Recognition Offline Handwritten Hindi Digits Using Multilayer Perceptron Neural Networks

NIDAL F. SHILBAYEH* MUSBAH M. AQEL** AND REMAH ALKHATEEB***
*Department of Computer Science, University of Tabuk,
Tabuk, Saudi Arabia,
nshilbayeh@ut.edu.sa
**Department of Computer Science, AlZarqa University,
Zarqa, Jordan,
Aqelm06@yahoo.com
***Department of Computer Science, Middle East University,
Amman, JORDAN

Abstract: - Handwritten Hindi digit recognition plays an important role in eastern Arab countries especially in the courtesy amounts of Arab bank checks. In this paper, we proposed an efficient offline handwritten Hindi digits recognition system and developed using Multilayer Perceptron Neural Network (MLP). The implemented system recognizes separated handwritten Hindi digits scanned using a scanner. The system has been designed, implemented and tested successfully. The error backpropagation algorithm has been used to train the MLP network. An analysis shows increasing in the recognition rate by doing some enhancements. The proposed system has been trained on samples of 1000 images and tested on samples of 600 images written by different users selected from different ages. The samples were scanned and preprocessed before entering the Neural Network for recognition. The An experimental result showed a very good recognition rate in comparison with other Hindi digit recognition systems that use the same recognition classifier tool especially if we consider the way of writing and number of training and testing samples. In addition, the recognition rate could be increased if we have a high resolution scanner and increase the number of training and testing samples.

Key-Words: - Offline Hindi Digit Recognition, Hindi Digits, Feature Extraction, MLP, Neural Networks, Backpropagation.

1 Introduction

Handwritten digits and character recognition recently have been an interesting domain for researchers who were interested in recognizing characters and digits in many languages. Handwritten character recognition is divided into two parts; online character recognition and offline character recognition. The online handwritten recognition involves the automatic conversion of text as it is written on a special digitizer or PDA, where a sensor picks up the pen-tip movements as well as pen-up/ pen-down switching. Online handwritten uses not only pens in PDAs, but also mouse gesture in personal computers and fingers in some cases of touch screens. The offline handwritten recognition is more different than the online since the handwritten will be written using a pen and a paper.

The texts and digits written by hand will be entered into the computers using many acquisition devices such as, scanners, digital cameras, and others. In offline, the recognition will be much harder than in online since the images are taken by acquisition devices which cause some effects on images like noises, different sizes, different colors, and others. These effects require a much harder preprocessing stage to deal with in the case of offline handwritten recognition. In this paper the recognition will be on recognizing Hindi digits, where the researches around Hindi digits are limited especially in the case of offline handwritten Hindi digits. These Hindi digits are shown in table1.

٠	١	٢	٣	٤	٥	٦	٧	٨	٩

Table 1 Hindi Digits from Zero to Nine

2 Related Work

Handwritten recognition in general has been an attractive field for researchers for many years; the researchers have been interested in this field for recognizing many different languages in many different countries. In specific the applications of handwritten recognition also attract researchers in this field. These applications are bank check recognition, postal code recognition, word document recognition and many others. Bank check is an attractive application that interested many researchers from different countries, bank checks have been recognized for many countries such as, Bengal, Brazil, China, France, Italy, Japan, Korea, Tunisia, the United States and many others[10, 13].

Handwritten digits and character recognition are important for all languages. Researchers have worked through many languages such as English, Spanish, French, and many others . But the work has been limited for some languages such as Arabic, Farsi, and others. In [12] they were interested in the recognition of Farsi numerals which are similar to Arabic ones.

In the state of art many works have been introduced in the field of handwritten recognition. In [1-3] a recognizer was designed to recognize Hindi digits that were written by a mouse. Another work was introduced by [5, 9] in which they presented a method for recognizing isolated handwritten Arabic/Persian digits. This method is based on Support Vector Machines (SVMs). In [4] the authors presented a system that recognizes online handwritten Hindi digits written by a mouse. There are many works for recognizing digits for different languages such as English, Bengali, Brazilian, French and many others, using many different classifiers such as Multi-Layer Perceptrons (MLPs), Support Vector machines (SVMs), Hidden Markov Model (HMM), and many others.

In addition to that, some other researchers designed a system based on cross pruning (CP) algorithm to optimize number of hidden neurons and number of iterations in Multi-Layer Perceptron (MLP) neural based recognition system. Their approach is to study the effect of varying the size if the network hidden layers (pruning) and number of iterations (epochs) on the classification and performance of the used MLP [2].

3 Offline Handwritten Hindi Digit Recognition System

The offline handwritten Hindi digit recognition system architecture is shown in fig.1.

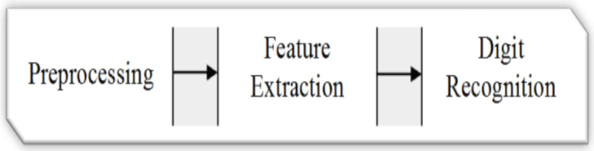


Fig. 1 Block Diagram for Handwritten Hindi Digits Recognition System

3.1 Preprocessing

The preprocessing stage concerns itself with processing input data to produce output data that is used as input to another stage. The importance of the preprocessing stage lies in preparing the digit to be in its final shape before entering the recognition stage. The images were first scanned using a scanner with a 300 dpi. These scanned images were colored images; the size of the samples was not defined and differs from one image to another since the digits were written in different handwriting styles from different people of different ages. There are many steps in the preprocessing stage which are shown in the fig.2.

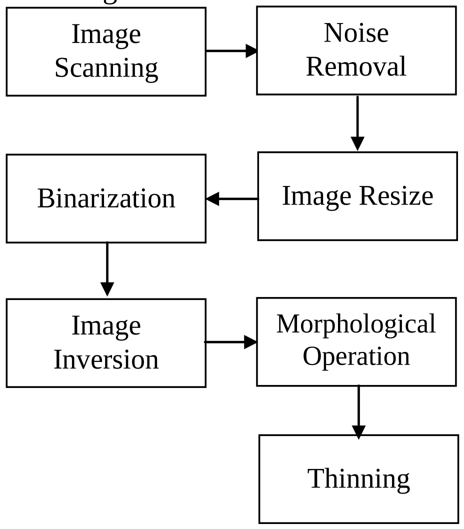


Fig. 2 Block Diagram for Hindi digits Preprocessing Stage

- **Image Scanning**: using a scanner with 300 dpi, the resulting image is colored, very large, and has some noise.

- **Noise Removal:** Removing the noise is done by applying the median filter. The noise coming from the scanner couldn’t be removed totally by the median filter. Therefore, the noise was removed by adjusting the contrast and sometimes the brightness using the windows picture manager. The fig.3 shows an image before and after noise removal.

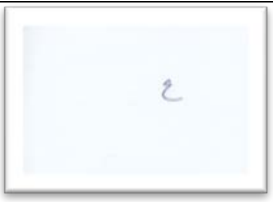
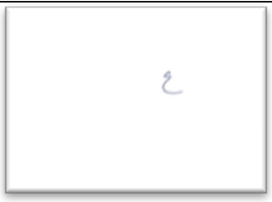
	

Image Before Noise Removal	Image After Noise Removal

Fig. 3 Sample before and after noise removal step

- **Image Resize:** Resizing the image using down sampling and interpolation bilinear methods. Interpolation method was used to ensure that the image will not lose too many pixels because losing too many pixels, results in losing some features of the original shape of the image. The image is resized in its final size to [180 180]. It was the best size to show the image without losing too many pixels.

- **Binarization**: it is the process of converting the image into black and white. The word binarization is coming from binary. To convert the image into binary image means to make all pixels zeros and ones. The technique used to convert the image to black and white is **(level = graythresh(image);)** this function from the Image Processing tool box in matlab is used to compute a global threshold (level) that can be used to convert an intensity image to a binary image. **(level)** is a normalized intensity value that lies in the range [0, 1]. The graythresh function uses Otsu's method, which exhaustively search for the threshold that minimizes the intra-class variance, defined as a weighted sum of variances of the two classes. The following description for the Otsu’s Method and Algorithm that were used for binarization:

Otsu’s Method [11]: Otsu’s Method expressed in equations 1 and 2

$$\sigma_w^2(t) = \omega_1(t)\sigma_1^2(t) + \omega_2(t)\sigma_2^2(t) \qquad (1)$$

Weights ω_i are the probabilities of the two classes separated by a threshold t and σ_i^2 variances of these classes.

Otsu shows that minimizing the intra-class variance is the same as maximizing inter-class variance:

$$\sigma_b^2(t) = \sigma^2 - \sigma_w^2(t) = \omega_1(t)\omega_2(t)\left[\mu_1(t) - \mu_2(t)\right]^2 \qquad (2)$$

Which is expressed in terms of class probabilities ω_i and class means μ_i which in turn can be updated iteratively. This idea yields an effective algorithm.

Otsu’s Algorithm [11]:

1. Compute histogram and probabilities of each intensity level
2. Set up initial $\omega_i(0)$ and $\mu_i(0)$
3. Step through all possible thresholds t=1..maximum intensity
    a. Update ω_i and μ_i
    b. Compute σ^2_b (t)
4. Desired threshold corresponds to the maximum σ^2_b (t).

- **Image Inversion:** image inversion means inversing the pixels of an image the zeros pixels to ones and vice versa. Inversing the Image is necessary since after applying the binarization, the background of the image takes the ones pixels and the foreground of the image takes the zeros pixels. As known in Image Processing using matlab, they deal with image as ones pixels. Therefore, inversion is necessary to give the objects or digits in the image the ones and give the background the zeros. In this stage the inverse function is applied to the image for this purpose.

- **Morphological Operations:** Applying some morphological operations on the image in order to enhance the image and remove distortions from the image. A set of closing and dilating with a structuring element of one pixel is applied on images. Closing operation is used to close the holes in the image, while dilate is used to thicken the image and connect broken lines.

- **Thinning stage**: that uses the thinning algorithm to bring the digit into one pixel thick.

Thinning Algorithm: Sivanandam, Sumathi & Deepa, (2006) The thinning algorithm in matlab is applied using the 'thin' function; the following syntax for using the thinning algorithm. BW = bwmorph (bw, ‘thin’, inf). This syntax means that the image will be thinned until infinity. Infinity means that the image will have its skeleton shape with one pixel thick. The following is the algorithm described in steps:

1. Divide the image into two distinct subfields in a checkerboard pattern.
2. In the first sub iteration, delete pixel **p** from the first subfield if and only if the conditions **G**$_1$, **G**$_2$, and **G**$_3$ are all satisfied.
3. In the second sub iteration, delete pixel **p** from the second subfield if and only if the conditions **G**$_1$, **G**$_2$, and **G**$_3$**'** are all satisfied.

Condition G1:

$$X_H(p) = 1$$

Where

$$X_H(p) = \sum_{i=1}^{4} b_i \qquad (3)$$

$$b_i = \begin{cases} 1 \text{ if } x_{2i-1} = 0 \text{ and } (x_{2i} = 1 \text{ or } x_{2i+1} = 1) \\ 0 \text{ otherwise} \end{cases} \qquad (4)$$

The values **x**$_1$, **x**$_2$... **x**$_8$ in equations 3 and 4 are the values of the eight neighbors of **p**, starting with the east neighbor and numbered in counter-clockwise order.

Condition G2:

$$2 \le \min\{n_1(p), n_2(p)\} \le 3$$

Where

$$n_1(p) = \sum_{k=1}^{4} x_{2k-1} \vee x_{2k}$$

and

$$n_2(p) = \sum_{k=1}^{4} x_{2k} \vee x_{2k+1} \qquad (5)$$

Condition G3:

$$(x_2 \vee x_3 \vee \bar{x}_8) \wedge x_1 = 0 \qquad (6)$$

Condition G3':

$$(x_6 \vee x_7 \vee \bar{x}_4) \wedge x_5 = 0 \qquad (7)$$

In this algorithm after applying all equations the user will get the skeleton shape of an image since the iterations are repeated until the image stops changing.
After applying all the steps of the preprocessing stage on the image the image will have its final shape that will be as an input to the recognition and classification stages. Fig.4 shows an image of the digit four after passing all preprocessing stages.

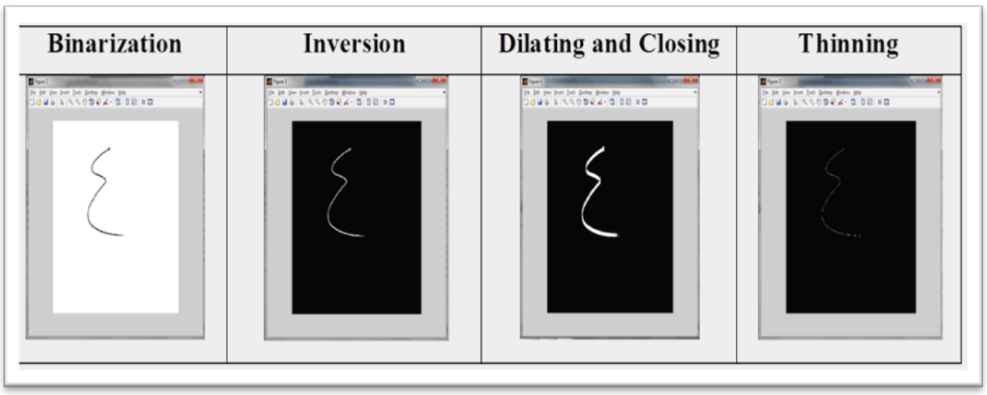


Fig.4 digit four after passing preprocessing stages

3.2 Feature Extraction

This stage is very important in the recognition stage. The digit is extracted depending on its blob analysis. Blob means the group of pixels that make up the digit. To get its analysis means to measure its properties such as area, center, perimeter, diameter, intensity, and many others. In feature extraction it determines the size of the input vector that enters the neural network. Instead of taking the whole image, features of the image will be extracted depending on its properties. These features implement the input vector to the neural network stage. In this system the image of size 180*180 pixels is divided into 9 blocks, each block of size 60*60 pixels.

The following three functions have been used to extract the features from each block:

- The first function is the region props function: this function depends on the properties of the image. The properties taken for each region are the area, center, perimeter, diameter, and the bounding box. The function first labels each region in the image. Then it calculates the properties of each region. The main property is taken into account here is the bounding box which is the smallest box containing the region. After the bounding box is taken, the features extracted will be of different sizes since each digit has different bounding box containing it depending on its geometrical shape.
- The standard deviation and average functions. These functions are calculated to normalize the features extracted by the region props function mainly the features extracted using the bounding box.

The features extracted from the region props function will be a vector of different sizes because the properties of each image differ from image to another. Therefore, the average and standard deviation are taken to normalize the size of the vector for all digits or images. The vector will be of size 1*18 for all images. This vector will be the input to the neural network for recognition stage.

3.3 Digit Recognition

In this stage a Multilayer Perceptron NN with backpropagation architecture was designed for digit classification and recognition. The used architecture has been chosen since it’s the most widely used in handwritten recognition fields. The rest of this section is a description of the Neural Network architecture in details, training the network, and testing it.

1. Neural Network Architecture:

The designed NN architecture consists of an Input layer, one hidden layer, and an output layer. The performance function used for measuring the network’s performance is the MSE (Mean Square Error) which is the average squared error between the network outputs and the target outputs. The MSE formula is described in equation 8:

$$F = mse = \frac{1}{N}\sum_{i=1}^{N}(e_i)^2 = \frac{1}{N}\sum_{i=1}^{N}(t_i - a_i)^2 \qquad (8)$$

The activation function used for both hidden and output layers is the log-sigmoid transfer function. This network architecture was chosen after a deep study of designing many other networks with different numbers of hidden layers, different numbers of neurons, and different types of activation functions. After training those different networks, it appeared that the network that was chosen is the best among others in performance and minimum error rate [4]. The training algorithm used in this architecture is the Gradient Descent Backpropagation with adaptive learning rate. The network is designed and trained to recognize the Offline Hindi Digits from Zero to Nine. Fig.5 shows the architecture of the designed network.

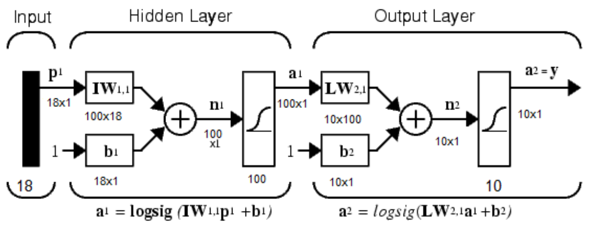


Fig. 5: The Neural Network Architecture

The neural network needs 18 inputs in the input layer and 10 neurons in its output layer to identify the Hindi Digits. The network is a two-layer log-sigmoid/log-sigmoid network. The log-sigmoid transfer function was picked because its output range (0 to 1) is perfect for learning to output Boolean values. The log sigmoid transfer function in equation 9 calculates a layer's output from its net input. Fig.6 shows the log-sigmoid graph and symbol.

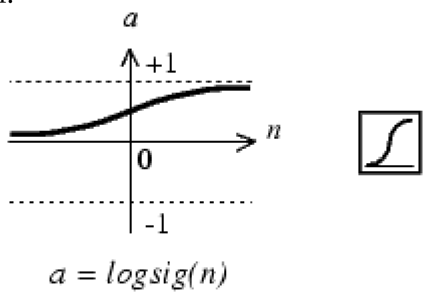


Fig.6 Log-Sigmoid Graph and Symbol

Activation Function Algorithm

$$\text{Logsig}(n) = 1 / (1 + \exp(-n)) \qquad (9)$$

The network consists of three layers:

- Input Layer: The Input Layer consists of 18 Boolean values as 18-element input vector. These 18 element input vectors come from dividing the original image of size 180*180 into

3*3 blocks. Each block is of 60*60 pixels, and for each block the properties are calculated for each region and then average and standard deviation are calculated to normalize the features extracted by the region props function as mentioned in the previous section. When calculating both the average and the standard deviation for all 9 blocks, we get two (the average and the standard deviation) multiplied by Nine (the total number of blocks in the image) the result is 18 element input vector.

Nine (total number of blocks in the image) * Two (average and standard deviation for normalizing features) = 18 (input vector).

- Hidden Layer: the hidden layer has 100 neurons. This number was chosen after trying many networks with different numbers of hidden neurons and by finding the minimum mean square error (MSE). In most cases choosing the number of hidden neurons depends on guesswork and experience. It also depends on getting the least error rate after trying many networks with different number of hidden neurons.

- Output Layer: the output layer consists of 10 neurons to identify the 10 digits. The network is trained to output a one (1) in the correct position of the output vector and to fill the rest of the output vector with zeros (0s).

The Hindi digits will pass through the three layered NN architecture for recognition as shown in the following scenario in fig.7. The input sample is number three in Hindi numerals, it will be entered to the input layer as an 18 vector element, and then it will be transmitted to the hidden and output layers to give a correct and recognized output digit.

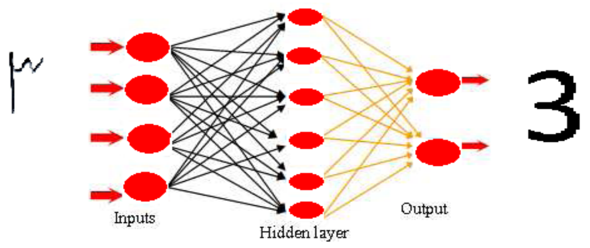
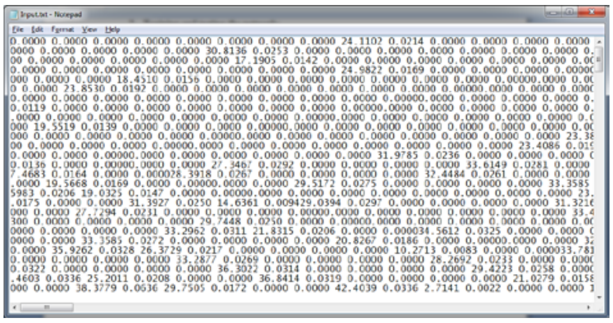


Fig.7 Recognition of Number 3 in the NN

2. Training and testing the network:

The Network has been trained and tested using samples of handwritten Hindi digits. The training and testing stages were achieved after the samples passed through the preprocessing and the feature extraction stages. The samples were stored as vectors. Each sample or image was stored as an 18 element vector; these vectors were stored into a text document. This text document was created to be the input set for training the network. The created input file contains the features for each digit. As shown in fig.8 the images were stored into a text document as vectors. These vectors were the result of the feature extraction function and they will be the input for the NN to recognize each digit.



```
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 24.1102 0.0214 0.0000 0.0000 0.0000 0.0000
0000 0.0000 0.0000 0.0000 0.0000 30.8136 0.0253 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.
00 0.0000 0.0000 0.0000 0.0000 0.0000 17.1905 0.0142 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 24.9822 0.0169 0.0000 0.0000 0.0000 0.0000
000 0.0000 0.0000 18.4510 0.0156 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.00000.0000 0.0
0 0.0000 23.8530 0.0192 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.00000.0000 0.0000 0.000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.00000.0000 0.0000 0.0000 0.0000 0.
0.0119 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.00000.0000 0.0000 0.0000 0.0000 0.0000 0.
.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.00000.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.
000 19.5519 0.0139 0.0000 0.0000 0.0000 0.00000.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0
000 0.0000 0.0000 0.0000 0.0000 0.00000.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 23.3
00 0.0000 0.0000 0.0000 0.00000.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 23.4086 0.01
0.0000 0.0000 0.00000.0000 0.0000 0.0000 0.0000 0.0000 0.0000 31.9785 0.0236 0.0000 0.0000 0.0000
0.0136 0.0000 0.00000.0000 0.0000 27.3467 0.0292 0.0000 0.0000 0.0000 0.0000 33.6149 0.0281 0.0000
7.4683 0.0164 0.0000 0.000028.3918 0.0267 0.0000 0.0000 0.0000 0.0000 32.4484 0.0261 0.0000 0.0000
.0000 19.5668 0.0169 0.0000 0.00000.0000 0.0000 29.5172 0.0275 0.0000 0.0000 0.0000 0.0000 33.3585
5983 0.0206 19.0325 0.0147 0.0000 0.00000.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 23.
.0175 0.0000 0.0000 31.3927 0.0250 14.6361 0.009429.0394 0.0297 0.0000 0.0000 0.0000 0.0000 31.321
000 0.0000 27.7294 0.0231 0.0000 0.0000 0.0000 0.00000.0000 0.0000 0.0000 0.0000 0.0000 0.0000 33.4
300 0.0000 0.0000 0.0000 0.0000 29.7448 0.0250 0.0000 0.00000.0000 0.0000 0.0000 0.0000 0.0000 0.0
0000 0.0000 0.0000 0.0000 33.2962 0.0311 21.8315 0.0206 0.0000 0.000034.5612 0.0325 0.0000 0.0000
0000 0.0000 33.3585 0.0272 0.0000 0.0000 0.0000 0.0000 20.8267 0.0186 0.0000 0.00000.0000 0.0000 3
0.0000 35.9262 0.0328 26.3729 0.0217 0.0000 0.0000 0.0000 0.0000 10.2713 0.0083 0.0000 0.000033.78
0.0000 0.0000 0.0000 0.0000 33.2877 0.0269 0.0000 0.0000 0.0000 0.0000 28.2692 0.0233 0.0000 0.000
0.0322 0.0000 0.0000 0.0000 0.0000 36.3022 0.0314 0.0000 0.0000 0.0000 0.0000 29.4223 0.0258 0.000
.4603 0.0336 25.2011 0.0208 0.0000 0.0000 36.8414 0.0319 0.0000 0.0000 0.0000 0.0000 21.0279 0.015
000 0.0000 38.3779 0.0636 29.7505 0.0172 0.0000 0.0000 42.4039 0.0336 2.7141 0.0022 0.0000 0.0000
```

Fig. 8 The input file entered to the NN for training

A description of the training and testing procedures and results are explained in details in the next section.

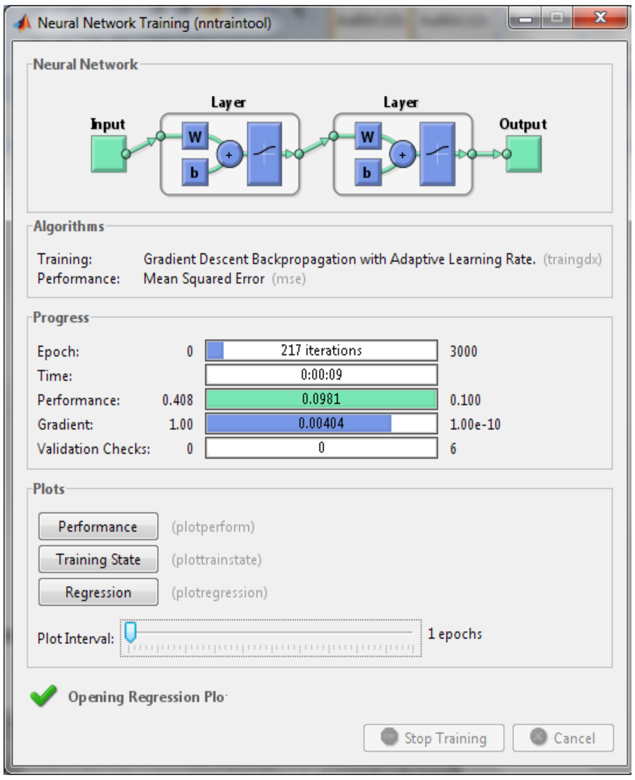
4 Results and Analysis

The developed recognition system is designed using Matlab R2008a, the Image Processing and NN Tool Boxes were mainly used. The system worked on samples of Hindi digits from zero to nine. The samples were written by hand and then entered to the computer using a scanner. The Image Processing Tool Box was used for manipulating the scanned images for preprocessing, segmentation, and feature extraction stages. The NN Tool Box was used for training and testing the samples for recognition stages.

In the Offline Handwritten Hindi Digits Recognition System, the digits were recognized as separate digits. The Hindi digits from one to nine were taken from real world samples. These samples were written by hand and they are divided into two types. The first type contains 1000 samples that were written using a regular blue pen. The second type contains 1000 samples that were written using a black marker. These samples must pass through all stages of the Offline Handwritten Hindi Digit Recognition system. The samples first pass through the preprocessing stage to enhance the handwritten digits, then they pass through the feature extraction stage that was used to extract the digits from images and store them as vectors to be the input for the NN. The NN structure used is the Multi-Layer Perceptron (MLP) with backpropagation training algorithm which consists of an input layer, one hidden layer, and an output layer. In the digit recognition stage the samples were used for training and testing. In the following paragraphs is a description for both training and testing stages and their results.

- **Training the network:**

The network is trained using two types of samples. The 1000 samples of Hindi digits that were written using a regular blue pen, and the 1000 samples of Hindi digits that were written using a black marker. In both cases the training results were excellent and the recognition rate ranges from 90% to 99%. In the training stage the samples first were entered to the network as input set. The digits were trained separately to record the rates accurately. While training a digit, a neural network training tool window is launched to show the progress while training as shown in fig.9, and to show the performance, training state, and the regression plot. Also the performance plot that is shown in fig.10 shows the network performance. Then the training state plot in fig.11 shows training state values which are Gradient, Validation, and Learning Rate. Finally the regression plot in fig.12 shows the linear regression of targets relative to outputs.


Neural Network Training (nntraintool)

Neural Network

Input — Layer — Layer — Output

Algorithms

Training: Gradient Descent Backpropagation with Adaptive Learning Rate. (traingdx)
Performance: Mean Squared Error (mse)

Progress

Epoch:	0	217 iterations	3000
Time:		0:00:09	
Performance:	0.408	0.0981	0.100
Gradient:	1.00	0.00404	1.00e-10
Validation Checks:	0	0	6

Plots

Performance (plotperform)
Training State (plottrainstate)
Regression (plotregression)

Plot Interval: 1 epochs

Opening Regression Plo

Stop Training Cancel

Fig. 9 Neural Network training tool window

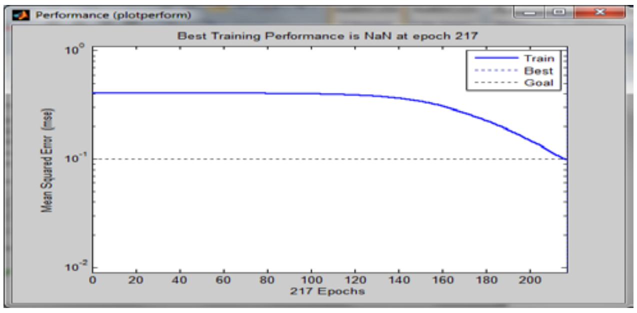


Fig.10 Performance Plot

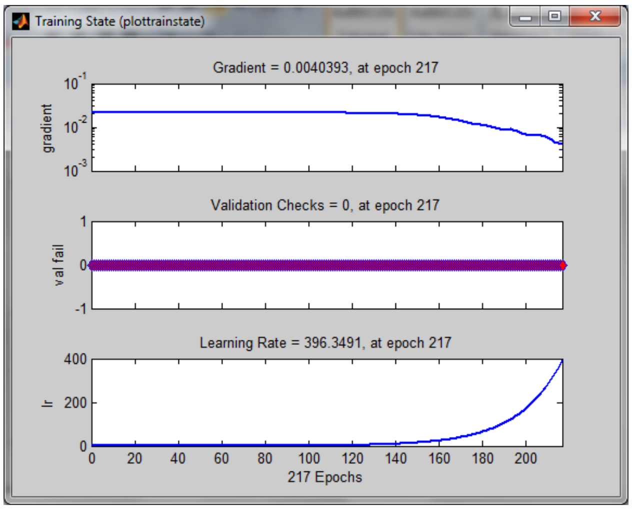


Fig.11 Training State plot

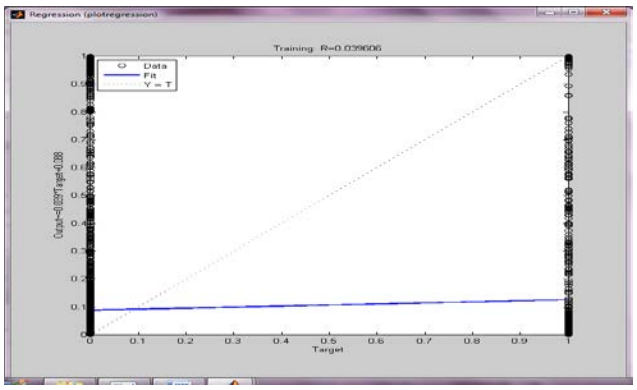


Fig.12 Regression plot

- **Testing the Network**

At this stage the network was tested using the testing samples. As mentioned previously there are two types of testing samples, 40 samples for each type. The samples that were written by a regular blue pen were written and tested before the samples that were written by a black marker. The reason behind this is the testing results. After testing the samples of a regular blue pen, the recognition rate was very bad and unexpected. Many solutions and attempts were introduced and applied to the samples in order to enhance the recognition rate, but the recognition rate didn't improve or increase. Therefore, other samples were written using a black marker to see if the recognition rate got better results. The samples were written by different people from different ages and having different writing styles. After testing the samples, the recognition rate increased significantly. It was 51.6%; however, this recognition rate was not as desired. Therefore, many attempts and solutions were introduced to enhance this rate. After many changes, the recognition rate improved gradually until it reached an acceptable rate of 74.4%. The rest of the section will be a description in details of how the recognition rate was enhanced gradually and got its final results.

First, the samples that were written by a regular pen were tested, but the recognition rate was very bad and unexpected. Therefore; many solutions were introduced to enhance the recognition rate.

First solution was performed on the preprocessing stage to change the size of the image. The imresize function was used to change the size. Using imresize directly was wrong since it deletes one raw with one column. Using this function changes the structure of the image and that's why the recognition rate was very low. Instead of using imresize function, down sampling with interpolation method was used. Also resizing the image into very small dimensions such as (30*30) or (60*60) was not acceptable since resizing causes losing too many pixels of the image which results in bad recognition.
The problem of resizing was solved, also many changes were performed to remove noise from images and to make the handwriting style appear clearer and thicker using the morphological operations. However, the recognition rate remained low and with no enhancement. After all the previous attempts, a decision was taken to write new samples using a black marker. The new samples consist of 1000 samples for training and 40 for testing. After writing the new samples, they were used for training and testing. As a result of testing the new samples, the recognition rate increased significantly and was recorded as 51.6%.

This result was a very big achievement compared with the results before changing the samples. However, the achieved recognition rate was not as desired, therefore, some changes were performed in order to increase the recognition rate. After trying many changes in the preprocessing stage, something was noticeable, which is the thinning step. The thinning step lets the digit have its skeleton shape. This step resulted in losing pixels from the original image and affected the shape of the original one. Therefore, the thinning step was excluded from the preprocessing stage as an attempt only to see if the recognition rate gets better. After these changes, training and testing were performed again. As a result of this change the recognition rate really increased and got recognition rate of 63.1%. Still this result was not as desired so another solution was taken in order to increase it. This step was increasing the testing samples. The testing samples were increased by adding 20 more samples to each digit to have a total of 60 samples for each digit. After this addition, the testing was performed again and the recognition rate increased and got better results than before. The recognition rate became after increasing the samples 74.4%. As described above, the testing stage passed through many changes in order to enhance the recognition rate. The recognition rate increased gradually after each step of enhancement until it reached the last recognition rate of 74.4%.

The following three steps will show the recognition rate resulted after the changes done for enhancement:

1. In table 2 shows the results of the recognition rate of each digit and the total recognition rate after using new samples written by a black marker.

Digit	Recognition rate
Zero	100%
One	87%
Two	42%
Three	35%
Four	37%
Five	53%
Six	45%
Seven	39%
Eight	58%
Nine	20%
Total Recognition rate:	**51.6%**

Table 2 Recognition Rate after changing the samples using a black marker

2. In table 3 shows the recognition rate after the second change which is removing the thinning step from the preprocessing stage.

ISBN: 978-960-474-336-0

Digit	Recognition rate
Zero	100%
One	60%
Two	37%
Three	55%
Four	55%
Five	42%
Six	80%
Seven	65%
Eight	80%
Nine	57%
Total Recognition rate:	**63.1%**

Table 3 Recognition Rate after removing the thinning step

3. Table 4 shows the last step applied for enhancing the recognition rate which is increasing the testing samples from 40 for each digit to 60 samples.

Digit	Recognition rate
Zero	100%
One	73%
Two	73%
Three	75%
Four	65%
Five	60%
Six	82%
Seven	73%
Eight	86%
Nine	57%
Total Recognition rate:	**74.4%**

Table 4 Accuracy of each digit and the total recognition rate

The Offline Handwritten Hindi Digits Recognition subsystem was designed for recognizing separate Hindi digits written by hand and entered to the system by a scanner. These digits passed through the preprocessing and feature extraction stages before recognition. Multilayer Perceptron (MLP) architecture was used to recognize these digits, and a recognition rate of 74.4% was attained after many attempts to enhance this rate.

The table 4 shows some comparative results of different approaches and different related works for the recognition of isolated digits or characters.

Name	Approach	Handwritten type	Recognition Rate
[6]	Offline Arabic Character Recognition Using RBF Network	Offline	76%
[5]	Handwritten Arabic/Persian digits Using Support Vector Machines (SVMs).	Offline	94.14%
[4]	Recognition of Hindi digits using Neural Networks	Online	91%

Table 4 The average recognition rate for different approaches

These results will be discussed and compared with the Offline Handwritten Hindi Digit recognition subsystem.

- In [6] they used the RBF networks for recognizing Arabic Characters. RBF is Radial-Basis Function Networks. In implementing their function they used the Brain Construction Kit (BCK) tool. BCK is a Java Package that enables users to create, train, modify, and use Artificial Neural Networks. They obtained a recognition rate of 76%. On the other hand, the designed subsystem is different in approach, it used the MLP architecture using the NN tool box in matlab, it was designed to recognize Hindi digits, and it obtained a recognition rate of 74.4% and the results could be improved by increasing the training and testing samples.
- In [5] system for offline handwritten Arabic and Persian digits recognition is designed using Support Vector Machines (SVM). They achieved a high recognition rate. Such rate was due to the use of the CENPARMI Indian digit database. This database contains 7390 isolated digits for training set, and 3035 digits for testing sets, and all these digits have been collected from real bank checks. On the other hand, the database used in our built system is very small compared to the CENPARMI database. It contained 1000 samples for training and 600 samples for testing.
- In [4] they designed a system to recognize online Hindi digits using Neural Networks. In contrast with our system, it recognized offline Hindi digits. In online, the digits are written directly on the screen and saved as an image. In online there is no noise and no size problems since the user determines the handwriting space. Also, less preprocessing problems appear. In offline, the digits are written on

papers, the papers entered by a scanner, noise problems result from scanners and size problems appeared since the size will be very large and resizing images may result in loosing pixels and changing the shape of the digit. Also many other preprocessing problems appeared. All these problems affect the efficiency of the recognition systems and explain the differences in recognition rates achieved.

5 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. The proposed system has been trained on samples of 1000 images and tested on samples of 600 images. This system achieved successfully the purpose of recognizing separate Hindi digits with a good recognition rate of 74.4%. The result showed a very good recognition rate in comparison with other Hindi digit recognition systems that use the same recognition classifier tool especially if we consider the way of writing and number of training and testing 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.

Acknowledgment

The authors would like to acknowledge financial support for this work from the Deanship of Scientific Research (DSR), University of Tabuk, Tabuk, Saudi Arabia, under grant no. 0061/1434/S

References:

[1] Shilbayeh, N.F, and Iskandarani, M.Z., An intelligent multilingual mouse gesture recognition system, *Journal of Computer Scien*ce, Vol.1, No.3, 2005, pp. 346-350.

[2] Shilbayeh, N.F and Iskandarani, M.Z., Effect of Hidden Layer Neurons on the Classification of OCR Typed Arabic Numerals, *Journal of Computer Science*, Vol.4, No.7, 2008, pp. 578-584.

[3] Shilbayeh, N.F, Raho, G. and Alkhateeb, M., An Efficient Structural Mouse Gesture Approach for Recognizing Hindi Digits, *Journal of Applied Science*, Vol.9, No.19, 2009, pp. 3469-3479.

[4] Shilbayeh, N.F, Alwakeel, M.M. and Naser, M.M., An Efficient Neural Network for Recognizing Gestural Hindi Digits, *American Journal of Applied Science*, Vol. 10, No. 9, 2013, pp. 938-95.

[5] Sadri, J., Suen, C.Y., and Bui, T.D., Application of support vector machines for recognition of Handwritten Arabic/ Persian digits. In: Proceedings of Second Iranian Conference on Machine Vision and Image Processing, vol. 1, 2003, pp. 300–307.

[6] Nawas, S.N., Sarfraz, M., Zidouri, A., and Al-Khatib, W.G., An Approach to Offline Arabic Character Recognition Using Neural Networks, *IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, Vol. 3, 2003, pp. 1328 – 1331.

[7] Demuth, H., Beale, M. and Hagan, M., *Neural network toolbox 5 user's guide*, The MathWorks, Inc, U.S., 2007.

[8] Lorigo, L.M. and Govindaraju, V., Off-line Arabic Handwriting Recognition: A Survey, *IEEE Transaction On Pattern Analysis And Machine Intelligenc*, Vol.28, No.5, 2006, pp.712-724

[9] Mahmoud, S.A. and Awaida, S.M., Recognition Of Off-Line Handwritten Arabic (Indian) Numeral Using Multi-scale Features And Support Vector Machines VS. Hidden Markov Models, *The Arabian Journal for Science and Engineerin*,Vol.34, No.2B, pp.429-444.

[10] Omari, S.A., Sumari P., Al-Taweel, S.A. and Husain, J.A., Digital Recognition using Neural Network, *Journal of computer science*, Vol.5, No.6, 2009, pp. 427-434.

[11] Otsu, N., A threshold selection method from gray-level histograms. *IEEE Trans. Sys., Man., Cyber*. Vol.9, 1979, pp. 62–66, (online), available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4310076

[12] Pan, W.M, Bui, T.D. and Suen, C.Y., Isolated Handwritten Farsi numerals Recognition Using Sparse And Over-Complete Representations, ***Center for*** *Pattern Recognition and Machine Intelligence, Concordia University*, IEEE Computer Society, Barcelona, Spain, 2009, pp. 586-590.

[13] Palacios, R., Gupta, A., and Wang, P.S., Handwritten Bank Check Recognition Of Courtesy Amounts, *International Journal of Image and Graphics*, Vol.4, No.2, pp., 2003, 1-20.