

An Efficient Neural Network for Recognizing Gestural Hindi Digits

¹Nidal Fawzi Shilbayeh, ¹Mohammad Mahmoud Alwakeel and ²Maisa Mohy Naser

¹Department of Computer Science,

Faculty of Computer and Information Technology, University of Tabuk, Tabuk, Saudia Arabia

²Department of Computer Science, Faculty of Information Technology, Middle East University, Amman, Jordan

Received 2012-06-20, Revised 2012-12-15; Accepted 2013-07-25

ABSTRACT

Handwritten Hindi digit recognition plays an important role in eastern Arab countries especially in the courtesy amounts of Arab bank checks, recognizing numbers in car plates, or in postal code for mail sorting. In our study, we proposed an efficient Hindi Digit Recognition System drawn by the mouse and developed using Multilayer Perceptron Neural Network (MLP) with backpropagation. The system has been designed, implemented and tested successfully. Analysis has been carried out to determine the number of hidden nodes that achieves high performance. The proposed system has been trained on samples of 800 images and tested on samples of 300 images written by different users selected from different ages. An experimental result shows high accuracy of about 91% on the testing samples and very close to 100% on the training samples. Experiments showed that our result is high in comparison with other Hindi digit recognition systems especially if we consider the way of writing (mouse and children) in our trained and tested results.

Keywords: Digit Recognition, Hindi Digits, Mouse Gesture, Neural Networks, MLP, Backpropagation, Feature Extraction

1. INTRODUCTION

Gesture recognition has emerged as one of the most important research areas in the field of motion-based image processing and recognition. Latest keyboard have been replaced by handwriting technology recognition in a palm and pocket PDA's (Shilbayeh *et al.*, 2009). In this study, we are focusing in developing a new system to recognize Hindi digits using a mouse device as an input device.

Eastern Arab countries are mostly use Hindi Digits in their writing. The term "Hindi digits" is used here to refer to the numerals written in these countries. **Table 1** shows the Arabic digits and the corresponding Hindi digits.

Hindi digit recognition is a complicated classification problem because each person draws these digits using the mouse in a different styles, sizes and possible orientation. In addition to that, most children and the people in old ages preferred to use the mouse instead of the keyboard.

In this study, we designed and implemented a user = friendly and comfortable system for recognizing handwritten Hindi digits using the mouse. We took in our considerations the simplicity, the speed of recognition and the accuracy rate of recognition.

Arabic digits recognition attracted many researchers (LeCun *et al.*, 1989; Mashor and Sulaiman, 2001; Al-Omari *et al.*, 2009; Ouchtati *et al.*, 2008; Shilbayeh and Iskandarani, 2005) but Hindi digits recognition has been started recently. Researchers in this field have proposed different approaches, such as statistical, structural, hidden Markov model and neural network approaches (Mahmoud and Awaida, 2009; Shilbayeh *et al.*, 2009).

The proposed system uses an off-line input for digit recognition using the mouse. The handwritten numbers entered into the system in different styles and sizes. The system recognizes isolated digits by acquiring an image that consist of Hindi digit, then process it through the recognition phases explained.

Corresponding Author: Nidal Fawzi Shilbayeh, Department of Computer Science, Faculty of Computer and Information Technology, University of Tabuk, Tabuk, Saudia Arabia

Table 1. Hindi and Arabic digits

9	8	7	6	5	4	3	2	1	0	Arabic digits
१	८	७	६	५	४	३	२	१	०	Hindi digits

2. MATERIALS AND METHODS

This section presents a general framework for the proposed Hindi digits recognition system. The developed system is divided into the following main stages:

- Input the hindi digit: The user draws a Hindi digit inside the special window using the mouse and then it is saved on a file as .jpg image
- Preprocessing: The goal of preprocessing is to simplify the digit recognition problem without throwing away any important information to be more concise representation for feature extraction stage. This operation involves converting the gray image into a binary image, resizing the image, reversing the binary image and skeletonizing
- Feature extraction: Features are a set of values of a given digit that are used to distinguish the digit from each other. The feature extraction phase calculates these values in order to produce a set of measurements, called the feature vector, for each object
- Digit recognition: The recognition step is based on the use of neural networks, or in more, it's based on MLPs. This step realizes a set of discriminated functions that associate a score to each possible class. These scores may be regarded as being representative of the probability of each class, to be one of the digits presented to the system
- Display digit: Displaying the output of a Hindi digit

2.1. Input the Hindi Digit

The interface is designed using C#. net. The user can draw the Hindi digit inside (160×200 pixels) window with any style, size and possible orientation, without limitation of the starting point as in Shilbayeh *et al.* (2009).

To start drawing a digit, the user presses the left mouse button as shown in **Fig. 1**, then he saves the image in a database folder (image sample) as .jpg image, the user draws as many images as he wants.

Figure 2 shows drawing of (0-9) Hindi digits using C#.net. The image then passed to the Matlab tools to start the next stage for preprocessing, because it has very powerful tools for implementing and manipulating images.

2.2. Preprocessing

The preprocessing stage is a very important step in image processing. After the image is stored in the input file, a number of preprocessing steps are performed for these images. These steps generally include: Binarizing, resizing the image from 160×200 pixels to 30×30 pixels, inverting the binary image and finally doing skeletonization. The preprocessing steps is shown in **Fig. 3**.

2.3. Binarization Step

This process aims to convert the gray values into the binary values. The binary images contain only 0's and 1's. Pixels with the value 0 are displayed as black, pixels with the value 1 are displayed as white and a binary image is stored as a logical array. This step is done by comparing the gray values of an image with a given threshold. This threshold value is measured by finding the dominant gray value in the input image and then choosing the threshold value to be the center point between the dominant value and the maximum gray value.

After determining the threshold value, each pixel in the image is compared with the threshold value. If the value of the pixel is less than the threshold, reset the pixel to zero. Otherwise, reset the pixel to one as in Equation 1:

$$P(x,y) = \begin{cases} 0 & P(x,y) < \text{threshold value} \\ 1 & P(x,y) \geq \text{threshold value} \end{cases} \quad (1)$$

where, $P(x, y)$ is the pixels of the image and the threshold value 255 is the value between the dominant and the maximum value.

After applying the binarization algorithm on the digital image, we obtain a binary image consisting of two values "0", "1", as illustrated in **Fig. 4**.

2.4. Resizing the Image Step

The images are normalized to a fixed size using the function image resize (imresize (baw1, [30 30])), where bw is the image which convert to binary. So the final size is 30×30 pixels. As shown in **Fig. 5** the background of the image takes the value 1, while the handwritten numbers take the value 0.

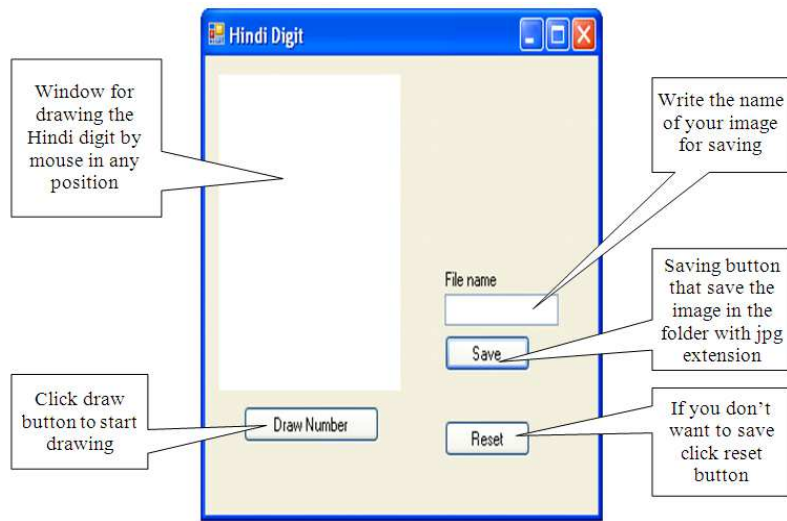


Fig. 1. C# Graphical User Interface for drawing Hindi digit

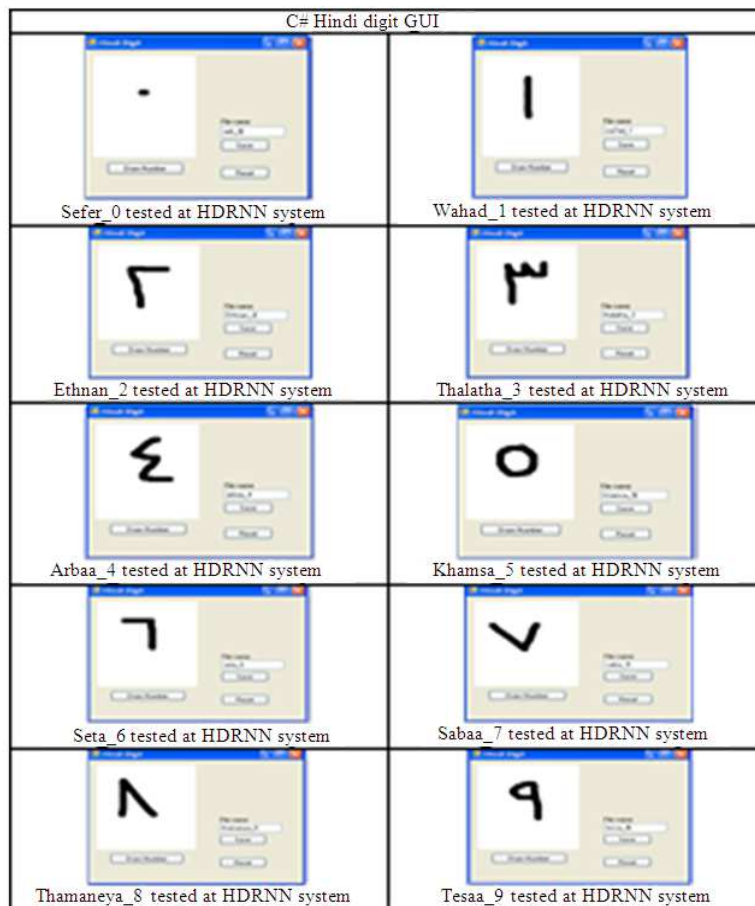


Fig. 2. (0-9) Digits drawn by mouse in C# System interface

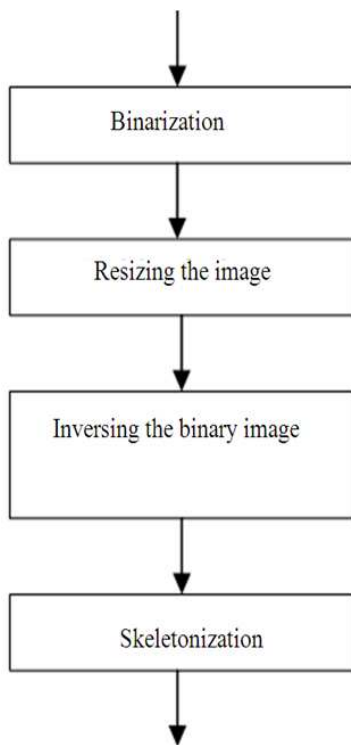


Fig. 3. Preprocessing Steps

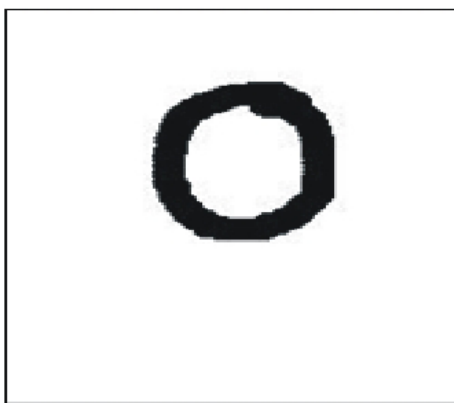


Fig. 4. Binary Image

2.5. Inverse the Binary Images Step

After we binarized the image, the background pixels took the 1's values and the object pixels took 0's values. Hence, we could not manipulate the image. We might need to invert the binary images when we use them, so that the 0's values are inverted as white and the 1's values are inverted as black.

Therefore, an inverse function should be applied to inverse background pixels into 0's and the object pixels into 1's in order to manipulate the image. **Figure 6** shows the inverse of the binary images.

After inverting the binary image, the background takes the 0 value, while the handwritten number takes 1 value, as shown in **Fig. 7**.

2.6. Skeletonization Step

While the preprocessing phase converts the form of the input image to another form, we need a technique to reduce all objects in an image to lines without changing the essential structure of the image and the information such as grayscale values and thickness of the digits actually assist in providing more information, so we need to reduce these values to simplify the feature extraction phase. Thinning algorithm is used to convert the original image into a skeleton image (Al-Rashaideh, 2006).

Skeletonization removes pixels of objects but does not allow objects to break apart. The remaining pixels make up the image skeleton that is shown in **Fig. 8**. This approach of acquiring the skeleton of the digit is widely used in the classifiers which mainly rely upon a well-defined input image for their accuracy.

We used the function `bwmorph (BW, 'thin', Inf)` for skeletonizing the digits.

2.7. Feature Extraction Stage

To build an application for a handwriting, all the phases of the automated handwritten recognition system should be designed carefully and precisely because of the variability and complexity of the problem. One of the most important stages is the preprocessing phase, which constructs the representation of the digits in order to simplify the feature extraction phase.

When the input data is too large to be processed and it is suspected to be redundant (much data, but not much information), the input data will be transformed into a reduced representation set of features (also named features vector). Transforming the input data into a set of features is called feature extraction. If the features extracted are carefully chosen, it is expected that the features set will extract the relevant information from the input data, in order to perform the desired task using this reduced representation instead of the full size input.

The choice of features extraction method limits or dictates the nature and output of the preprocessing step, as well as the decision to use gray-scale versus binary image, filled representation or contour, thinned skeleton versus full-stroke image depend on the nature of the features to be extracted.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
8	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
9	1	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1
10	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1
11	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1
12	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1
13	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1
14	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1
15	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1
16	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1
17	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
18	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
19	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
20	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
21	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
22	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
23	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
24	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
25	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
26	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
27	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
28	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
29	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
30	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Fig. 5. Number Khamsa_5 after binarization and resizing

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	
8	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	
9	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	
10	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	
11	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	
12	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	
13	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	
14	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	
15	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	
16	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	
17	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
29	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Fig. 6. Number Khamsa_5 after inversing the binary image

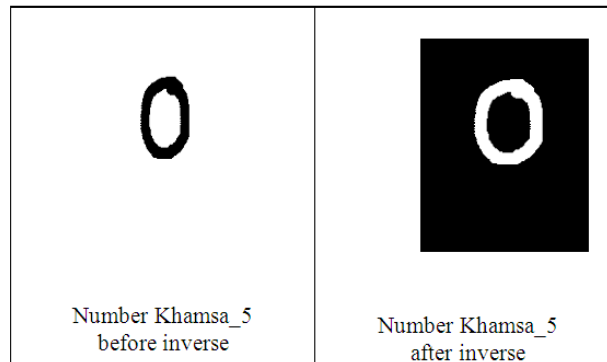


Fig. 7. Binary images before and after inverting



Fig. 8. Number khamsa_5 after and before skeletonization

Extract measurements from the input to distinguish between classes, average and standard deviation suggests that the problem of extracting features from input data is done by selecting information which is more relevant for classification purposes and able to discriminate between classes, these features are a set of values of a given digit that are used to distinguish it from each other. The feature extraction phase calculates these values, producing a set of measurements called a feature vector for each object.

In the proposed system, we ignored some bad features such as the height, the width and the bounding box; on the other hand, we added additional best features. This process of extracting the features relies on a direct and simple mathematical method. This means that it is capable of recognizing the single digit. This method could be summarized in the following points:

- Store each image in a matrix with size 30×30
- The system divides the image into blocks where each block has a size of 10×10 by dividing the image horizontally into 3 blocks and vertically into 3 blocks as shown in **Fig. 9**
- For each block, store the following two features of each digit in a one dimension array called feature vector

- The average which is the easiest to use and involves reduction of the size of each digits to a normalized dimension A grid area is superimposed on the image of the digit and the average value of the pixel of each area is computed
- Calculate the standard deviation, see the standard deviation as in Equation 2:

$$S = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2} \quad (2)$$

Now the vector contains 18 elements as shown in **Table 2**.

2.8. Digit Recognition

The recognition stage is the stage that classifies and recognizes the digits. By comparing the feature of a new digit with all features which are stored in the databases to determine the recognized digit. There is a famous way to recognize the digit which is the neural network digit recognition which uses the feed-forward neural network and back-propagation.

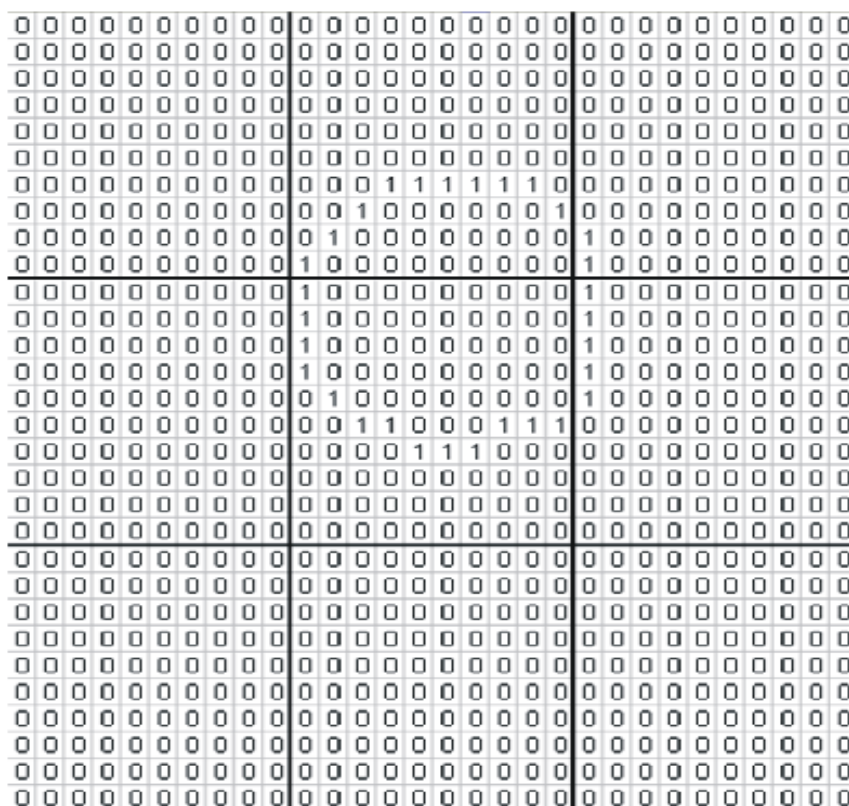


Fig. 9. Blocks resulting from dividing an image

Table 2. Number khamsa 5 (18) feature vector

avg = 0	avg = 0.1000	avg = 0.0200
s = 0	s = 0.3000	s = 0.1400
avg = 0	avg = 0.1300	avg = 0.0500
s = 0	s = 0.3363	s = 0.2179
Avg = 0	avg = 0	avg = 0
s = 0	s = 0	s = 0

Neural Networks (NNs) are computer software or possible hardware that consists of a large number of interconnected simple processing elements called nodes or units whose functionality is based on the biological elements. As in nature, these nodes are connected together with weighted connections called links. Like biological neuron, learning in neural network involves the adjustments of the weighted connection that exist between the neurons during the training process. Neural networks have the ability to learn by example through training or by exposure of a set of input and output data. For example, NNs can be trained to recognize the image of digit by showing them many examples of these digits (Fausett, 1994; Stergiou and Siganos, 1996).

2.9. The Proposed System Network Architecture

The proposed feedforward neural network architecture is shown in **Fig. 10** which consists of, input layer for information processing nodes, one hidden layer with hidden nodes and finally output layer which consists of output nodes that is usually equal to the number of recognition digits.

where, IW is the input weight, LW is the output weight, b1 is the input bias and b2 is the output bias.

The number of nodes in each layer.

The total number of input nodes is equal to the number of input feature for a problem domain. As we mentioned before, we stored each digit in a matrix with size 30×30, the system divided the image into blocks where each block has a size of 10×10 by dividing the image horizontally into three blocks and vertically into three blocks. For each block, we stored the average and the standard deviation of each block in the feature vector, this vector contains 18 elements and these are the number of input nodes.

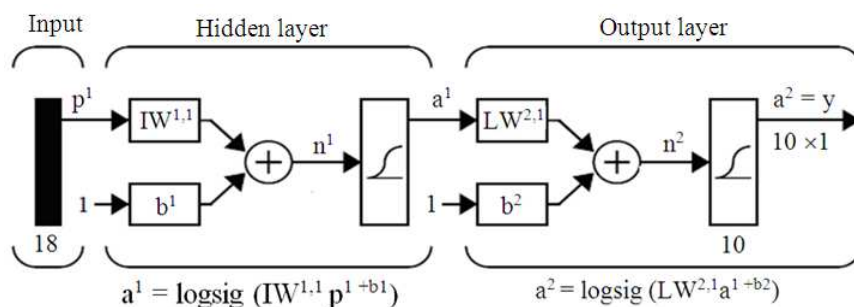


Fig. 10. Three layers feedforward neural network

In order to make the neural network adequate, an optimal number of the hidden nodes must be chosen. To do that, we used one of the ways to determine the suitable number of the hidden node trying many networks with different numbers of hidden units and by finding the minimum Mean Square Error (MSE).

In most situations, there is no way to determine the best number of hidden nodes without training several networks and estimating the generalization error of each. If you have too few hidden nodes, you will get a high training error and high generalization error due to under-fitting and high statistical bias. Larger numbers of nodes in the hidden layer give the network more flexibility because the network has more parameters it can optimize. (Increase the layer size gradually. If you make the hidden layer too large, you might cause the problem to be under-characterized and the network must optimize more parameters than there are data vectors to constrain these parameters). Geman (1992) discusses how the number of hidden nodes affects the bias/variance trade-off. The network architecture with the least number of hidden nodes and with the minimal error is selected. In this case, it was 100 hidden nodes in the hidden layer.

The output vector has 10 elements each represent a digit. To operate correctly, the network should respond with a 1 in the position of the digit being presented to the network. All other values in the output vector should be 0.

According to the previous analysis, we select 18×100×10 Neural Network model.

2.10. The Activation (Transfer) Function

In our recognition system, logistic-sigmoid for both hidden and output nodes are used, the sigmoid transfer function shown below takes the input, which can have any value between plus and minus infinity and squashes the output into the range 0 to 1 defined as in Equation 3:

$$\text{Logsig}(n) = 1 / (1 + \exp(-n)) \tag{3}$$

This function is illustrated in Fig. 11.

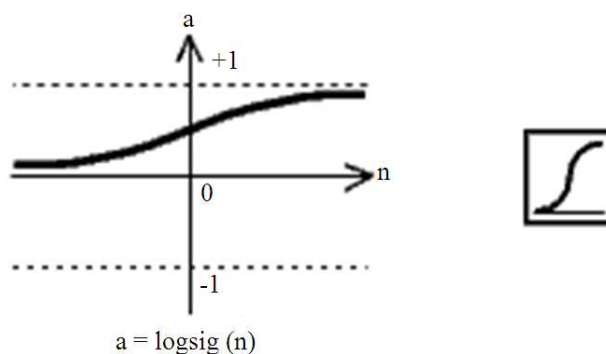


Fig. 11. Logistic-sigmoid, range (0, 1)

2.11. The Learning Rule

This procedure can also be referred to as a training algorithm. The recognition performance of the MLP network will highly depend on the structure of the network and training algorithm.

In our system, we selected backpropagation as a learning method due to its simplicity and because it has been previously used in a number of OCR problems. The term backpropagation refers to the manner in which the gradient is computed for nonlinear multilayer networks (Gołda, 2005).

This method is used for training a set of digits which are presented to the network. The network then computed its output and if there is an error due to the difference between the actual and desired pattern, the network backwards to the hidden layer. So this method is suitable for training the digits.

2.12. The Training Algorithm

The activation function defined in the previous section can be used in the standard backpropagation algorithm given here.

The form of the data is an important factor in choosing the appropriate function. The algorithm is as the following steps:

- Initialize weights, you can set to small random values
- While stopping, condition is false do steps 2-9
- For each training pair, do steps 3-8 Feedforward:
- Each input unit (x_i , $i=1$ to 18) receives an input signal x_i and to all units in the layer above (the hidden units)
- Each hidden unit (z_j , $j=1$ to 100) sums its weighted input signals

$$z - in_j = v_{0j} + \sum_{i=1}^n x_i v_{ij}$$

Apply its activation function to compute its output signal:

$$z_j = f(z - in_j)$$

And sends this signal to all units in the layer above (output units).

Each output unit (y_k , $k = 1$ to 10) sums its weight input signals:

$$y - in_k = w_{0k} + \sum_{j=1}^n z_j w_{jk}$$

Backpropagation of error:

Each output unit (y_k , $k = 1$ to 10) receives a target pattern corresponding to the input training pattern. Computes its error information term:

$$\partial_k (t_k - y_k) f' (y - in_k)$$

Calculate its weight correction term (used to update w_{jk} later):

$$\Delta w_{jk} = \alpha \partial_k z_j$$

Calculates its bias correction term (used to update w_{0k} later):

$$\Delta w_{0k} = \alpha \partial_k$$

and sends ∂_k to unit in the layer below.

Each hidden unit (z_j , $j=1$ to 100) sums its delta inputs (from units in the layer above):

$${}^m \partial - in_j = \sum_{k=1}^n \partial_k w_{jk}$$

Multiplies by the derivative of its activation function to calculate its error information term:

$$\partial_j = \partial - in_j f' (z - in_j)$$

Calculates its weight correction term (used to update v_{ij} later) $\Delta v_{ij} = \alpha \partial_j x_i$ and calculates its bias correction term (used to update v_{0j} later) $\Delta v_{0j} = \alpha \partial_j$ Update weights and biases.

Each output unit (y_k , $k = 1$ to 10) updates its bias and weights ($j = 0$, to 100) $w_{jk}(\text{new}) = w_{jk}(\text{old}) + \Delta w_{jk}$ Each hidden unit (Z_j , $j=1$ to 100) updates its bias and weights ($i = 0$ to 18):

$$v_{ij}(\text{new}) = v_{ij}(\text{old}) + \Delta v_{ij}$$

2.13. Test Stopping Condition

There are different algorithms to train the network. Different neural networks can be chosen to recognize the handwritten digits.

Neural network with different number of layers and different neurons per layer have been trained and simulated. Also different transfer functions have been implemented and their performance has been compared.

We selected the NN which has the minimum error consists of three layers, logsig activation function for both the hidden layers, in addition to the output layer. Mean Squared Error (MSE) used to judge the network performance to perform numeral recognition, as shown in **Fig. 12 and 13**.

2.14. The Experimental Results

Our Hindi digit recognition system has been built using two different platforms. First, the Microsoft visual C#.net 2005 software is used as a side of gesturing the Hindi digit. Then the Matlab software is used as a side for handling, manipulating and recognizing neural network techniques.

A neural network consists of three layers has been experimented. The network has been trained and tested using feed forward backpropagation algorithm. We used 1100 different images of digits. 800 of these images are used to train the neural network and the other 300 are used to test the performance of the network.

The training error is an important indicator for the model performance. The backpropagation training algorithm tries to minimize this error. We selected the neural network that consists of three layers that has a minimum number of errors, logsig activation function for both the hidden layers and the output layer. Mean Squared Error (MSE) used to judge the network performance to perform numeral recognition. **Figure 12** shows the chosen neural networks with these selected functions.

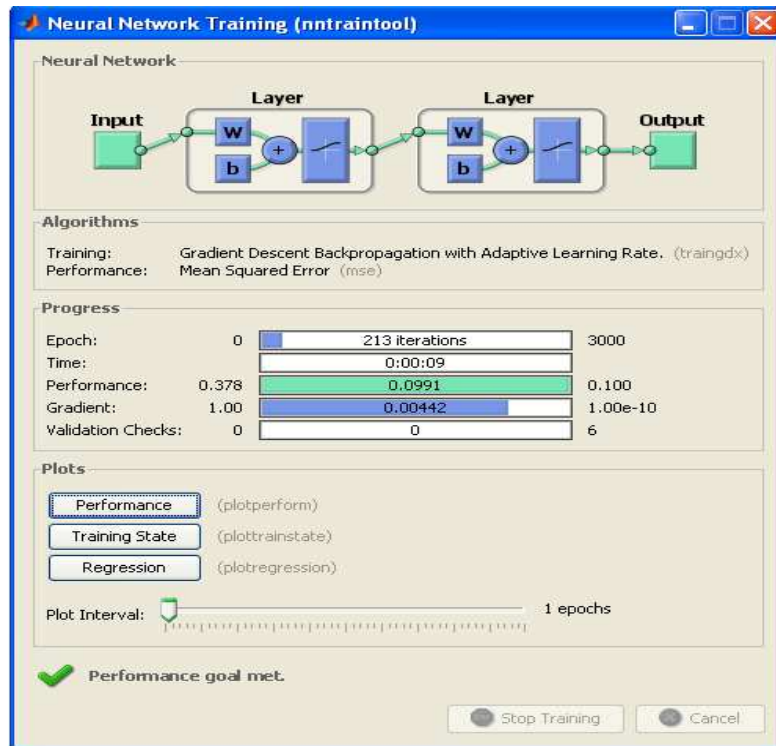


Fig. 12. NN with one hidden layer

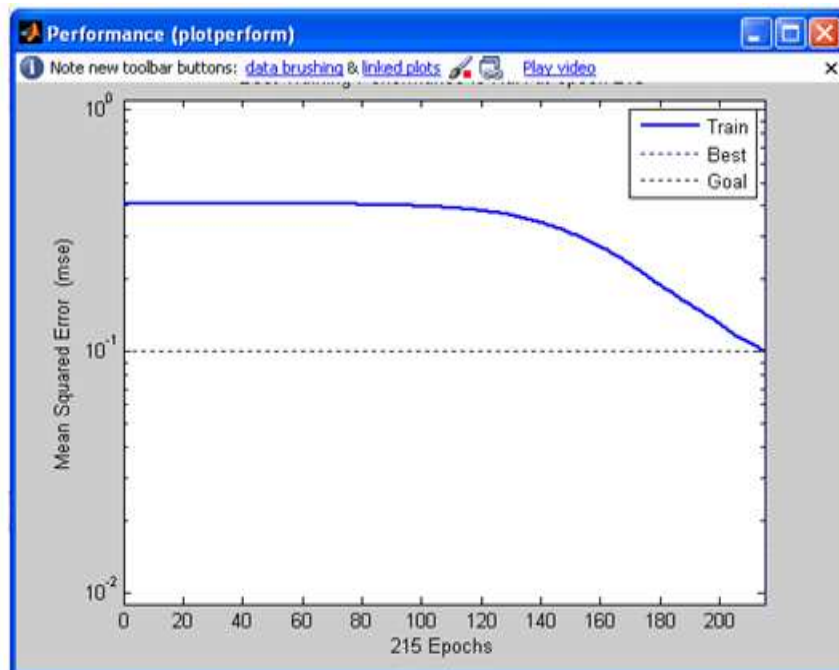


Fig. 13. The performance NN with one hidden layer using MSE and logsig

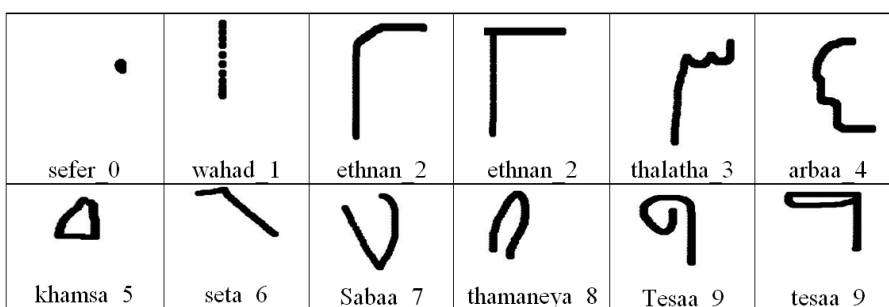


Fig. 14. Some examples of poorly written digits and correctly classified

Table 3. The function `trainidx` parameters

Training parameters	Description
<code>net.performFcn = 'mse';</code>	Mean square error
<code>net.trainParam.goal = 0.1;</code>	Performance goal
<code>net.trainParam.show = 20;</code>	Epochs between displays
<code>net.trainParam.epochs = 3000;</code>	Maximum number of epochs to train
<code>net.trainParam.mc = 0.95;</code>	Momentum constant

Table 4. Recognition accuracy testing results

Hindi no.	Accuracy (%)
Sefr-0	100
wahad-1	97
ethnan-2	93
thalatha-3	90
arbaa-4	87
khamsa-5	93
seta-6	87
sabaa-7	93
thamanya-8	87
tesaa -9	87
Accuracy of all digits	91

Figure 13 represents a learning curve for our model. The training error is plotted versus the number of epochs used in training data in order to show how fast a neural network model is learning. In our example, the goal is specified to be less than 0.1. **Figure 13** shows number of epochs needed to train the neural network in the training process. At the beginning the (MSE) was big, but after 200 epochs, the error of the network became 0.0991 and the network could recognize the training digits with a very small error.

The (`trainidx`) function is used to adjust the weights and the bias in the different layers depending on the gradient in the output error. The activation function used is Logistic-sigmoid, which is the most typical activation functions that has a range of (0, 1) for both hidden and output nodes. The objective of choosing these functions is to minimize the error in the output. The

Mean Squared Error (MSE) is used to judge the network performance to perform a numeral recognition. The parameters used in the `trainidx` is shown in **Table 3** with their values.

The network is simulated with the training images. The 800 digits which used to train the network are correctly classified; all these digits are classified in the correct class. Then 300 images, 30 images in each class that are not used in the training process which are collected from different persons, are tested. Each digit is tested alone, it took time to finish this process for each digit (0:00:08-0:00:10 sec). It was a brief little time for this process.

Table 4 shows high recognition rates for all digits 91%. The digit (0) achieved the best average accuracy 100% and the digit (1) had a high average accuracy 97%. But the digits (4, 6, 8 and 9) had the least average accuracy (87%).

The results shows that the network was capable to recognize the handwritten of Hindi digits using MLPs with backpropagation and a number of hidden nodes equal to 100 with a good percentage. These results, which depend on the elements, have been used to train the network. In this network, only 80 elements used to train the network for each class (800 for all classes). Using more digits to train the network, the result should be better. Both the feature selection and extraction have an effect on the recognition process. It is the most important and the hardest part of the classification.

The reason for inaccuracy was the use of the poorly written digits, because we used the mouse which is the most difficult handwritten tool especially for children and deliberately written in some strange and unusual way. Although these examples were poorly written, they were classified correctly as shown in **Fig. 14**. On the other hand, **Fig. 15** shows some examples of digits were poorly written, they were not classified correctly. Most of these samples were taken from children with the ages less than 5 years old. They have some problems by using the mouse to draw the numbers.











 wahad_1 Recognized as 0	 ethnan_2 Recognized as 3	 Thalatha_3 Recognized as 2	 arbaa_4 Recognized as 3	 khamsa_5 Recognized as 8
 arbaa_4 Recognized as 2	 seta_6 Recognized as 2	 sabaa_7 Recognized as 9	 thamaneya_8 Recognized as 6	 tesaa_9 Recognized as 7

Fig. 15. Some examples of poorly written digits and badly classified

3. RESULTS AND DISCUSSION

The tested data representing the Hindi Digits 0-9 collected from many users and different ages and MLP was built for recognizing these data. After training, which was an essential part in comparisons with other approaches, the network accuracy of 800 samples was 100%. The testing accuracy was found out to be 91% in average of 300 samples of digits, but we have the digit sefer_0 with the best accuracy result, it was 100%. The digit (1) has a high accuracy result, it was 97%. The time needed to recognize each digit was between (0:00:08-0:00:10 sec.), which was a fast time for this process.

Table 5 shows the recognition resulted from different approaches. Comparison between the proposed approach with the results from other approaches as follows.

Has better performance in comparison with the neural network used by LeCun *et al.* (1989). The accuracy of the system depends on many factors like whether there is noise in the test data, if the digit is poorly written, deliberately written in some strange and unusual way. It should take into account the writing process itself, it is subjective and depends on the person writing style. If the test data are carefully selected then the system could give a higher accuracy rate. Despite of these factors, our system has the advantage, which is that it could be modified to work on letters and other characters.

The implemented system has better performance in comparison with the Mouse Gesture Hindi Digit MGH (Shilbayeh *et al.*, 2009). MGH system used (FCC) with eight connectivity and then a template to recognize the Hindi digits. Their results show an accuracy rate of about 89% on the sample of 1350 digits and the time needed to recognize each digit was (39.81-40.68 sec). So, the proposed neural network gives better accuracy and less time in recognition.

Ahmad and Maen (2008) used the structural approach for recognizing on-line handwritten digits, recognizing the handwritten digits by using the primitives to determine and identify the changes in the slope's signs around the zero and the infinity values (break points). This technique is independent of the type of drawing (upward or downward). A special grammar has been used to match the string of primitives to the corresponding digit. The proposed method is tested on a sample of 3000 digits written by 100 different persons. The method achieved accuracy of about 95% on the sample test. This shows that the technique used can achieve a high recognition accuracy just for the shapes of the digits represented in his study. From the testing process we have noticed that the drawing speed may affect the recognition process. If the user draws very quickly, the system might not capture all the input pixels representing the digit, the drawing must be connected, so the user has to draw the digit as one connected line.

Mahmoud and Awaida (2009) described a technique for automatic recognition of off-line handwritten Hindi digits using Support Vector Machines (SVM) and Hidden Markov Models (HMM). The research used a database of 44 writers with 48 samples of each digit totaling 21120 samples. The SVM and HMM classifiers were trained with 75% of the data and tested with the remaining data. The average recognition rates were 99.83 and 99.00%, which was a very high result. These results may be due to the following reasons: Using the scanner for inserting the image as input instead of the mouse. It is worth mentioning that the samples are written by pen and also they used very large and different samples for training, so using more samples and choosing them carefully for training will increase the recognition rate.

Table 5. The average recognition rate for different approaches

Average Name	Approach	Features used	Handwritten type	recognition rate
LeCun <i>et al.</i> (1989)	Backpropagation neural network	Shared-weight averaging	Zip code	87-89%
Shilbayeh <i>et al.</i> (2009)	FCC with eight connectivity	FCC with 8 connectivity	Hindi digits	89%
Ahmad and Maen (2008)	Structural features and transition network	Shape-based features like curve, line, dot	Arabic digits	95%
Ahangar and Ahangar (2009)	Multilayer perceptron with one hidden layer	used global input	Farsi characters	80%
Mahmoud and Awaida (2009)	SVM and HMM classifiers	Local, intermediate and large scales features	Hindi digits	99.83%
Al-Omari <i>et al.</i> (2009)	Feed forward back propagation neural network algorithm	Feature extraction is not part of their work	Arabic digits	95%

Al-Omari *et al.* (2009) presented a system for dealing with this problem. The system started by acquiring an image containing Arabic digits. This image was digitized using some optical devices. After applying some enhancements and modifications to the digits within the image, it could be recognized using feed forward back propagation algorithm. The studies were conducted on the Arabic handwriting digits of 10 independent writers, who contributed a total of 1300 isolated Arabic digits. These digits were divided into two data sets: Training 1000 digits, testing 300 digits. An overall accuracy meet using this system was 95% on the test data set used. As we mentioned before, using more samples for training will increase our percentage.

Ahangar and Ahangar (2009) have made an attempt to recognize handwritten Farsi characters by using a multilayer perceptron with one hidden layer as the proposed system. The system has been trained using several different forms of handwriting provided by both male and female participants of different age groups. The experiments were carried out on 250 samples of 5 writers. The results showed that the MLP networks, trained by the error backpropagation algorithm, are superior in recognition accuracy and memory usage. The MLP networks take longer training time because they use iterative training algorithm such as EBP, but shorter classification time because of simple dot-product calculations. The result indicates that the backpropagation network provides recognition accuracy about 80% of handwritten Farsi characters.

It was found that the recognition accuracy depends on how good and how large the data sets are and which classifiers are used.

4. CONCLUSION

In this study, a handwritten recognition system has been implemented to detect Hindi digits. The implemented system is constructed using an MLP neural

network classifier with backpropagation. An image collected from many users and selected from different ages (including children). The handwritten input tool was the mouse which is the most difficult way in writing.

In this study, many variations proposed to improve the standard backpropagation algorithm as well as other learning algorithms. The proposed system has been trained on samples of 800 images and tested on samples of 300 images. An experimental results shows high accuracy results of about 91% on the sample test and very close to 100% on the sample of training test. In the testing set, the time needed to recognize each digit was between (0:00:08-0:00:10 sec.) which was a fast time for recognition. From the testing results it was concluded that the system has more trouble in identifying numerals 4, 6, 8 and 9. The overall recognition system was very high in using the mouse as a tool for writing all samples.

To further improve the recognition rate, possible future works are as follows. (1) Increasing number of training and testing data set. (2) combining multiple classifiers can improve the recognition accuracy but decrease the speed of the recognition.

5. REFERENCES

- Ahangar, R.G. and M.F. Ahangar, 2009. Handwritten farsi character recognition using artificial neural network. *Int. J. Comput. Sci. Inform. Sec.*, 4: 1-2.
- Ahmad, A.T. and H. Maen, 2008. Recognition of on-line handwritten Arabic digits using structural features and transition network. *Informatica*, 32: 275-281.
- Al-Omari, S.A., P. Sumari, S.A. Al-Taweel and A.J. Husain, 2009. Digital recognition using neural network. *J. Comput. Sci.*, 5: 427-434. DOI: 10.3844/jcssp.2009.427.434
- Al-Rashaideh, H., 2006. Preprocessing phase for Arabic word handwritten recognition. *Inform. Transmissions Comput Netw.*, 6: 11-19.

- Fausett, L., 1994. Fundamentals of Neural Networks Architectures, Algorithms and Applications. 1st Edn., Prentice Hall, Englewood Cliffs, ISBN-10: 0133341860, pp: 461.
- Geman, S., E. Bienenstock and R. Doursat, 1992. Neural networks and the bias/variance dilemma. *Neural Comput.*, 4: 1-58. DOI: 10.1162/neco.1992.4.1.1
- Golda, A., 2005. Principles of training multi-layer neural network using backpropagation.
- LeCun, Y., B. Boser, J.S. Denker, D. Henderson and R.E. Howard *et al.*, 1989. Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, 1: 541-551. DOI: 10.1162/neco.1989.1.4.541
- Mahmoud, S. A. and S.M. Awaida, 2009. Recognition of off-line handwritten Arabic (Indian) numerals using multi-scale features and support vector machines vs. hidden markov models. *Arabian J. Sci. Eng.*, 34: 429-444.
- Mashor, M.Y. and S.N. Sulaiman, 2001. Recognition of noisy numerals using neural network. *Int. J. Comput. Internet Manage.*, 9: 45-52.
- Ouchtati, S., M. Bedda and A. Lachouri, 2008. Segmentation and recognition of handwritten numeric chains. *J. Comput. Sci.*, 3: 242-248. DOI: 10.3844/jcssp.2007.242.248
- Shilbayeh, N. and M. Iskandarani, 2005. An intelligent multilingual mouse gesture recognition system. *J. Comput. Sci.*, 3: 346-350. DOI: 10.3844/jcssp.2005.346.350
- Shilbayeh, N., G. Raho and M. Alkateeb, 2009. An efficient structural mouse gesture approach for recognizing Hindi digits. *J. Applied Sci.*, 9: 3469-3479. DOI: 10.3923/jas.2009.3469.3479
- Stergiou, C. and D. Siganos, 1996. Introduction to neural networks.