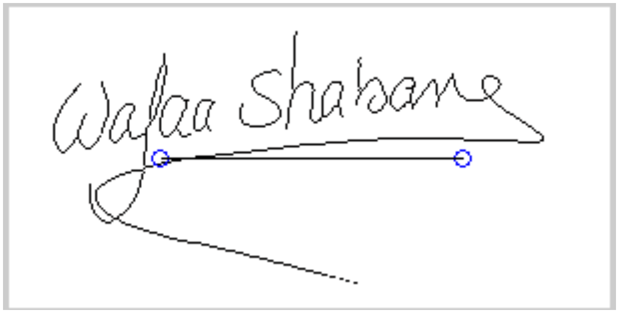


Figure (3.7) Slope

3.2.2 Texture Features

Co-occurrence matrices of signature image are used to extract the texture features of signature. Co-occurrence matrix **P*d* [*i,j*]** is defined by first specifying a displacement vector **d= (dx,dy)**and by counting all pairs of pixels separated by d and having gray level values *i* and *j*. Co-occurrence matrix is a 2 by 2 matrix describing the transition of black and white pixels. It can be defined as

$$\mathbf{P}_d\,[i,j]=\begin{bmatrix} P00 & P01 \\ P10 & p11 \end{bmatrix} \qquad \text{…………………..(4)}$$

Where

P00 is number of times that two white pixels occur, separated by distance d.
P01 is the number of times that a combination of a white and black pixels occurs, separated by distance d.
P10 is same as P01.
P11 is the number of times two black pixels occur, separated by distance d.
The method that is used to calculate the texture features is , image is divided into six rectangular segments (3×2). From every segment the $P_{(0,0)}$,$P_{(1,0)}$,$P_{(0,1)}$ and $P_{(1,1)}$ matrices extraction.

3.2.3 Global Features

3.2.3.1 Maximum Vertical Projection

The Vertical projection n of preprocessed image is calculated by summing all pixels along the columns. It can be defined as [10]:

$$\mathbf{MVP\ (i)=\sum_{i} image(i,j)} \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{(5)}$$

3.2.3.2 Maximum Horizontal Projection

The Vertical projection n of preprocessed image is calculated by summing all pixels along the columns. It can be defined as [10]:

$$\mathbf{MHP(j)=\sum_{j} image(i,j)} \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\text{(6)}$$

3.2.3.3 Edge Points Calculation

An edge point of a signature is a defined point that has only one eight neighbor.

Figure (3.8) Edge Points

EDGE Find edges in intensity image.

EDGE takes an intensity or a binary 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 function has different edge finding methods, canny edge method we used for our work. This method finds the edges of preprocessed image by looking for local maxima of the gradient of image. Using Gaussian filter derivative, gradient is calculated. To detect the weak and strong edges of signature, two thresh holds are being used. It includes the weak points edges in the output only if they are connected to strong edges.

For more refinement fspecial ( ) function is used. FSPECIAL Create predefined 2-D filters. H = FSPECIAL('sobel') returns 3-by-3 filter that emphasizes horizontal edges utilizing the smoothing effect by approximating vertical gradient. If you need to emphasize vertical edges, transpose the filter H: H'.

[1 2 1;0 0 0;-1 -2 -1].

3.2.3.4 Image Cropping

The preprocessed image is cropped into two equal halves and separately determined the centers of gravity of these two halves.

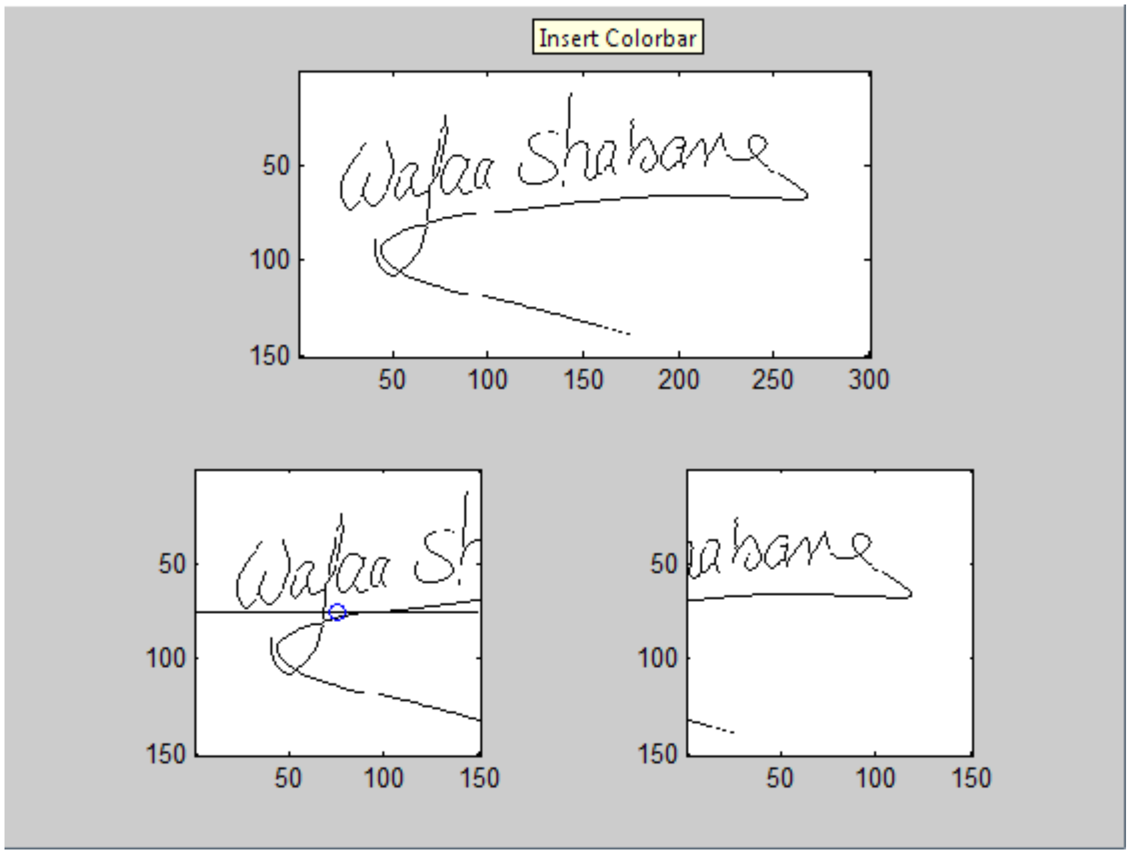


Figure (3.9) Image Crop

Matlab ***imcrop()*** function is an interactive image cropping tool, associated with the image displayed in the current figure, called the target image.

Imcrop [XMIN YMIN WIDTH HEIGHT] **,** XMIN is the starting point of X-axis pixels, YMIN is the starting point of Y-axis pixel from cropped image. WIDTH and HEIGHT are the size of cropped area.

3.3 Neural Network Design

The last stage of our system is neural network design stage.

A neural network consists of number of nodes which has the ability to learn and generalized from the given training set or data. Pattern matching is one of the most prominent tasks of neural networks that provide more fast and efficient results as compare to traditional computer architecture. Neural networks are excellent at developing human made systems that can perform the same type of information processing that our brain performs. [12]

Neural networks are composed of simple elements operating in parallel. These elements are inspired by biological nervous systems. As in nature, the network function determined largely by the connections between elements. We can train a neural network to perform a particular function by adjusting the values of the connections (weights) between elements.
Commonly neural networks are adjusted, or trained, so that a particular input leads to a specific target output. Such a situation is shown below. There, the network is adjusted, based on a comparison of the output and the target, until the network output matches the target. Typically many such input/target pairs are used, in this *supervised learning*, to train a network [19].

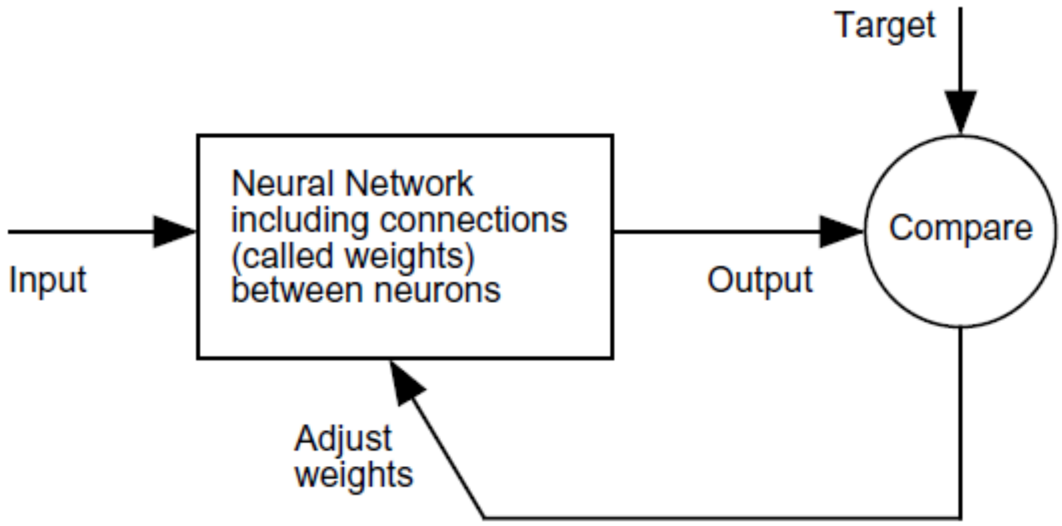

Target
Neural Network including connections (called weights) between neurons
Input
Output
Compare
Adjust weights

Figure (3.10) Network Design

There are many successful applications of neural networks in the variety of areas. For example, Sejnowski/Rosenberg's NETtalk for text to speech conversion[13], Fukushima *etal.*'s "neocognitron" for visually recognizing hand-written Arabic numbers[14], Grossberg'svision processing network to synthesize coherently three-dimensional form. Color, and brightness percepts[15], the control of cart-pole problem by Barto*et al.*[16] and Anderson [17], and the neural network for nonlinear self-tuning adaptive control[18].

3.3.1 Feed Forward Network

Feed forward network is used for verification system. Feedfarward networks often have one or more hidden layers of sigmoid neurons followed by an output layer of linear neurons.

There are four steps in training process.

1. **Assemble the training data**
2. **Create the network object**
3. **Train the network**
4. **Simulate the network response to new inputs**

3.3.1.1 Assemble the training data

In our system, we use single layer and multiple neurons.

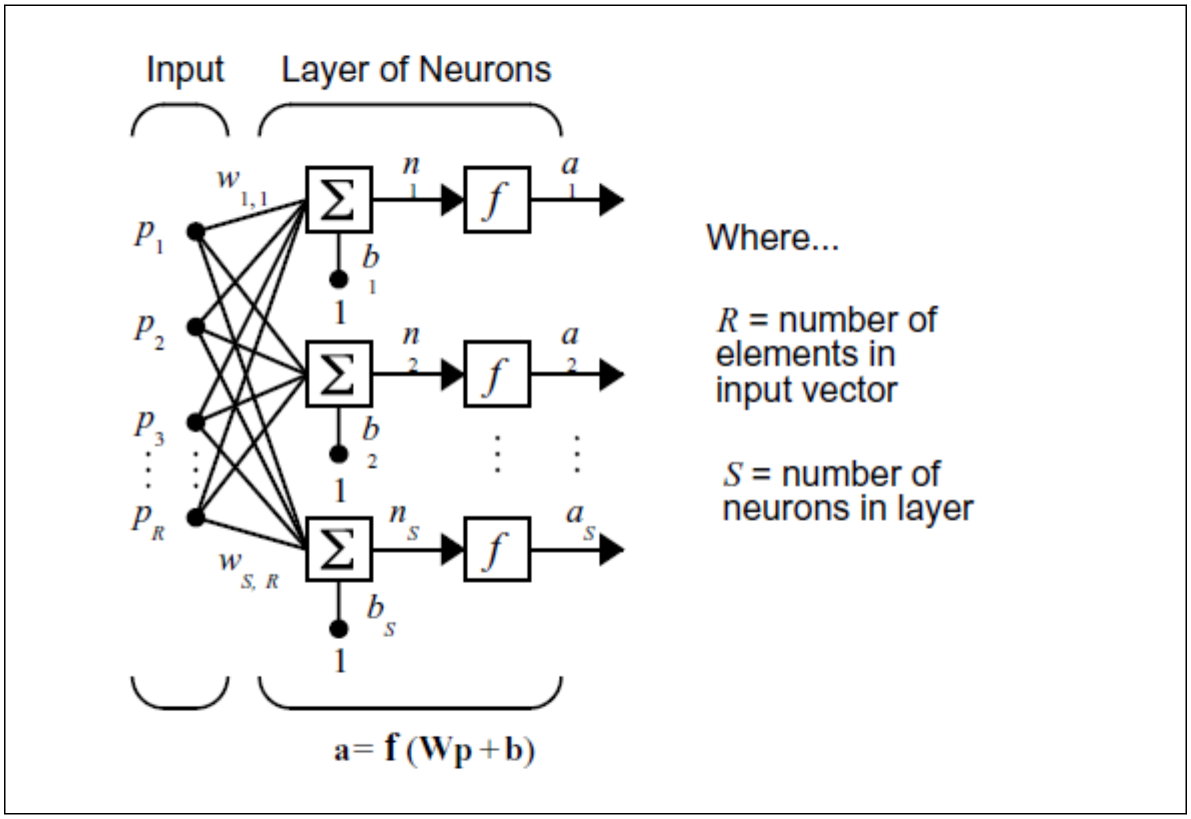

Input
Layer of Neurons
$w_{1,1}$
p_1
p_2
p_3
p_R
$w_{S,R}$
n_1
a_1
b_1
n_2
a_2
b_2
n_S
a_S
b_S
1
f
Where...
R = number of elements in input vector
S = number of neurons in layer
$\mathbf{a} = \mathbf{f}(\mathbf{Wp}+\mathbf{b})$

Figure (3.11) A one-layer network with R input elements and S neurons [19].

3.3.1.2 Network Creation

The first step in training a feed forward network is to create a network object. The MATLAB newff ( ) function is used to create a feed forward network. newff required four input parameters, and returns the network object. The first input parameter is the number of training sample signatures. The second parameter is the supervised output of every input train sample. The third input is a cell array containing the names of the transfer function to be used in layer. The final input contains the name of the training function to be used.

```
net=newff(m,out,[11
1],{'tansig','purelin'},'trainlm','learngdm','mse');
```

in our system “m” is an input matrix, out is the supervised target matrix. The network consists of single layer having eleven neurons. ‘tansig’ function is used for Transfer function of layer, and `'purelin'` for output layer. `'trainlm'` is a back propagation network training function. `'learngdm'` is a back propagation weight/ baise learning function. For the performance function `'mse'` is used.

3.3.1.3 Training

After the creation of network, now it is ready to train. The ***train function*** is used for this purpose.

```
train(net,m,out);
```

train function required three parameters, the first is designed network **“net”,** other is input matrix **“m”**, and the output or target matrix **“out”**.

3.3.1.4 Simulation (sim)

The function ***sim*** simulate the network. ***sim*** takes the network input ‘m’, and the network object ***net,*** and returns the network output ***“output”.***

```
output=sim(net,m);
```

3.3.2 Test

To test the sample signature whether they are valid or invalid, test samples are simulated. The major difference between the test main train function is, in train function we provide the network, input matrix as well the output. Whereas for the testing purpose we just use the simulate function and pass the parameters sample signatures and net, without the output.

```
sim(net,x);
```

“x” is sample test signatures, net is network.

Chapter No.4
Implementation and Results

4.1 Introduction

After feature extraction in preprocessing step, then training of network should be started. As back -propagation algorithm in used in this thesis so network should be created with newf() function and train with train() function.
There are following things should be considered during training

- Signature extracted features as input
- Number of neurons
- Target output
- Number of Sample of Signatures

4.2 Training Results of 2 Person Sample Signatures with different parameters.

Sr.	Train1 (7-500)	Train 2 (7-300)	Train 3 (7-400)	Train4 (9-500)	Train5 (8-500)	Train6 (5-500)	Train7 (3-500)
1.	-0.0064	-0.0064	-0.0285	0.0782	0.0177	-0.2194	-0.6166
2.	-0.0072	-0.0072	0.0130	0.0997	0.0186	0.1101	-0.6079
3.	-0.0017	-0.0017	-0.0105	0.1949	-0.1990	-0.5846	-0.6143
4.	-0.0187	-0.0187	-0.1908	0.1055	0.0279	-0.1094	-0.6090
5.	-0.0021	-0.0021	0.1171	0.1833	0.0298	0.0932	-0.6137
6.	-0.0071	-0.0071	0.0218	0.1028	-0.0463	0.0317	-0.6123
7.	-0.0635	-0.0635	0.01123	0.0456	0.1588	0.1484	-0.6049
8.	-0.0837	-0.0837	-0.4007	0.0012	-0.0006	-0.2434	0.3902
9.	0.0020	0.0020	-0.3033	-0.0341	-0.0615	-0.5191	0.3853
10.	-0.0004	-0.0004	-0.3005	-0.0811	0.0013	-0.3060	0.3920
11.	-0.0305	-0.0305	-0.2984	0.1499	-0.0084	-0.3283	0.4017
12.	-0.0014	-0.0014	-0.2748	0.0129	-0.0028	-0.3383	0.3854
13.	0.0004	0.0004	-0.3969	-0.0256	-0.0214	-0.5542	0.3895
14.	0.0004	0.0004	-0.3969	-0.0256	-0.0214	-0.5542	0.3895

Table 4.1 a: Learning Rate of Sample with different parameter

Sr.	Train8 (20-500)	Train 9 (23-500)	Train 10 (21-500)	Train11 (22-500)	Train12 (11-500)
1.	-0.0000	-0.0333	0.0000	0.0701	0.0000
2.	-0.0000	0.0157	-0.0110	-0.0512	-0.0000
3.	-0.8071	-0.1175	0.00000	-0.0100	-0.0000
4.	-0.0000	-0.0304	-0.0000	-0.1218	0.0000
5.	-0.0000	0.1087	-0.0000	0.0029	0.0000
6.	-0.0262	0.0225	0.0000	-0.0249	0.0000
7.	-0.0000	0.0049	0.0000	-0.0612	-0.0085
8.	-0.3817	0.0050	0.0000	-0.0693	0.0299
9.	0.0000	-0.0962	0.0073	-0.0553	-0.0049
10.	-0.0000	-0.0259	0.4798	-0.0751	0.0000
11.	0.0093	0.0110	0.0000	0.1634	0.0002
12.	-0.000	-0.0114	0.2408	-0.0295	-0.0000
13.	-0.0000	-0.0258	-0.0000	-0.0714	0.0001
14.	-0.0000	-0.0258	-0.0000	-0.0714	0.0001

Table 4.1: Learning Rate of Sample with different parameter

The above training would be done by using 2 person signature samples 7 from each person. The above result would be calculated from following formula.

Training Result = Actual Output – Desired Output

The training result will be refer to as Learning Rate (*lr*). The *lr* directly affect the performance of neural network. With the help of *lr* the mean squared error (mse) should be calculated. If the value of M.S.E is high the performance is decrease. To increase system performance the value of MSE should be minimize by increasing *lr*.

The results in the table simulate the idea that number of epochs does not affect the result. On the other hand, the factor which affect the results is number of neurons. As this system is designed with only one layer, it may affect the performance somehow but with more research it can be converted into a sustainable performance oriented by either increasing number of samples or number of neurons within the bearable limitation of performance.

4.3 Training Results of 25 Person Signature with 7-Sample Signatures from each person

Now number of sample will be increase to 25 people with 7 Signature from each person. The learning rate with minimum value And maximum value will be shown as below.

Sample 1-14	Sample 13-24	Sample 25-36	Sample 37-48	Sample 49-60	Sample 61-72	Sample 73-84	Sample 85-96	Sample 97-108
-0.2377	-0.7057	0.3066	-0.3166	-0.1777	-0.4567	0.635	-0.3595	-0.657
0.2613	-0.7057	0.6356	-1.2448	0.0431	-0.0402	-0.2732	-0.2622	-0.1669
-0.5299	-0.5505	-0.0356	-1.0139	0.0936	-0.3038	0.1697	-0.3	0.08
-0.8098	-0.3535	1.2633	-0.8153	-0.0631	0.0245	-0.3826	-0.3	-0.1658
-0.7179	-0.179	-0.3503	0.0257	0.0244	-0.3778	0.4218	-0.0433	0.201
-1.0247	-0.7861	-0.1676	-0.2228	-0.6971	-0.1287	0.1967	-0.0594	-0.0769
0.1417	-0.7187	-0.5587	-0.3035	0.1651	-0.2099	0.079	-0.0012	-0.0964
-0.8925	-0.085	-0.5297	-0.4898	0.1157	-0.0163	-0.1071	-0.5526	-0.2387
-0.2753	-0.7187	-0.458	-0.1615	-0.4089	-0.0163	-0.2554	-0.4973	-0.084
-1.1014	-0.1012	0.138	-0.1267	-0.0152	-0.1591	-0.4412	0.1529	0.1883
-1.0635	0.0172	0.138	-0.7128	-0.2567	-0.1009	-0.3098	-0.4775	0.309
-0.7121	-0.1801	-0.6715	-0.6202	-0.0504	-0.3076	-0.094	-0.3173	0.4022

Sample 109-120	Sample 121-132	Sample 133-144	Sample 145-156	Sample 157-168	Sample 169-175
0.4116	0.1524	0.2432	0.6119	0.3605	0.5765
0.1081	0.022	-0.028	0.6119	0.3605	0.5765
-0.0124	0.2613	0.131	-0.1564	0.2715	0.6336
0.4932	0.0275	0.0818	-0.0516	0.4764	0.6336
-0.2287	-0.079	-0.0714	0.1431	-0.0627	0.9852
0.4668	0.2336	-0.0714	-0.0516	0.1598	0.4711
0.3762	0.2432	-0.434	-0.0516	0.1598	0.4711
-0.3651	0.2422	-0.028	0.1431	0.0898	
-0.3764	-0.0585	-0.1564	0.1431	0.0102	
-0.3699	0.0628	-0.1869	0.3571	0.0102	
-0.2333	0.0013	-0.1098	0.2715	0.3219	
0.2336	0.0013	-0.1098	0.3947	0.3219	

Minimum	-1.2448
Maximum	1.2633

Table 4.2: Learning Rate of Sample of 25 people, 7 from each people

From above table, there are the results of neural network with 175 signatures as input with multiple outputs for each sample. After training the result will be simulated as performance, regression and Validation result. The minimum & maximum value in the result defines the limit of signature verification i.e. between these limits all the signature become valid.

4.4 Regression

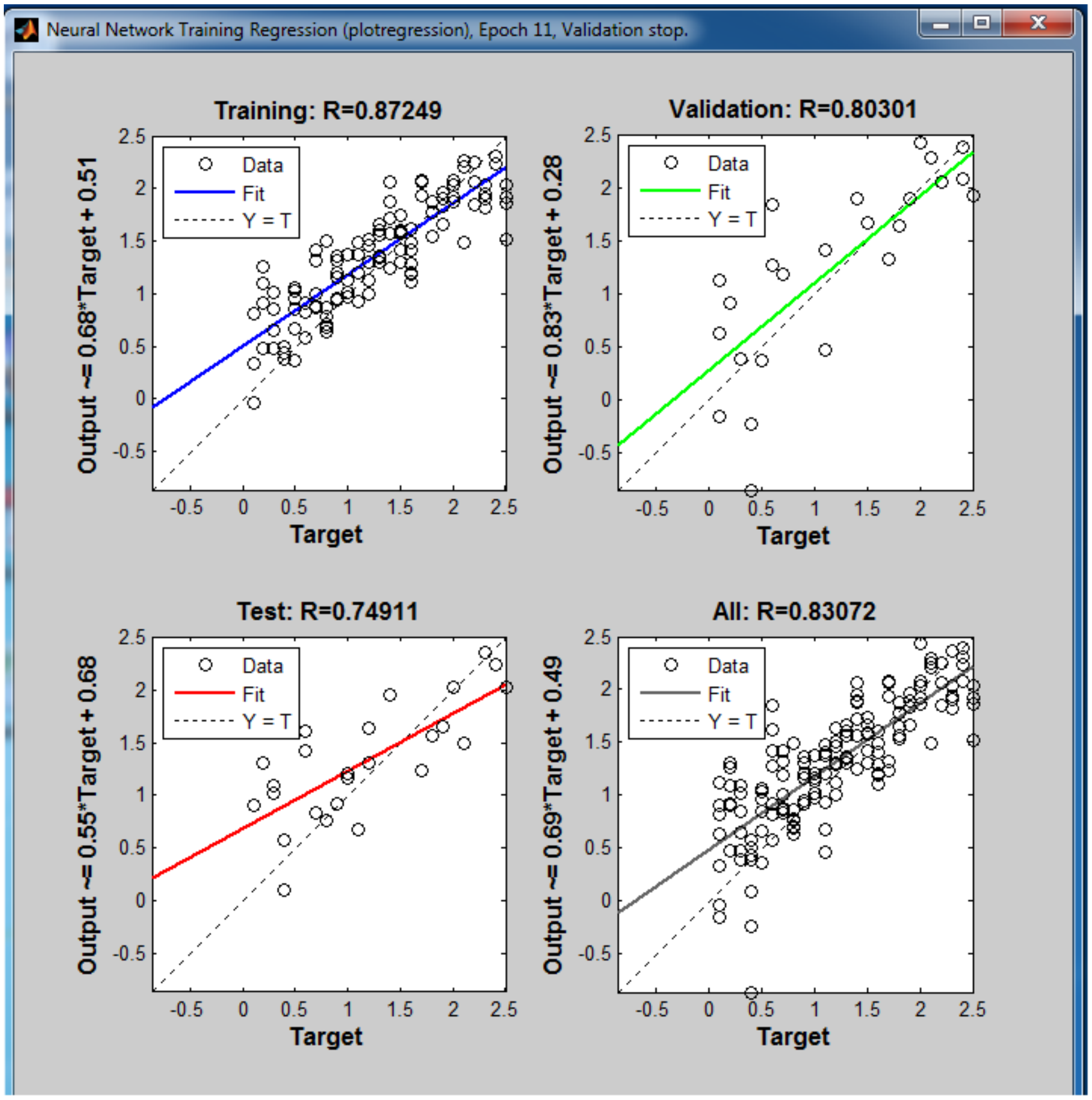


Figure (4.1) Regression Result

4.5 **Validation Test Result**

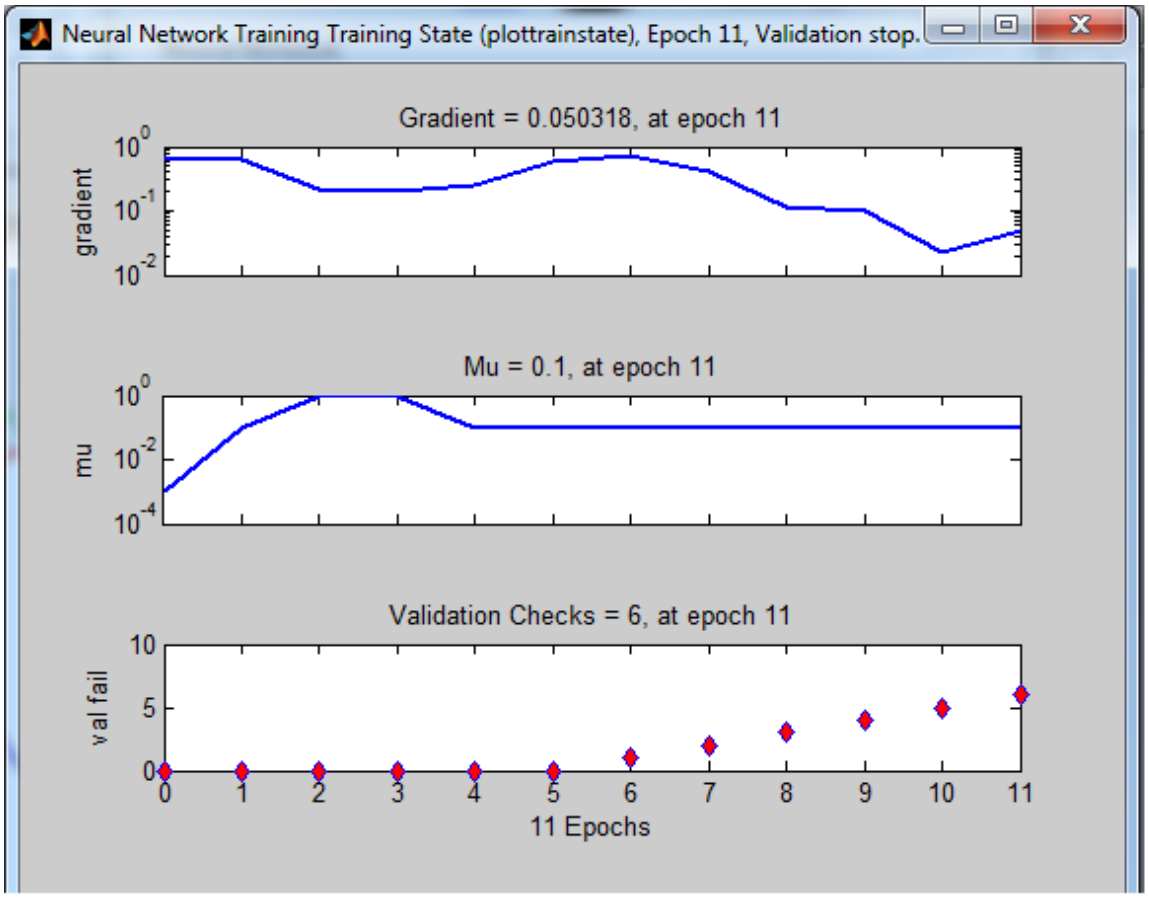


Figure (4.2) Validation Result

4.6 Performance Result

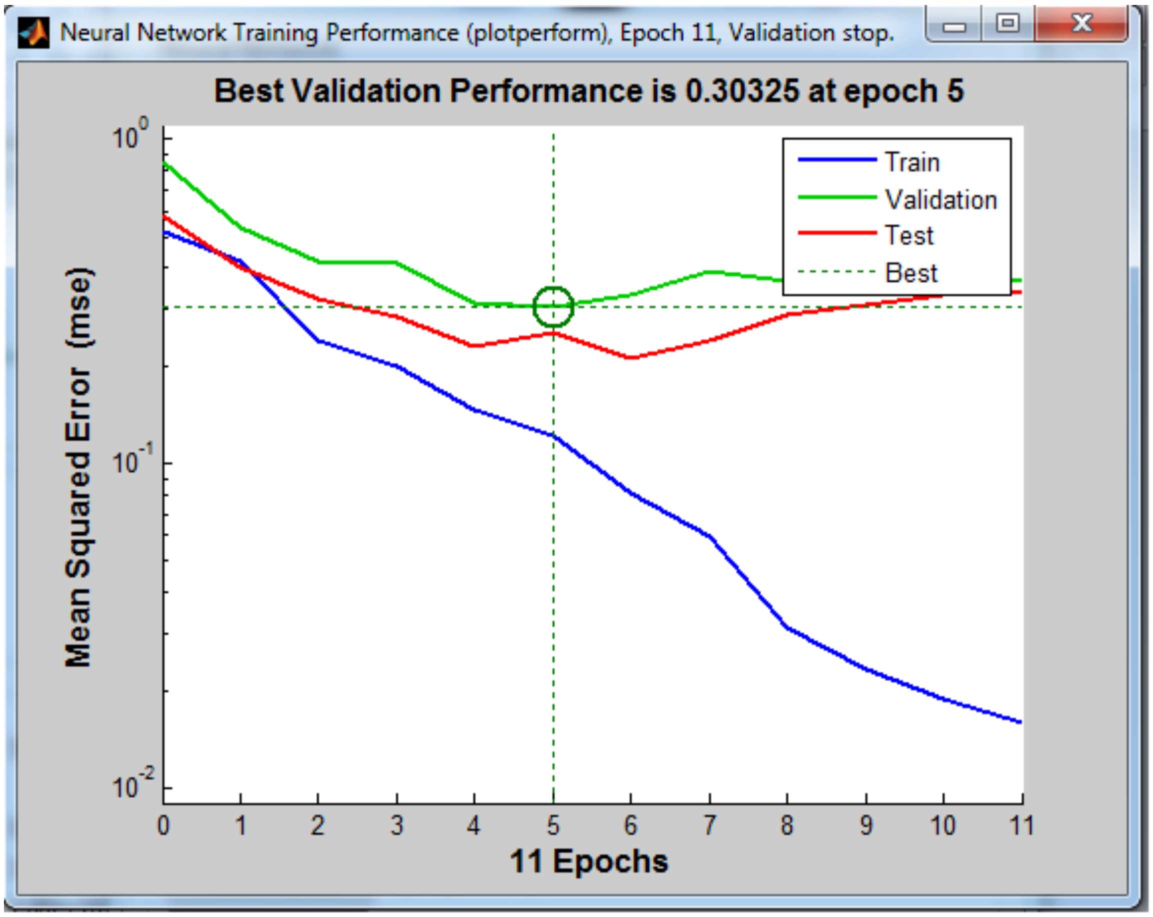


Figure (4.2) Performance Result

Chapter No. 5
Code

In this section we outline the code of our project.

5. 1 Preprocessing Code

5.1.1 Background Elimination

```
%%%%%...........Background Elimination
m=zeros(27,14)
N = 14
img = cell(2,2);
FNAMEFMT = 'a%d.png';

% Load images
for i=1:N

 img{i} = imread(sprintf(FNAMEFMT, i));

%Remove Background
pgrey=rgb2gray(img{i});

%Convert to Binay Image
pbw=im2bw(pgrey,graythresh(pgrey));
```

5.1.2 Noise Elimination

```
%%%%%...............Noise Elimination
%Median Filter
FLT=medfilt2(pbw,[3 3]);
```

5.1.3 Width Normalization

```
%Resize Image
resize=imresize(pbw, [150 300]);
```

5.1.4 Image Thinning

```
%Thinnig
thin=imcomplement(resize);
new=bwmorph(thin,'thin',1);
newthin=imcomplement(new);
%figure(4);
```

5.2 Feature Extraction Code

5.2.1 Aspect Ratio

```
%1.......Ascept Ratio......
[height width d]=size(newthin);
acceptratio = height/width;
m(1,i)=acceptratio;
```

5.2.2 Center Of Gravity

```
%2...........Center of Gravity
[YCoord, XCoord]=find(newthin);
Centroidx=mean(XCoord)
Centroidy=mean(YCoord)
m(2,i)=Centroidx;
m(3,i)=Centroidy
```

5.2.3 SLOP

```
3.Slop
%Get the center of start point cordinates
[YCoord1,XCoord1]=find(newthin);
%figure(5);
%imshow(newthin);
Centrodx11=mean(XCoord1)+mean(XCoord1)/2
Centrody11=mean(YCoord1);
m(4,i)=Centrodx11;
m(5,i)=Centrody11;

%hold on;
%plot(Centrodx11,Centrody11,'o');
%hold off;
%Get the center of end point cordinates
[YCoord1,XCoord1]=find(newthin);
Centrodx111=mean(XCoord1)-mean(XCoord1)/2
Centrody111=mean(YCoord1);
%hold on;
%plot(Centrodx111,Centrody111,'o');
%hold off;
m(6,i)=Centrodx111;
m(7,i)=Centrody111;



%Draw line Between Both Points
xstart=Centrodx11;
ystart=Centrody11;
xmax=Centrodx111;
ymax=Centrody111;
line('xdata',[xstart xmax],'ydata',[ystart ymax]);
```

5.2.4 *Area Normalization*

```
%4Area  Normlization
%Get the total image Area
total_im_area=bwarea(newthin);

%get the Image area
```

```
im_area=bwperim(newthin,8);
%figure(7)
%imshow(im_area);
%signature Area
sig_area=bwarea(im_area);
%figure(8);
%imshow(sig_area);
%normalized aRea
normalizedarea=sig_area/total_im_area;
m(8,i)=normalizedarea;
```

5.2.5 *Image Crop*

```
%6.Cropping
%first part of the image
[j k]=size(newthin);
x1=imcrop(newthin,[0 0 j k]);
%figure(9)
%imshow(x1)
%Second Part of the image
j1=j*2;
k1=k*2;
x2=imcrop(newthin,[j 0 j1 k1]);
%figure,subplot(2,2,1:2),subimage(newthin)
%subplot(2,2,3),subimage(x1)
%for first half of the signature Draw line
[YCoord1,XCoord1]=find(x1)
Centerx1=mean(XCoord1)
Centery1=mean(YCoord1)
xstart1=0;
ystart1=Centery1;
xmax1=max(XCoord1);
ymax1=Centery1;
m(9,i)=Centerx1;
m(10,i)=Centery1;

%for 2nd half of the signature Draw line
[YCoord2,XCoord2]=find(x2)
Centerx2=mean(XCoord1)
Centery2=mean(YCoord1)
xstart2=0;
ystart2=Centery2;
xmax2=max(XCoord2);
ymax2=Centery2;

m(11,i)=Centerx2;
m(12,i)=Centery2;
``````````````````````````````````````````````````````
```

### 5.2.6 *Maximum Horizontal and Vertical Projection*

```
    ....Max Horizontal/Vertical Projection
ver_proj=sum(newthin,1);
hor_proj=sum(newthin,2);

max_ver_proj=mean(ver_proj);
```
```

```
max_hor_proj=mean(hor_proj);

m(13,i)=max_ver_proj;
m(14,i)=max_hor_proj;
```

5.2.7 Edge Detection

```
.......Edge Detection...............
I=uint8(newthin);
%BW1 = edge(I,'prewitt');
bw = edge(I,'canny');
%imshow(bw);
%figure, imshow(bw);
if ~(isa(bw,'double') || isa(bw,'single')); bw = im2single(bw); end
h=fspecial('sobel');
gh = imfilter(bw,h);
gv = imfilter(bw,h');
g=(gh.^2+gv.^2).^(1/2);
%figure(2);
%imshow(g);

n=sum(g);
y=std(n);
%hist(y);
m(15,i)=y;
```

5.2.8 Texture Features

```
.........Texture Features........
%.........first segment of Image.....
txt1=imcrop(newthin,[0 0 100 75]);
      %figure(2);
      glcm1=graycomatrix(txt1);
      glcm1
     % hist(glcm1);
      max_row_glcm1=max(glcm1);
      max_row_glcm11=max(max_row_glcm1);

      min_row_glcm1=min(glcm1);
      min_row_glcm11=max(min_row_glcm1);
m(16,i)=(max_row_glcm11);
m(17,i)=(min_row_glcm11)

 %.....................2
 txt2=imcrop(newthin,[0 76 100 75]);
      %figure(2);
      glcm2=graycomatrix(txt2);
      glcm2
     % hist(glcm2);
      max_row_glcm2=max(glcm2);
      max_row_glcm22=max(max_row_glcm2);

      min_row_glcm2=min(glcm2);
      min_row_glcm22=max(min_row_glcm2);

 m(18,i)=(max_row_glcm22);
m(19,i)=(min_row_glcm22)
```

```
 %.........3
      txt3=imcrop(newthin,[101 0 100 75]);
      %figure(2);
      glcm3=graycomatrix(txt3);
      glcm3
      %hist(glcm3);
      max_row_glcm3=max(glcm3);
      max_row_glcm31=max(max_row_glcm3);

      min_row_glcm3=min(glcm3);
      min_row_glcm31=max(min_row_glcm3);

m(20,i)=(max_row_glcm31);
m(21,i)=(min_row_glcm31);



    %.........4
      txt4=imcrop(newthin,[101 76 100 75]);
      %figure(2);
      glcm4=graycomatrix(txt4);
      glcm4
      %hist(glcm4);
      max_row_glcm4=max(glcm4);
      max_row_glcm41=max(max_row_glcm4);

      min_row_glcm4=min(glcm4);
      min_row_glcm41=max(min_row_glcm4);

m(22,i)=(max_row_glcm41);
m(23,i)=(min_row_glcm41);

  %.........5
      txt5=imcrop(newthin,[201 0 100 75]);
      %figure(2);
      glcm5=graycomatrix(txt5);
      glcm5
      %hist(glcm5);
      max_row_glcm5=max(glcm5);
      max_row_glcm51=max(max_row_glcm5);

      min_row_glcm5=min(glcm5);
      min_row_glcm51=max(min_row_glcm5);

 m(24,i)=(max_row_glcm51);
m(25,i)=(min_row_glcm51);



    %.........6
      txt6=imcrop(newthin,[201 76 100 75]);
      %figure(2);
      glcm6=graycomatrix(txt6);
      glcm6
      %hist(glcm6);
      max_row_glcm6=max(glcm6);
      max_row_glcm61=max(max_row_glcm6);

      min_row_glcm6=min(glcm6);
      min_row_glcm61=max(min_row_glcm6);

m(26,i)=(max_row_glcm61);
m(27,i)=(min_row_glcm61);
```

5.3 Test Code

```
yy123=sim(net,x);
diff1=meanvalue-yy123;
    if(diff1>=-1.5687 && diff1<= 0.9921)
      % answer=1;
      msgbox(['valid signature']);
   %     break;
    else
     %   answer=0;
     msgbox(['Not valid signature']);
    end
%end
```

5.4Neural Network Code

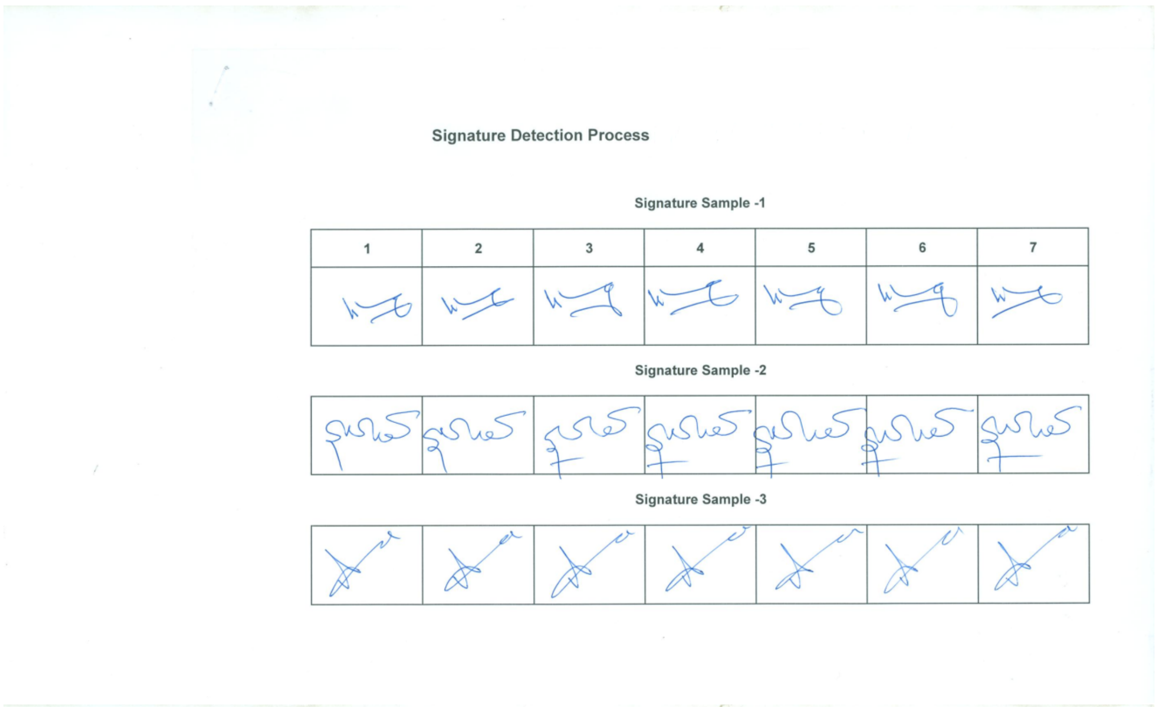
```
%diff=zeros(14)
%net=newff(minmax(m),out,[11
1],{'tansig','purelin'},'trainlm','learngdm','mse');
net=newff(m,out,[11
1],{'tansig','purelin'},'trainlm','learngdm','mse');
net.trainParam.epochs=500; %(number of epochs)
%net.trainParam.goal = 0.1;
%net.trainParam.show = 20;
%net.trainParam.lr=0.9;
%net.trainParam.mc=0.65;
[net,tr123,netoutput]=train(net,m,out);
output=sim(net,m);
%yy1=floor(output);
%yy1=round(output);
yy1=sim(net,m);

%net.trainParam.epochs=500%(number of epochs)
```

```
for i=1:14
   diff(i)=out(i)- yy1(i);
end
prform=mse(diff);
```

5.5 Sample Signatures

In this section we provide some of the sample signatures ,that we use for database.



Signature Detection Process

Signature Sample -1

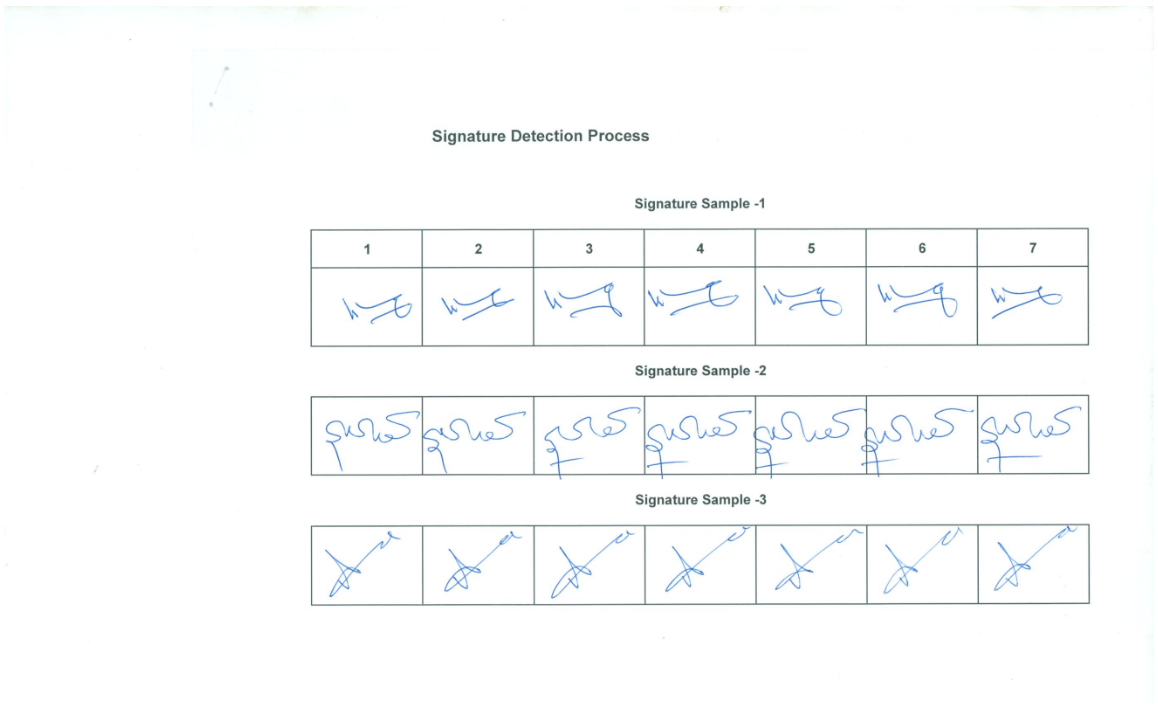
1	2	3	4	5	6	7
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Signature Sample -2

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Signature Sample -3

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]



Signature Detection Process

Signature Sample -1

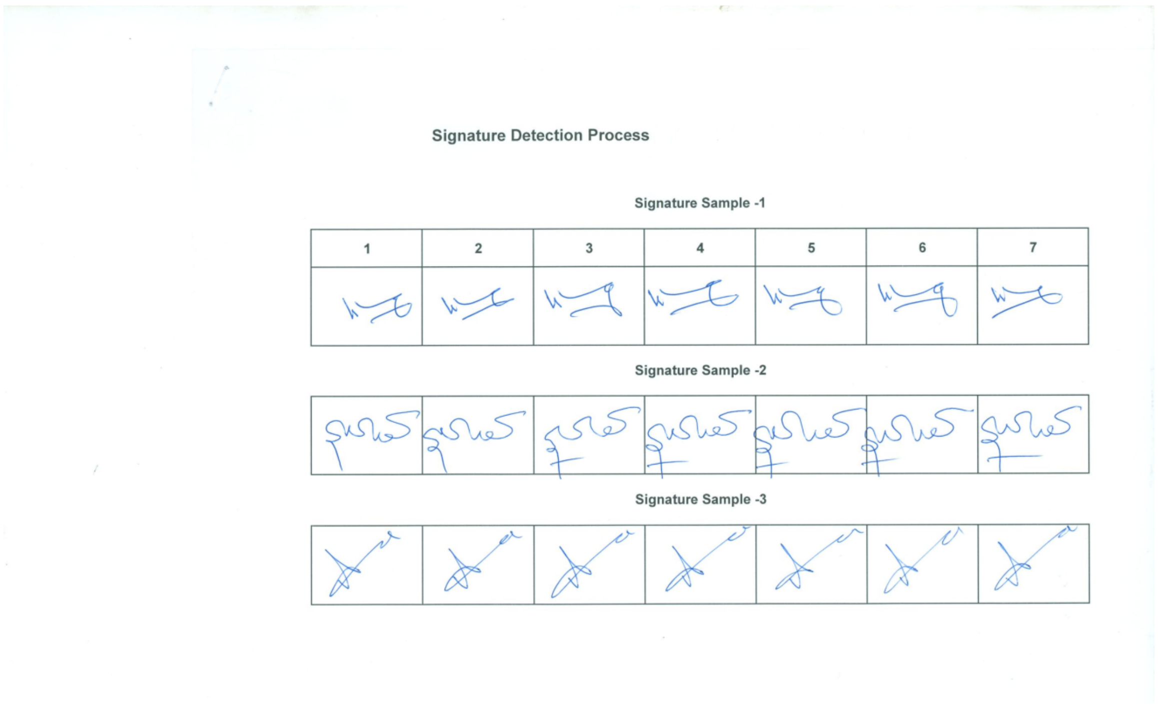
1	2	3	4	5	6	7
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Signature Sample -2

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Signature Sample -3

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]



Signature Detection Process

Signature Sample -1

1	2	3	4	5	6	7
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Signature Sample -2

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Signature Sample -3

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

Chapter No. 6
CONCLUSION AND FUTURE WORK

6.1 Conclusion

The research presented here describes a way for offline signature verification system through neural network. The strength of this research is the use of single neural layer with 11 neurons. There are about 27 values of features to be extracted. The efficiency and accuracy of the system depends upon increased *learning rate.* By increasing learning rate mean square error would be minimized. The epoches in the network training does not affect its efficiency; those factors which affect the result of neural network are number of neurons, number of training sample, variation in the relevant samples and also number of features to be extracted in preprocessing step.

As the signature verification involves image processing so use of feature extraction techniques & Neural Network is very valuable. As the implementation is to be done in matlab, so built-in-command for most respective implementations is almost available.

6.2 Future Work

In future there exist a lot of work need to be done, as finding the effect of neural network by increasing number of features to be extracted, keeping other condition same. There arises a major change in the result if any rotated image of signature should be included in the input sample. So, a brief discussion of image rotation will be the part of future work.

References:

[1]. A. I. Abdullah, “Handwritten Signature Verification Using Image Invarients and Dynamic Features”, *Proceeding of the International Conference on Computer Graphics, Imaging and Visualisation*, 2006.

[2]. G. F. Russel. A. Heliper. B. A Smith. J. Hu. D.Markman. J. E. Graham. T. G Zimmerman. And C. Drews, “Retail Application of Signature Verification,” *Proceedings of SPIE 2004*, vol. 5404, pp.206-214, August 2004.

[3]. S. Reddy.B. Maghi. And P.Babu,”Novel Features for Offline signature verification.,” *Journal of Computer, Communication and Control*., vol. 1, pp. 17-24,2006

[4]. http://www.advancedsourcecode.com/signatureverification.asp

[5]. Maya V. Karki, K. Indira, Dr. S. SethuSelvi, “Off-Line Signature Recognition and Verification using Neural Network” , “*International Conference on Computational Intelligence and Multimedia Applications*”, 2007.

[6]. H. Baltzakis, N. Papamaroks,”A new signature verification technique based two-stage neural network classifier”, “*Engineering Applications of Artificial Intelligence 14 (2001) 95-103*”; accepted 1 September 2000.

[7]. Stephane Armand, Michael Blumenstein and VallipuramMuthukkumarasamy, “Off-line Signature Verification based on the Modified Direction Feature”, ”*The 18th International Conference on Pattern Recognition (ICPR’06) 0-7695-2521-0/06* ”, © 2006 IEEE.

[8]. O.C Abikoye, M.A Mabayoje, R. Ajibade ,"Offline Signature Recognition & Verification using Neural Network", "*International Journal of Computer Applications (0975-8887) Volume 35-No.2*", December 2011

E. Messerli and G. Primault, "Cryptographic Algorithms with Co-design Approach Optimisation" Reconigurable and embedded digital systems, Institute REDS-EIVD, 2005.

[9]. http://www.mathworks.com/access/helpdesk/help/toolbox/images/f11-12251.html#f11-14283

[10]. Scott Umbaugh, "Computer Vision and Image Processing: A Practical Approach using CVIP Tools", Prentice-Hall. Inc., USA, 1998.

[11]. Rafael C. Gonzalez and Richard E. Woods, *Digital image processing*, Addison-Wesley, 1993.

[12].Chin-Teng Lin, Student Member, *IEEE*, and C.S. George Lee, Senior Member, *IEEE* ,"*Neural-Network-Based Fuzzy Logic Control and Decision System*" , IEEE TRANSCATIONS ON COMPUTERS, VOL. 40, NO. 12, December 1991.

[13]. T. Senjowski and C. Rosenberg, "NETtalk: A parallel network that learns to read aloud", JHU/EECS-86/01, Tech. Rep., EECS Dep,. Johns Hopkins Univ,. 1986.

[14]. K. Fukushima, S. Miyaka, and T. Ito, "Neocognitron: A neural nerwork model for Mechanism of visual pattern recognition," *IEEE Trans, Syst,. Man, Cybern,* vol. SMC-13, no. 5 , pp. 826-834, 1983.

[15]. S. Grossberg, "Cortical dynamics of three-dimensional from color, and brightness perceptions, " Perception and Psychophys., vol. 41, no. 2, pp. 87-116,1987.

[16]. A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can sol"e difficult learning control problems," *IEEE Trans, Syst,. Man, Cybern,* vol. SMC-13, no. 5 , pp. 834-847, 1983.

[17]. C. W. Anderson, "Learning to control an inverted pendulum using neural network," *IEEE Cobtr. Syst. Mag.,* pp.31-36, Apr. 1989.

[18]. F. C. Chen,"Back-propagation neural network for nonlinear self-tuning adaptive control," *Proc. IEEE Intelligent Machine*, pp. 274-279. 1989.

[19]. Howard Demuth, Mark Beale, "Neural Network Toolbox, For Use with MATLAB", User Guide, Version 4

[20]. B. Herbst. J. Coetzer. and J. Preez, "Online Signature Verification Using the Discrete Randon Transform and a Hidden Markov Model," *EURASIP,. Journal on Applied Signal Processing,* vol. 4, pp. 559-571, 2004

[21]. F. Bortolozi E. R. Justino., A. E. Yocoubi. And R. Sabourin, "An Off-line Signature Verification System Using HMM and Graph metric features, " *DAS 2000,4th IAPR International on Document Analysis Systems, Rio de Jenerio,* 2000

[22]. S.I Abuhaiba, "Offline Signature Verification Using Graph Matching" , *Turk J Elec Engine,* vol. 15, no. 1, 2007.

[23]. T. Senturk. E. Ozgunduz. And E. Karshgil, "Handwritten Signature Verification Using Image Invarient and Dynamic Features" , Proceedings of the 13th European Signal Processing Conference EUSIPCO 2005, Anatalya Turkey, 4th -8th September, 2005.

[24]. H. Hammandlu and V.M. Kishna, "Off-line Signature Verification and Forgery detection using Fuzzy modeling, " *Pattern Recognition* , Vol. 38, pp. 341-356, 2005