Mouse Gestural Hindi Digits Recognition Using Neural Network

By

(MAISA MOHY ALDEEN ABDULA NASER)

Supervised By

Prof. Nidal Shilbayeh

Master Thesis

Submitted in Partial Fulfillment of the Requirements for the Master Degree in Computer Science

Department of Computer Science

Faculty of Information Technology

Middle East University

Amman - Jordan

August, 2010

Middle East University

Authorization Statement

Name: Maisa Mohy Aldeen Abdula Naser.

Signature:

[illegible]

Date: 10-08-2010

Middle East University

Examination Committee Decision

This is to certify that the thesis entitled "Mouse Gestural Hindi Digits Recognition Using Neural Network" was successfully defended and approved on 2010.

Examination Committee Members	**Signature**
Prof. Musbah Mah'd Aqel Department of Computer Science Middle East University	[illegible]
Prof. Nidal F. Shilbayeh Department of Computer Science Middle East University	[illegible]
Dr. Jehad Ahmed Al-Sadi Faculty of Computer Studies Arab Open University	[illegible]

Dedication

I dedicate this thesis to my husband for his great concern and unwavering support, to my parents and my family.

Acknowledgments

By the name of Allah the most Gracious and the most Merciful. “Al Hamdo Lellah”. First, I’m very grateful to go to my advisor, Prof. Nidal Shilbayeh for his infinite support, suggesting, availability, patience and the great ideas about this research topic that helped me to develop this software. Working with him was an honor for me.

And I would like to thank all my teachers and all of people they assisted and supported me to complete my thesis.
Finally, I would like to thank my wonderful husband for supporting and patience. Also my wonderful kids and my parents for every thing they did.

Abstract

An efficient system for recognizing Hindi digits drawn by the mouse is proposed. Our Hindi Digit Recognition using Neural Networks (HDRNN) system is designed and tested successfully. The system deals with recognition the handwritten Hindi digits by using a Multilayer Perceptron (MLP) with one hidden layer. The error backpropagation algorithm has been used to train the MLP network. In addition, an analysis has been carried out to determine the number of hidden nodes to achieve high performance of backpropagation network in the recognition of handwritten Hindi digits.

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 sample test. Experiments showed that this approach is flexible and can achieve high recognition rate for the shapes of the digits represented in this study.

Key words: Digit Recognition, Mouse Gesture, Neural Networks, Backpropagation, Multi layer Perceptron, Feature Extraction.

الملخص

لقد تم بناء نظام فعال للتعرف على الارقام الهندية المرسومة عن طريق الفأرة. هذا النظام صمم عن طريق استخدام طريقة الشبكات العصبونية وقد تم اختباره بنجاح. وقد تم التعرف على الارقام الهندية عن طريق الشكبات العصبونية التي تستخدم خوارزمية ال Backpropagation التي تحتوي على عدة طبقات منها طبقة واحدة مخفية.

بالاضافة الى اجراء تحليل لهذه الشبكة لتحديد عدد الوحدات التي سوف يتم استخدامها في الطبقة المخفية للحصول على اداء عالي لها.

لقد تم استخدام 800 عينة لتدريب النظام و300 عينة للإختبار، هذه العينات تم جمعها من عدة اشخاص بأعمار مختلفة. وقد بينت النتائج بعد الاختبار دقه عالية للنظام في التعرف على الارقام الهندية حيث بلغ اجمالي الدقة لجميع العينات 91%.

نستنتج من ذلك ان نظامنا المصمم مرن ويستطيع ان يحقق نتائج عالية بالنسبة للعينات المستخدمة.

Content

Page No.

List of Tables

Page No.

List of Figures

Page No.

List of Abbreviations

ANN	Artificial Neural Network
ASL	American Sign Language
FCC	Freeman Chain Code
GP	Genetic Programming
HCI	Human-Computer Interaction
HDRNN	Hindi Digits Recognition Using Neural Network
HMM	Hidden Markov Models
LM	Levenberg- Marquardt
Logsig	Logistic Sigmoid
Matlab	Matrix Laboratory
MLP	Multi Layer Perceptron
MSE	Mean Square Error
OCR	Optical Character Recognition
PC	Personal Computer
PCA	Principal Component Analysis
SSE	Sum Square Error
SVM	Support Vector Machines
Tansig	Tan Sigmoid
UI	User Interface

Chapter 1

Introduction

1.1 Introduction

Computer technology has dramatically increased the efficiency of office and business, and we rely on electronics more than ever before. Many of handwritten documents are produced, and how to process this information efficiently has become a bottleneck of office automation and electronic business. Even though handwritings by different people vary, computers are more powerful than ever before, and machine recognition of handwriting has become possible.

Numerous works on handwritten recognition have already been conducted. In such systems, handwritten documents are created by a mouse. Handwritten recognition has emerged as one of the most important research areas based on image processing and recognition. A lot of works were done by depending on computer in order to reduce the processing time and provide more accurate results. For example, depending on different types of data, such as characters and digits, in addition to the numbers those are used frequently in normal life operation. These different types of data are used in order to automate systems that deal with numbers such as postal code, banking account numbers and numbers on car plates.

Hindi digit recognition is a system that consists of ten Hindi numbers; starting from (0) through (9). It is a complicated classification problem because each person writes the digits differently and the numbers given to the system handwritten in a different styles, sizes and possible orientations.

In this thesis, we focus on developing a system that recognizes the Hindi digits using mouse device as the input device for the digits Table 1.1 shows the Hindi digits.

9	8	7	6	5	4	3	2	1	0	**Arabic Digits**
٩	٨	٧	٦	٥	٤	٣	٢	١	٠	**Hindi digits**

Table 1.1 Hindi digits and Arabic digits

The graphical devices such as a mouse can be used to provide simple input to a mouse gesture recognizer.

Recently, Artificial Neural Network (ANN) has become more popular as a technique to perform handwritten recognition because it has the ability to produce high recognition accuracy. And Neural Networks (NNs) are capable of providing good recognition in the presence of noise that other methods normally fail at. Although Neural Networks have been used for digit recognition for many years, most of the work has been developed to Arabic digits recognition. Up to the time being, a very little work has been reported for Handwritten Hindi digits recognition.

Indian numerals are used in Arabic writing, while Arabic numerals are used in Latin languages. The term 'Hindi digits' is used here to refer to the numerals used in Arabic writing (Mahmoud & Awaida, 2009).

There are two kinds of input for handwritten recognition: off-line and on-line. Offline handwritten recognition takes a raster image from a scanner, digital camera or other digital input source. The image is binarized using a threshold technique, either it is colored or gray-scaled, so that the image pixels are either on (1) or off (0). The rest of the preprocessing is similar to the on-line version with two key differences: Off-line processing happens after the writing of characters is complete and the scanned image is preprocessed.

Secondly, off-line inputs have no temporal information associated with the image. The system is not able to infer any relationships between pixels or the order in which strokes were created. Its knowledge is limited to whether a given pixel is on or off (Senior, 1992), (Lorigo & Venu, 2006).

On-line character recognition accepts (x, y) coordinate pairs from an electronic pen touching a pressure-sensitive digital tablet. On-line processing happens in the real-time while the process of writing is taking place. Also, relationships between pixels and strokes are supplied due to the implicit sequencing of on-line systems that can assist in the recognition task (Klassen, 2001).

In our thesis, we use the off-line input for digit recognition using the mouse. The numbers will be given to the system handwritten in different styles and sizes. A system for recognizing isolated digits can be used as an approach for dealing with an application. In other words, a new system can be used to let the computer understand

the Hindi numbers that are written manually by users and view them according to the computer process.
The proposed system recognizes isolated Hindi digits as the system acquires an image that consists of digits. The image will be processed into several phases such as binarization, resizing the image as well as skeletonization before recognizing the digit. A multilayer neural network will be used for the recognition phase; a feed forward back propagation algorithm will be applied for training the network.

1.2 Statement of the Problem

On the basis of the problem that kids and most people preferred to use a mouse instead of a keyboard, we designed a user friendly and simple system that can be user friendly and simple when used. In building a new mouse handwritten recognition system for Hindi numerals we developed a system to overcome with the following features:

1. The idea of using neural networks for our purpose of recognizing handwritten Hindi digits a new one.
2. Can match the pattern recognition for an isolated Hindi Numerals.
3. Have a high degree of input recognition.
4. Able to recognize the Hindi digits as quickly as possible.
5. There are 10 different digits, so the classifier has 10 output classes. It is a complicated classification problem because each person writes the digits differently.
6. It is important to choose good features to be able to classify the different digits correctly and train the classifier with a large amount of digits.
7. From the different techniques to classify, a multilayer neural network with backpropagation algorithm is chosen for the handwritten digits recognition.
8. An infinite variety of writing styles are produced by different writers.

1.3 Thesis Objectives

The following are the objectives of this thesis:

1. To recognize and classify a handwritten Hindi digits number. We also aim to implement a system which can match the pattern recognition for an isolated Hindi digit.
2. To let computer system understand different users handwriting styles and recognize them as the standard writing.

3. To let the computer understand the Hindi digits which are written manually by users and view them according to the computer process.
4. To train the network data by using feed forward back propagation algorithm.
5. To test the system efficiency achieving high recognition rate.

1.4 Thesis Significance

The developed Hindi Digits Recognition, which uses the Neural Network system, can provide a lot of features that can help the users do the following:

1. Use the environment of the computer system more easily.
2. Control all windows of the operation system.
3. Provide a quick access to common functions of a computer application.
4. Allow users to use the mouse to do more than pointing and clicking, so the mouse can be used to start and end a program, even to shutdown the operation system, move forward and backward in browsers such as internet explorer browser.
5. Try to use the mouse instead of keyboard.
6. To let the computer systems understand different handwriting styles and recognize them as a standard writing system.

1.5 Thesis Organization

In addition to this chapter, the thesis includes four other chapters:
Chapter 2 provides an overview of the handwritten recognition along with listing and explaining different related works in the area of the developed system. Chapter 3 explains in detail the proposed system architecture and the different models and algorithms that are used in all parts of the proposed system it also presents the user interface in the details of all options that represent activities and tasks in the proposed system as well. Chapter 4 represents a complete experimental result.
Finally, conclusion and future works, including recommendations, are presented in chapter 5.

Chapter 2
Literature Review

2.1 Artificial Neural Network

An Artificial Neural Network (ANN), also called "Neural Network" (NN), is a mathematical model or computational model that tries to simulate the structure and/or functional aspects of biological neural networks.

ANN is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. NNs, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons as shown in Figure 2.1. This is true of NNs as shown in Figure 2.2.

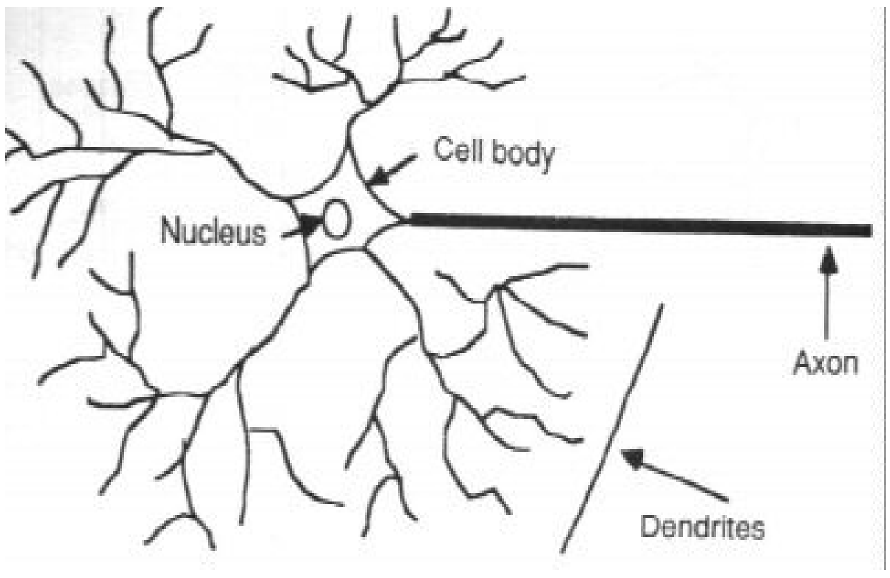


Figure 2.1: Biological Neural Network (Kröse & Smagt, 1996)

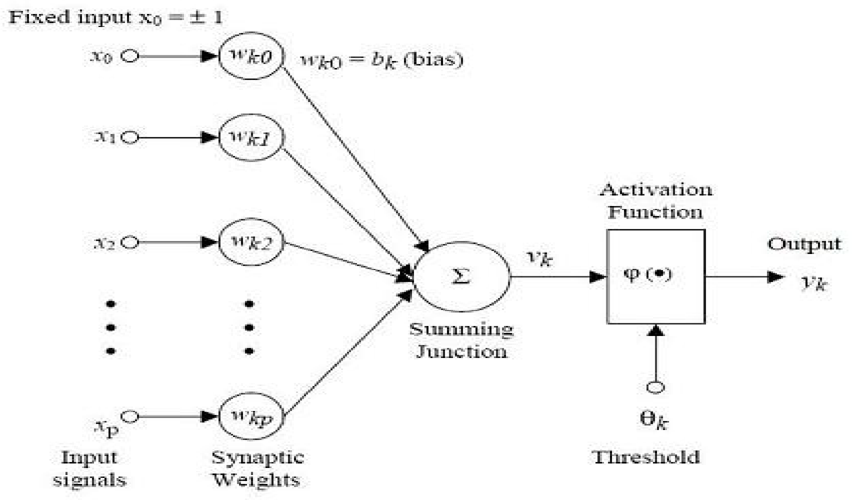


Figure 2.2: Artificial Neural Network (Ben Kröse & Smagt, 1996)

Historical background

ANN simulations appear to be a recent development. However, this field was established before the advent of computers, and has survived at least one major setback and several eras (Stergiou, Siganos, 1996).

Neural networks versus conventional computers

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

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

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

Neural networks and conventional algorithmic computers are not in competition, but rather complete each other. There tasks are more suited to an algorithmic approach like arithmetic operations and tasks that are more suited to neural networks. Even more, a large number of tasks require systems that use a combination of the two approaches (normally a conventional computer is used to supervise the neural network) in order to perform at the maximum efficiency bossible (Stergiou, Siganos, 1996).

Types of NNs:

- Single Layer Perceptrons
- Multilayer Perceptrons (MLPs)
- Radial-Basis Function Networks (RBFs)
- Hopfield Networks
- Boltzmann Machines
- Self-Organization Maps (SOMs)
- Modular Networks (Committee Machines)
- Support Vector Machines
- Bayesian Networks
- Probabilistic Graphical Models
- Hidden Markov Models

Advantages of NNs

The reason that many researchers preferred to use neural network model for solving a given problem is related to many advantages:

- Pattern recognition is a powerful technique for harnessing the information in the data and generalizing about it. Neural nets learn to recognize the patterns which exist in the data set.
- The system is developed through learning rather than programming. Programming is much more time consuming for the analyst and requires the analyst to specify the exact behavior of the model. Neural Networks teach themselves the patterns in the data freeing the analyst for a more interesting work.

- NNs are flexible in a changing environment. Rule based systems or programmed systems are limited to the situation for which they were designed--when conditions change, they are no longer valid. Although neural networks may take some time to learn a sudden drastic change, they are excellent at adapting to constantly changing information.
- NNs can build informative models where more conventional approaches fail. Because neural networks can handle very complex interactions, they can easily model data which is too difficult to model with traditional approaches, such as inferential statistics or programming logic.
- The performance of neural networks is at least as good as classical statistical modeling, and better on most problems. The neural networks build models that are more reflective of the structure of the data in significantly less time.
- NNs now operate well with modest computer hardware. Although neural networks are computationally intensive, the routines have been optimized to the point that they can now run in reasonable time on personal computers. They do not require supercomputers as they did in the early days of neural network research.
- Speed: The using of NNs is very fast especially when it is implemented as a hardware structure, but it may be very slow when implemented on a serial computer. Smaller NNs are very fast to work which make them suitable to work with many applications like pattern recognition.
- Stability of a solution: When the system performs training of the NNs, you can see that the error is decreasing while learning a set of input data and is increasing for the test data. Hence, this property is very suitable for pattern recognition.
- Multitasking**:** Same NNs can be used in different classes of tasks.

But there are some limitations that should be understood such as:

- It is difficult to extract rules from neural networks. This is sometimes important to people who have to explain their answers to others and to people who have been involved with artificial intelligence, particularly expert systems which are rule-based.
- As with most analytical methods, you cannot just throw data at a neural net and get a good answer. You have to spend time understanding the problem or the

outcome you are trying to predict. You must also be sure that the data used to train the system is appropriate and is measured in a way that reflects the behavior of the factors. If the data is not representative of the problem, neural computing wills not product good results.

- Finally, it can take time to train a model from a very complex data set. Neural techniques are computer intensive and will be slow on low end PCs or machines without math coprocessors. It is important to remember though that the overall time given find out to results can still be faster than other data analysis approaches, even when the system requires extra time for training.

Pattern recognition is one of the most important abilities of human beings. Relying on this ability, human beings extract useful information about their surroundings. Today, as the digital computer technology has been largely used and developed to simulate this unique, the stimulation of this unique human ability with automated machines is becoming more realistic (Mike, 2006).

Pattern recognition is a well-established field of study and digit recognition that has been long seen as one of its important contributions.

Simply, digit recognition includes the following three procedures:

1 Digit acquisition and preprocessing.

2 Feature extraction that gets the most useful information from the pool of input data and get rid of the irrelevant information.

3 Classification of the extracted features with specially designed algorithms those are usually task-oriented or general-purpose-based in some cases.

The automatic recognition of printed and written text was one of the first goals of early research in digit recognition. Optical Character Recognition (OCR) was a very successful technique that was based on statistical pattern classification methods. OCR technique has been widely used for printed text recognition and achieved recognition rates higher than 90%. Because of the high degree of variation in shape and size of handwritten, OCR is completely not suitable for handwriting recognition. Later, Artificial Neural Network was proposed to solve this tough problem.

Handwriting recognition techniques can be classified into two major groups, the online and offline recognition. Online recognition processes the data input such as handwritten characters, digits and signatures with special devices, pen or mouse that are attached directly to the system. Offline processing happens after the writing of characters is complete and the scanned image is preprocessed.

2.2 Image Preprocessing

Preprocessing is an important stage which removes and reduces the noise from the images. The most common method used is a skeleton to simplify feature extraction phase. Skeletonization is a morphological operation that is used to remove selected foreground pixels from binary images. It is normally applied to binary images, and produces another binary image as output.

Many techniques used to convert original image into skeleton image. And it removes pixels of objects but does not allow objects to break apart. Hence the information, such as grayscale values and thickness of the digits, actually assist in providing more information.

Areas:

- Handwritten and printed characters
- Fingerprint patterns
- Chromosomes & biological cell structures
- Circuit diagrams
- Engineering drawings

Skeleton representation capabilities:

- Reducing the amount of data required to be processed.
- Reducing the time required to be processed.
- Extraction of critical features such as end-points, junction-points, and connection among the components.
- The vectorization algorithms often used in pattern recognition tasks also require one-pixel-wide lines as input.
- Shape analysis can be more easily made on line like patterns.
- Good way to represent the structural relations between components in the pattern.
- Wide-range used for digit, word, signature and handprint recognition systems (Al-Rashaideh, 2006).

Many researchers have been used skeletonization in their preprocessing stage like Al-Rashaideh, (2006), Choudhari, (2006) to increase the quality of hand printed data. This means more precisely that the character is transformed to such that they are more similar to mean the class they belong to. In preprocessing, the preliminary steps include normalization, digitization and thinning.

In digitization, object is converted into binary form by binarization method. Object is separated from background (1-represent as region, 0 represent as no region). This binarized image is put through preprocessing routines that smooth the image and eliminate noise, artificial holes and other artifacts produced by the digitizing process.

Jain & Ko, (2008) undertook the approach of processing the training set images to reduce the data by thresholding the given image to a binary image. They also looked at various processing methods such as edge-detection, and thinning of the digit image to get a skeleton of the digit. This approach of acquiring the skeleton of the digit is widely used in the classifiers which mainly reply upon a well-defined input image for their accuracy.

Al-Omari, et al., (2009) used thinning and skeletonization for **r**epresenting the shape of the object in a relatively smaller number of pixels.

Thinning algorithms can be parallel or sequential. The parallel algorithm is applied on all pixels simultaneously. On the other hand, the sequential algorithm examines pixels and transforms them depending on the preceding processed results.

2.3 Feature Extraction

The feature extraction is the feature classification process. Usually, the digit recognition problem is considered as a classification problem. So that it is simple to benchmark different recognition methods by testing their respective classifiers or algorithms.

Features should contain information required to distinguish between classes, be insensitive to irrelevant variability in the input, and also be limited in number to permit efficient computation of discriminate functions and to limit the amount of training data required.

Feature extraction plays the most important role in character recognition Choudhari, (2006). It will extract the well-defined characteristics, which classify the character in classification stage. The features are extracted by projection methods from original and skeleton image. They also extracted special points from a thinned image.

Jain & Ko, (2008) used the Principal Component Analysis (PCA) it is a fundamental multivariate data analysis method which is encountered into a variety of areas in neural networks, signal processing, and machine learning. It is an unsupervised method for reducing the dimensionality of the existing data set and extracting important information. PCA does not use any output information; the criterion to be maximized is the variance.

Jain & Zongker, (1997) used multidimensional scaling for feature extraction which is a well-known technique to obtain an appropriate representation of the patterns from the given proximity matrix.

The feeding data to the recognition systems are usually a pool of irrelevant information containing many features. Features can be symbolic or numerical matter that bears useful or irrelevant information about the recognition tasks. The most important thing to do for the recognition systems is to be able to extract the relevant features from the input data. It is easy to understand that it will be impossible to fulfill the recognition task if too many features are presented, if they are not all relevant. So, how to effectively and accurately extract relevant features form the feeding is the first step to do for handwriting recognition to be practical. To be relevant for the extracted features is not enough to accomplish the tasks though. Feature extraction is a procedure similar to the judgment flow by human beings. The practical judgment criteria can be variant depends on the task itself. But, in general, the features should be relevant that they can provide continuous information flow for the following recognition procedures.

2.4 Recognition Algorithms

Offline recognition systems process is the input data that has been attained with optical scanner from paper documents or an image form digital camera. Compared to online recognition, offline recognition could be more difficult in that much useful information can be lost such as the handwriting duration, writing sequence, number of strokes and writing direction. For this reason, successful offline recognition has a significant meaning to the people in the area to deal with static and existing handwritten matter. Several techniques have been used for offline recognition; such as statistical, structural and neural network approaches. It has been reported that neural networks have achieved good results in handwritten character recognition due to their ability to learn and generalize. Neural network approaches have been applied for online and offline segmented character and numeral recognition, and online and offline cursive words recognition.

Dibeklioglu, (2007) used two different types of MLP network are trained for recognition task. MLP networks have 10 output units and 35 (7x5 binary image), and 100 (10x10 binary image) input units, respectively, due to different re-sampling rates. Hidden unit counts are determined by applying ten-fold cross-validation.

After determining hidden unit counts, each MLP network is trained again by using whole training data and tested on test data.

Al-Omari, et al., (2009) experimented two networks, one with two layers and another with three layers. Both networks have been trained on the same training data set using feed forward back propagation algorithm. They used different ordering for the data set in the training process. The same test data set has been used for testing both networks.

Mashor, & Sulaiman, (2001) said, the recognition of handwritten letters is a very complex problem. Since the letters could be written in different size, orientation, thickness, format and dimension. These will give infinity variations. The capability of neural network to generalize and be insensitive to the missing data would be very beneficial in recognizing handwritten letters. So in their study, they explore the capability of neural network to recognize noisy numerals. Noisy numeral or character is the common problem in vehicle registration plate recognition where the characters captured by the camera may be noisy due to dirt, rain, reflection and so on. Multi Layered Perceptron (MLP) network trained using Levenberg-Marquardt algorithm has been used to recognize noisy number 0 to 9.

Some methods based on neural networks require long time to learn. Once the learning procedures are accomplished, their response to the feeding is extremely fast and reliable. Even though, fast response and high reliability are important considerations for automated recognition systems, long learning or training procedure and requirements of large training samples are critical for building practical recognition systems.

2.5 Backpropagation Network

Back-propagation algorithm is designed for multilayer perceptron (MLP) to fix and compute its parameters so as to minimize an appropriate cost function of its output.

The training algorithm of back propagation involves four stages:

1. Initialization of weights (some small random values)
2. Feed forward
3. Back propagation of errors (target – output)
4. Update the weights and biases.

As soon as you have trained net, you can test it. Select the digit (or test all of them).

LeCun, et al., (1989) reported their results on handwritten zip code recognition by using backpropagation network. The backpropagation network is multilayered (input layer, hidden layer and output layer) and arranged in a feedforward architecture with the ability to send back the error between the actual output and the target output values. Each layer contains interconnected elements that behave like soft linear classifiers. Each element computes a weighted sum of the inputs and transforms the sum through a nonlinear squashing function. The learning is fulfilled by iterations of modifying the weights on each connection so as to minimize an objective function that was popularly the mean square error between the actual output and the desired output. Backpropagation was used to calculate the gradient of the objective function.

Their network was trained for 23 passes through the training set. The performance of their backpropagation network was measured both on the training and test set. In order to achieve 1% error rate on the non-rejected data for the test set, a rejection rate up to 12.1% was required. If the reject rate was counted to the overall error rate, the performance of their algorithm would be far from the amazing 1% error rate. Most misclassifications were thought to be due to erroneous segmentation of the images.

The authors believed other misclassifications were caused by ambiguous patterns, lower solution effects or writing styles not present in the training set.

They also reported another result by the same algorithm on the same data set that was supplemented with 3349 printed digits in 1990. The error rate was 1% for 9% reject rate. So the error rate for the whole data set should be around 10%, which was improved by about 3% compared to the previous result. Considering the fact that 3349 printed digits had been added to the same sample as the one used previously, the improvement is barely contributed by the printed digits since the author found no error were made on the printed characters. This time, the learning was finished after 20 passes through the training set which contained 2549 printed digits. Even though the authors thought that the learning time for their network was short due to the redundant nature and constraints imposed on the network, the training procedure was necessary.

Mashor, & Sulaiman, (2001) reviewed that the recognition performance of the MLP network will highly depend on the structure of the network and training algorithm. In their study, Levenberg- Marquardt (LM) algorithm has been selected to train the network. They found that the algorithm has much better learning rate than the famous back propagation algorithm.

2.6 Related work

Handwriting is one of a challenging problem in the field of computer and the approaches Neural Networks have become a popular technique for digits recognition. Many attempts have been made to achieve successful recognition of handwritten digits.

Knerr, et al., (1992) show that neural network classifiers with single-layer training can be applied efficiently to complex real-world classification problems such as the recognition of handwritten digits. And introduce the STEPNET procedure, which decomposes the problem into simpler sub-problems which can be solved by linear separators. Provided appropriate data representations and learning rules are used, performances which are comparable to those obtained by more complex networks can be achieved. They present results from two different data bases: a European data base comprising 8,700 isolated digits and a zip code data base from the U.S. Postal Service comprising 9,000 segmented digits.

Klassen, (2001) introduced a novel Arabic letter recognition system that adapted to the demands of hand-held and digital tablet applications. Their system uses neural networks for feature extraction and classification. Linear networks are employed as classifiers because of the low computational overhead during training and recall.

Al˙Imo˘Glu & Alpaydin (2001) investigated techniques to combine multiple representations of a handwritten digit to increase classification accuracy without significantly increasing system complexity or recognition time. In pen-based recognition, the input is the dynamic movement of the pen-tip over the pressure sensitive tablet. There is also the image formed as a result. On a real-world database of handwritten digits containing more than 11,000 handwritten digits, they notice that the two multi-layer perceptron (MLP) based classifiers using these representations make errors on different patterns implying that a suitable combination of the two would lead to higher accuracy. They implemented and compared voting, mixture of experts, stacking and cascading. Combining the two MLP classifiers they indeed get higher accuracy because the two classifiers/representations fail on different patterns. They especially advocate multistage cascading scheme where the second costlier image-based classifier is employed only in a small percentage of cases.

Teredesai, et al., (2002) described a classification method for on-line handwritten digits based on off-line image representations. The goal was to use image-based features to improve classifier accuracy for on-line handwritten input. And they described an initial framework that can be used to achieve this goal. This framework for handwritten digit classification is based on genetic programming (GP). Several issues in pre-processing, transformation of data from on-line to off-line domains and feature extraction are described.

Al-Taani, (2008) proposed an efficient structural approach for recognizing on-line handwritten digits. After reading the digit from the user, the coordinates (x, y) of the pixels representing the drawn digit are used for calculating and normalizing slope values of these coordinates. Successive slope values are then used to record the change of direction used to estimate the slope. Based on the changing of signs of the slope values, the primitives are identified and extracted. These primitives represent a specific string which is a production of a certain grammar. Each digit can be described by a specific string. In order to identify the digit they determine which grammar the string belongs. A Finite Transition Network which contains the grammars of the digits is used to match the primitives' string with the corresponding digit to identify the digit.
The proposed method was tested on a sample of 3000 digits written by 100 different persons; each person wrote the 10 digits three times each. The method achieved accuracy of about 95% on the sample test.

Al-Rashaideh, (2006) reviewed the importance of the pattern classification and its application. They also listed the characteristics of the Arabic language's writing style, furthermore focused on the preprocessing step of the recognition system. Then described and tested algorithm to create skeleton which will be the base representation of Arabic words which that used for feature extraction phase. Also, they discussed and implemented the algorithm of baseline detection. The algorithms of skeleton and baseline detection were tested using database IFN/ENIT of handwritten.

Dibeklioglu, (2007) implemented a system for on-line digit recognition with the feature set consisting of the binary image stream and the sequence. Best results for MLP recognition task is obtained at the highest re-sampling rate 100 (10x10 binary image) as 70.5%. Overall result for the previously developed and trained HMM recognition system was obtained as 62%. Best result for digit recognition was obtained with HMM-MLP hybrid system as 77.9%. As it is seen hybrid method increases the success ratio significantly. Because MLP and HMM sub-systems uses different features of the handwritten digits data. This study may be extended by adding extra features (global features) to MLP networks like number of strokes in the digits. While global features provide information about specific cases concerning the structure of the digit, spatial features are intended to provide overall digit appearance information.

Jain et al., (2008) implemented a classification algorithm to recognize handwritten digits (0-9). It has been shown in pattern recognition that no single classifier performs the best for all pattern classification problems consistently. Hence, the scope of the project also included the elementary study the different classifiers and combination methods, and evaluate the caveats around their performance in this particular problem of handwritten digit recognition. This report presents their implementation of the Principal Component Analysis (PCA) combined with 1-Nearest Neighbor to recognize the numeral digits, and discusses the other different classification patterns. They were able to achieve an accuracy rate of 78.4%.

Shilbayeh, et al., (2009) proposed an efficient structural approach for recognizing Hindi digits drawn by mouse. Their system (MGHD) was designed and tested successfully. Their system deals with representation of shape based on a new boundary (FCC) with eight connectivity and then uses template to recognize the Hindi digit. In their work FCC has been modified to extract the boundary of the shape and specify an area of the object were it has been drawn in the image as first check, then matching the result of the recognizer with the result of matching templates as a second check for improving accuracy of the recognition. The proposed method was tested on a sample of 1350 digits written by 27 different writers selected from different ages, genders and jobs each one wrote 10 digits 5 times. And experimental result shows high accuracy of about 89% on the sample test.

Al-Omari et al., (2009) the main objective for their system was to recognize isolated Arabic digits exist in different applications. They presented a system for dealing with such problem. The system started by acquiring an image containing digits, this image was digitized using some optical devices and after applying some enhancements and modifications to the digits within the image it can 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 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.

Ahangar & Ahangar, (2009) they have made an attempt to recognize handwritten Farsi characters by using a multilayer perceptron with one hidden layer. The error backpropagation algorithm has been used to train the MLP network. In addition, an analysis has been carried out to determine the number of hidden nodes to achieve high performance of backpropagation network in the recognition of handwritten Farsi characters. The system has been trained using several different forms of handwriting provided by both male and female participants of different age groups. In this work, the experiments were carried out on two hundred fifty samples of five writers. The results showed that the MLP networks trained by the error backpropagation algorithm are superior in recognition accuracy and memory usage. The result indicates that the backpropagation network provides good recognition accuracy of more than 80% of handwritten Farsi characters.

Mahmoud, & Awaida, (2009) described a technique for automatic recognition of off-line handwritten Indian numerals using Support Vector Machines (SVM) and Hidden Markov Models (HMM). Local, intermediate, and large scale features are used. SVM parameters, producing the highest recognition rates, are experimentally found by using an exhaustive search algorithm. In addition, SVM classifier results are compared to those of the HMM classifier.
The researchers use 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. Other divisions of data for training and testing were performed and resulted in comparable performance. The achieved average recognition

rates were 99.83% and 99.00%, using the SVM and HMM classifiers. SVM recognition rates proved to be better for all digits. Comparison at the writer's level (Writers 34 to 44) showed that SVM results outperformed HMM results for all tested writers. The classification errors of the SVM classifier were analyzed.

Chapter 3
Hindi Digits Recognition using Neural Network

3.1 The System Framework

This section presents a general framework for the proposed system which is divided into the following stages:-

- **Input Hindi Digit**

  The user draws a Hindi digit inside the special window, and then it is saved 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.

The framework of Hindi digit recognition system is presented as processing stages and shown in Figure 3.1.

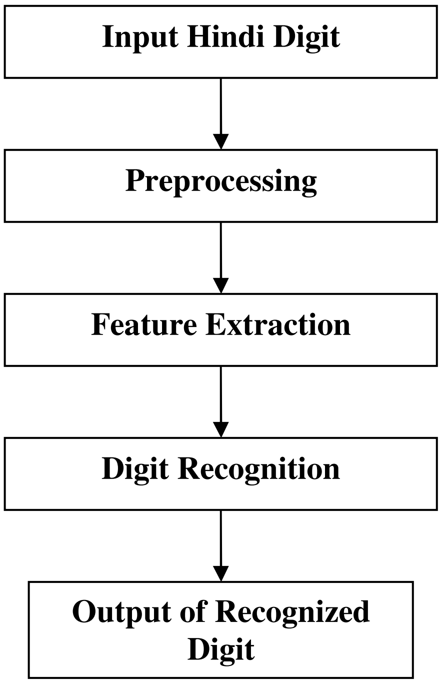


Figure 3.1: The System framework

3.2 Input the Hindi Digit Stage

The interface is designed using C#.net. The user can draw the Hindi digit inside (160x200 pixels) window with any style, size, and possible orientation, without limitation of the starting point as shown in Table 3.1.

Number	Description	Number	Description
	Sefr_0		**Number Khamsa_5**
	Number wahad_1		**Number Seta_6**
	Number Ethnan_2		**Number Sabaa_7**
	Number Thalath_3		**Number Thamanya_8**
	Number Arbaa_4		**Number Tessa_9**

Table 3.1: Hindi digits drawn by the mouse

To start drawing a digit, the user presses the left mouse button as shown in Figure 3.2, then he saves the image in a database folder (image sample) as .jpg image, and can draw another Hindi digit if he wants.

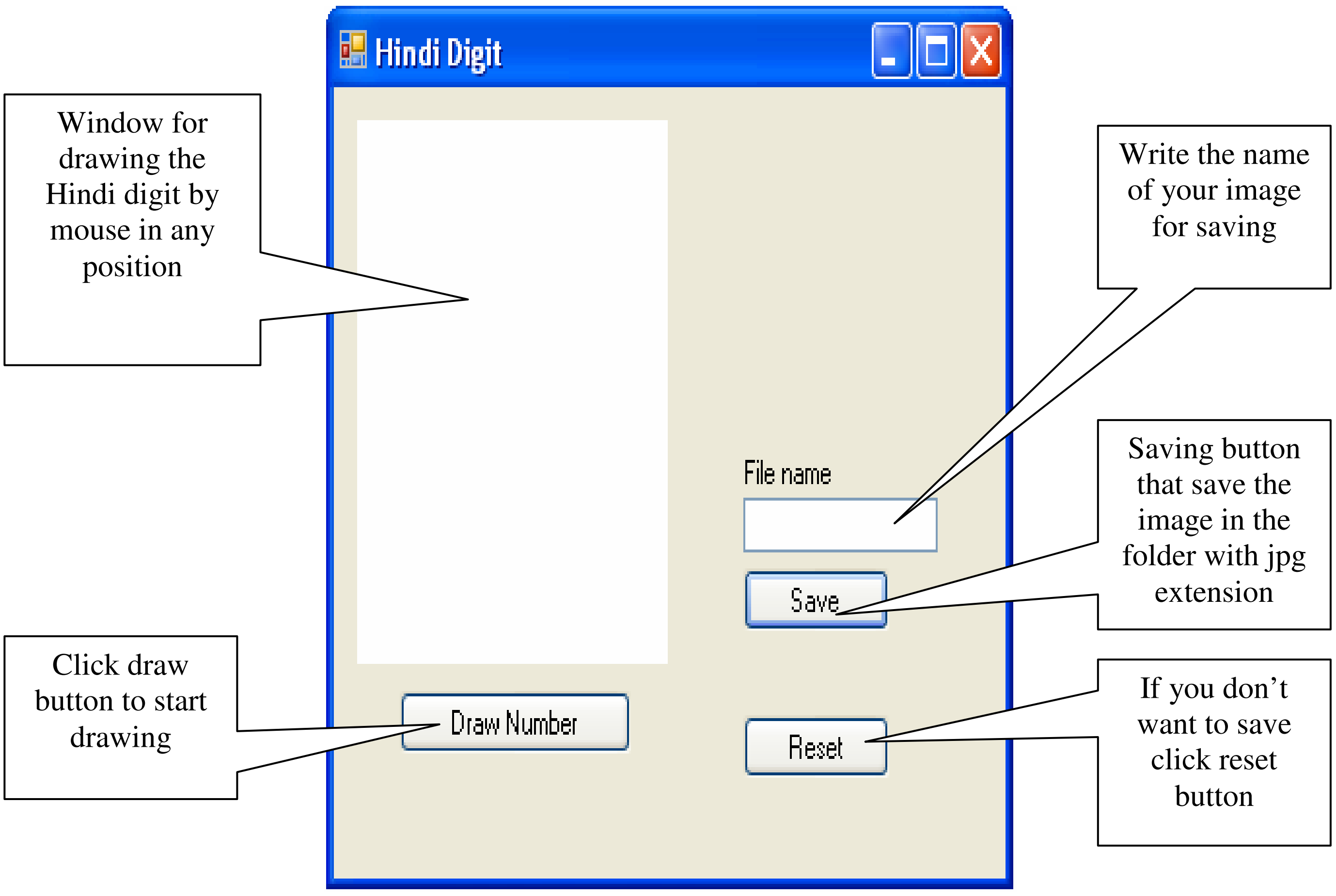

Hindi Digit
Window for drawing the Hindi digit by mouse in any position
Write the name of your image for saving
File name
Saving button that save the image in the folder with jpg extension
Save
Click draw button to start drawing
Draw Number
Reset
If you don't want to save click reset button

Figure 3.2: C# Graphical User Interface for drawing Hindi digit

Jpeg extension is used here because it is the right format for images which must be very small files.

And Figure 3.3 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.

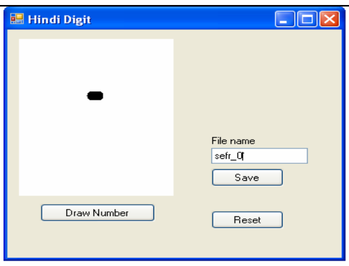
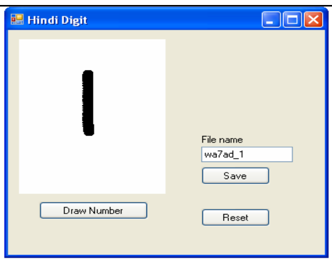
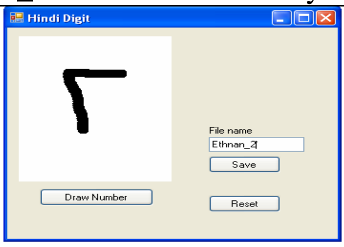
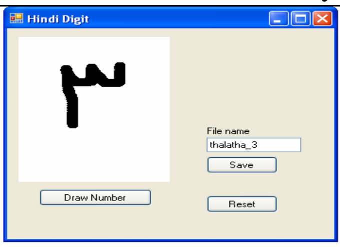
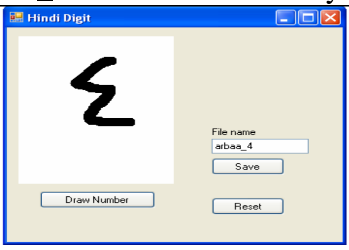
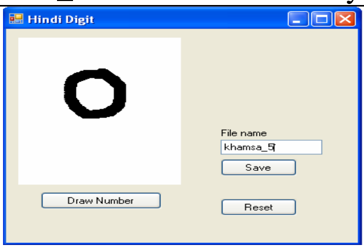
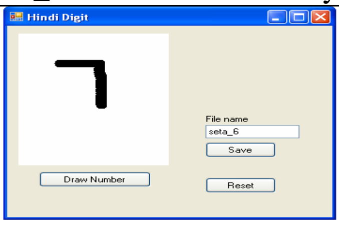
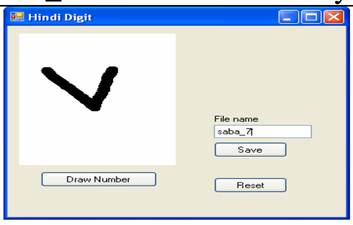
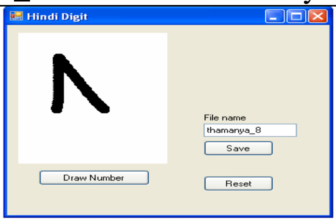
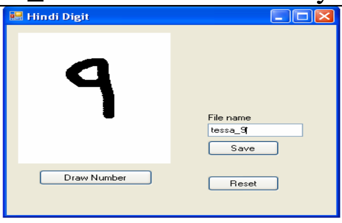
C# Hindi digit GUI	
 File name: sefr_0 Sefr_0 tested at HDRNN system	 File name: wa7ad_1 Wahad_1 tested at HDRNN system
 File name: Ethnan_2 Ethnan_2 tested at HDRNN system	 File name: thalatha_3 Thalatha_3 tested at HDRNN system
 File name: arbaa_4 Arabaa_4 tested at HDRNN system	 File name: khamsa_5 Khamsa_5 tested at HDRNN system
 File name: seta_6 Seta_6 tested at HDRNN system	 File name: saba_7 Sabaa_7 tested at HDRNN system
 File name: thamanya_8 Thamaanya_8 tested at HDRNN system	 File name: tessa_9 Tesaa_9 tested at HDRNN system

Figure 3.3: (0-9) Digits drawn by mouse in C# System Interface

3.3 Preprocessing Stage

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 160x200 pixels to 30x30 pixels, inversing the binary image and finally skeletonizing.

The algorithm of preprocessing steps is shown in Figure 3. 4.

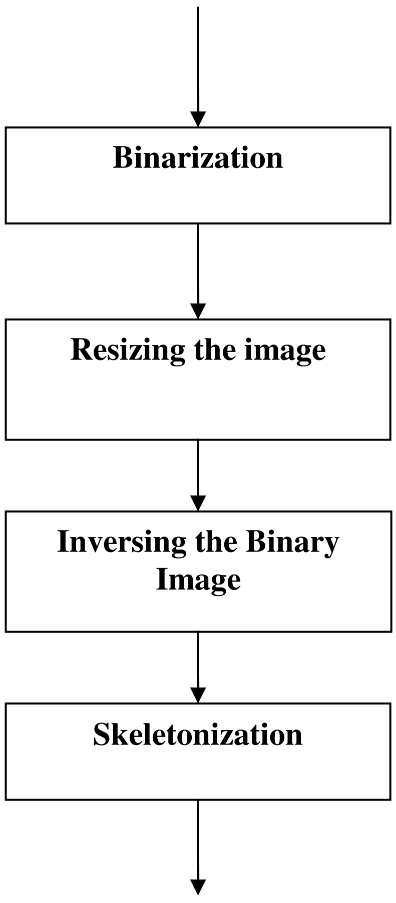


Figure 3.4: Preprocessing Steps

3.3.1 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. The algorithm of the binarizing image is shown in Figure 3.6.

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 3.1.

$$P(x,y) = \begin{cases} 0 : P(x,y) < \text{threshold value.} \\ 1 : P(x,y) \geq \text{threshold value.} \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \; 3.1$$

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 Figure 3.5**.**

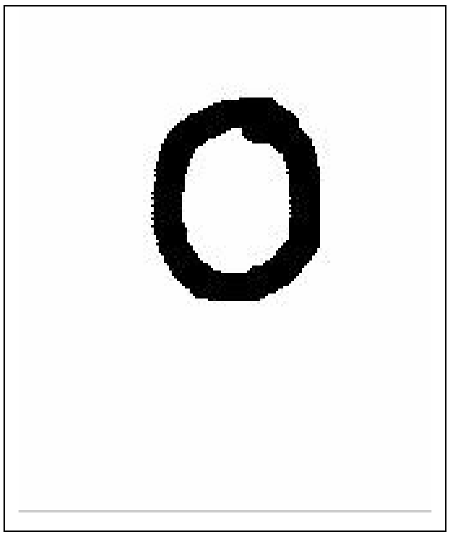


Figure 3.5 Binary Image

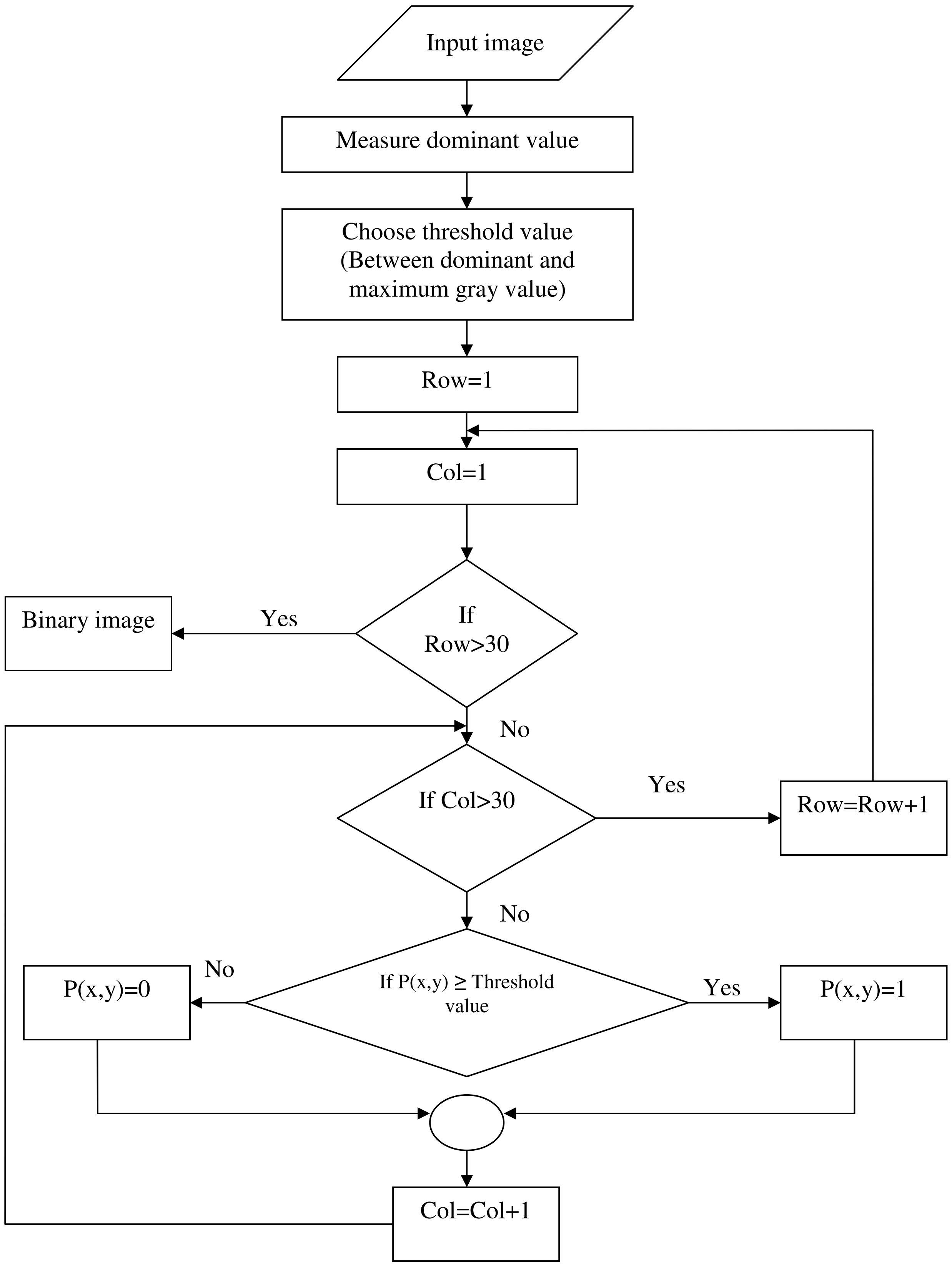
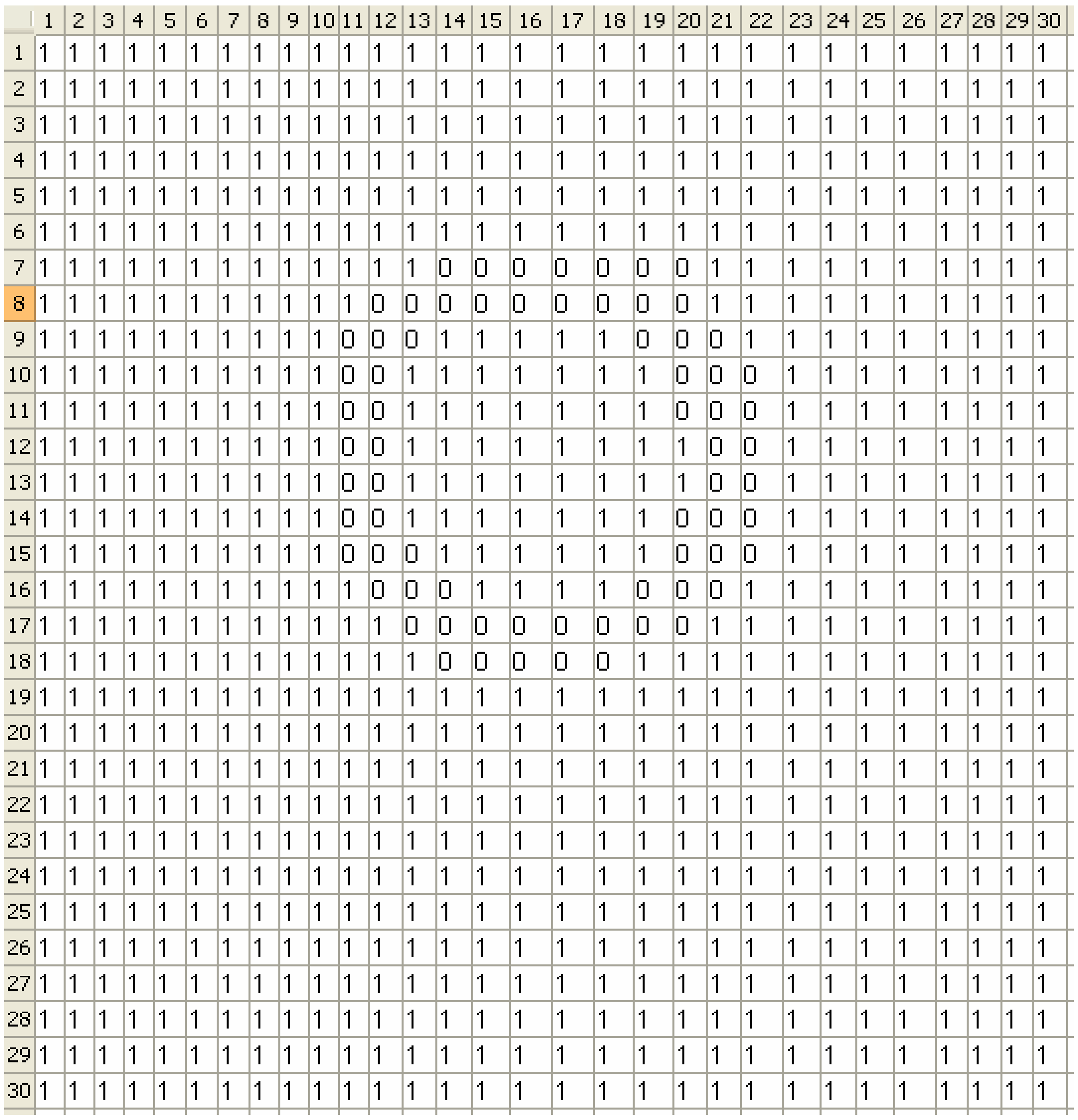


Figure 3.6: Binarization Flowchart

3.3.2 Resizing the Image Step

The images are normalized to a fixed size using the function image resize (imresize (bw, [30 30]), where bw is the image which convert to binary. So the final size is 30x30 pixels. As we see in Figure 3.7, the background of the image takes the value 1, while the handwritten numbers take the value 0.



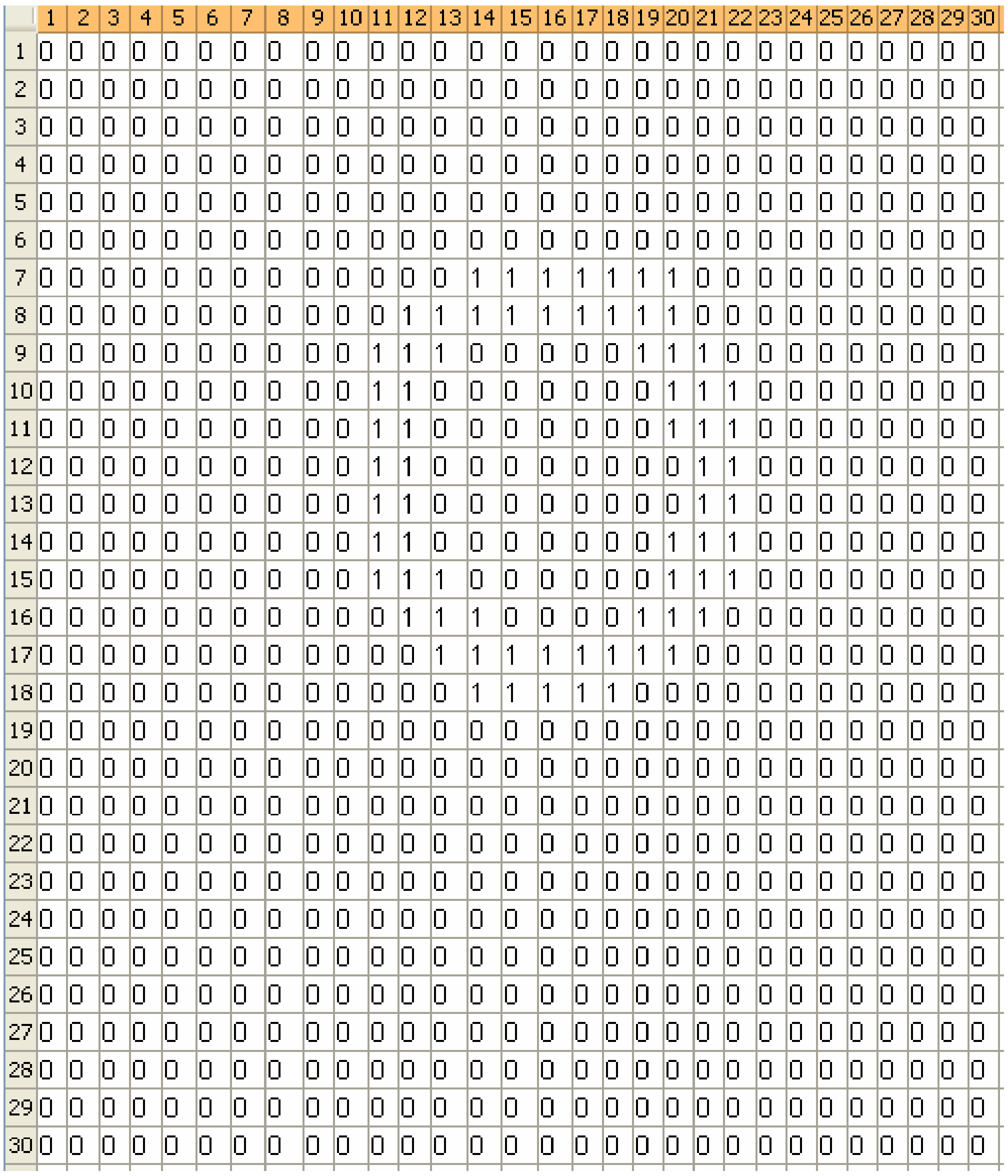
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1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

Figure 3.7: Number Khamsa_5 after binarization and resizing

3.3.3 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 3.8 shows the inverse of the binary images.



	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	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	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	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	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	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	0
7	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	0	0
8	0	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	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	0
11	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	0
12	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	1	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	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	1	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
16	0	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	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	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	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	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	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	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	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	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	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	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	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	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	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	0

Figure 3.8: Number Khamsa_5 after inversing the binary image

After inversing the binary image, the background takes the 0 value, while the handwritten number takes 1 value, as shown in Figure 3.9.

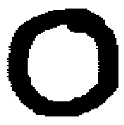
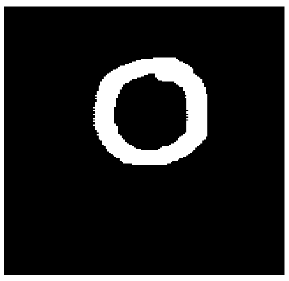
 Number Khamsa_5 before inverse	 Number Khamsa_5 after inverse

Figure 3.9: Binary images before and after inversing

3.3.4 Skeletonization Step

While the preprocessing phase converts the form of 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.

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 Figure 3.10. This approach of acquiring the skeleton of the digit is widely used in the classifiers which mainly reply upon a well-defined input image for their accuracy.

We used the function bwmorph(BW,'thin',Inf) for skeletonizing the digits. Figure 3.11 shows the logical view of the image after skeletonization.

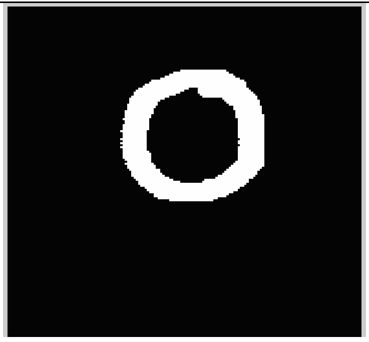
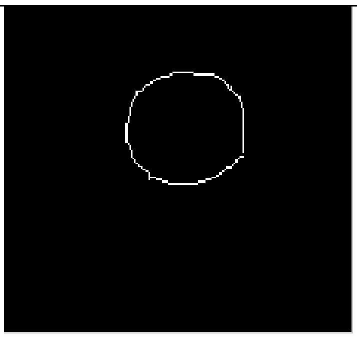
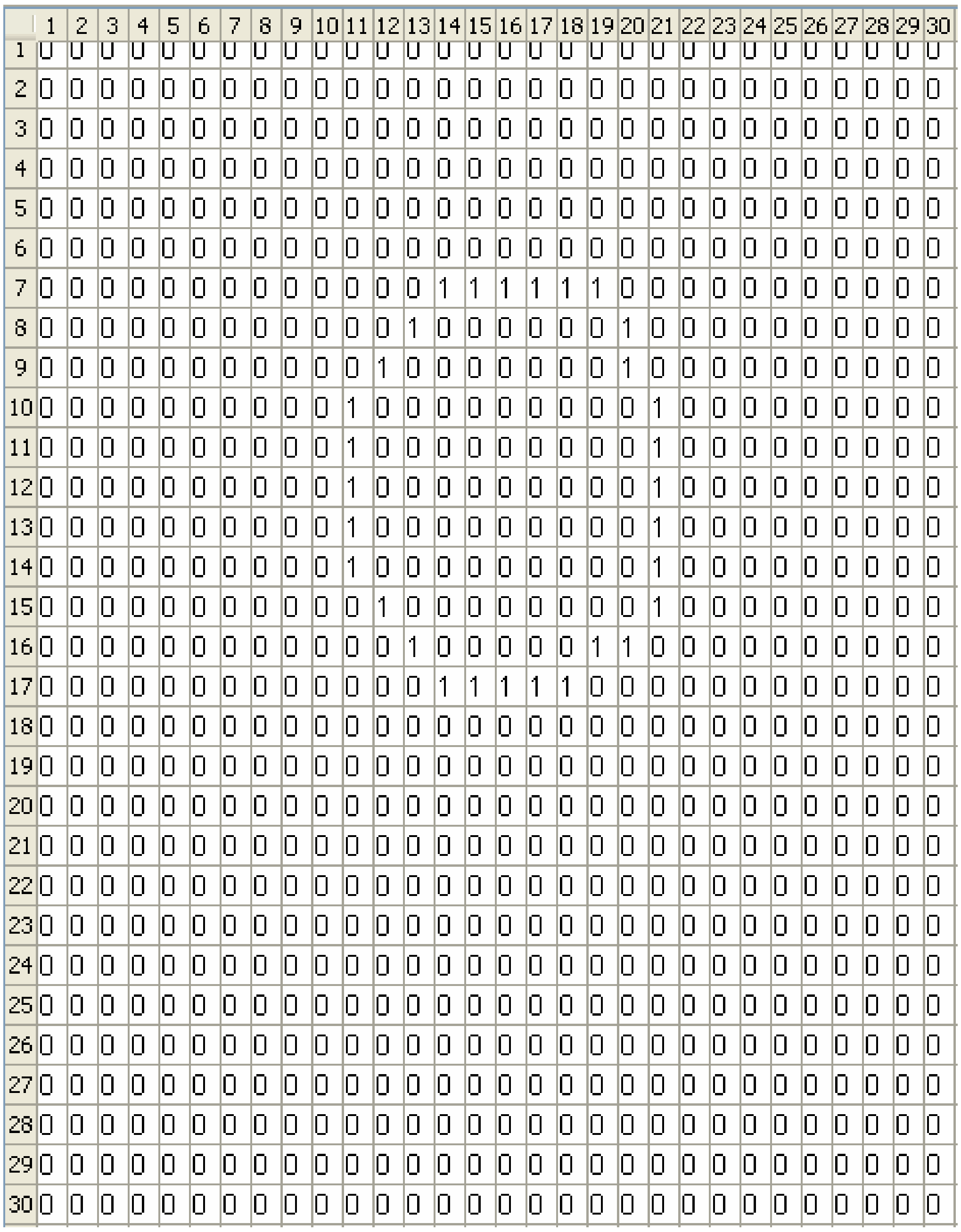
	

Figure 3.10: Number khamsa_5 after and before skeletonization



	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	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	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	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	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	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	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	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	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	0	1	0	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	0	0	1	0	0	0	0	0	0	0	0	0
15	0	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
16	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	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	0	0	0	0	0	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	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	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	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	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	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	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	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	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	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	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	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	0

Figure 3.11: The logical view of number khamsa_5 after skeletonization

When used with the 'thin' option, the function bwmorph uses the following algorithm:

1. Divide the image into two distinct subfields in a checkerboard pattern.
2. In the first subiteration, delete pixel p from the first subfield only if the conditions G1, G2, and G3 are all satisfied.
3. In the second subiteration, delete pixel p from the second subfield only if the conditions G1, G2, and G3' are all satisfied.

Condition G1:

$X_h(p)=1$

Where

$$X_h(p)=\sum_{i=1}^{4} b_i$$

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

x1, x2, ..., x8 are the values of the eight neighbors of p, starting with the east neighbor and numbered in counter-clockwise order.

Condition G2:

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

Where

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

$$n_2(p)=\sum_{k=1}^{4} x_{2k} \text{ or } x_{2k+1}$$

Condition G3:

$(x_2 \text{ or } x_3 \text{ or } x_8) \text{ and } x_1=0$

Condition G3':

$(x_6 \text{ or } x_7 \text{ or } x_4) \text{ and } x_5=0$

The preprocessing stage consists of four steps which are: converting the images into binary, resizing the images to 30x30, inversing the binary images and skeleton Figure 3.12 shows the result for the Hindi digits after preprocessing stage.

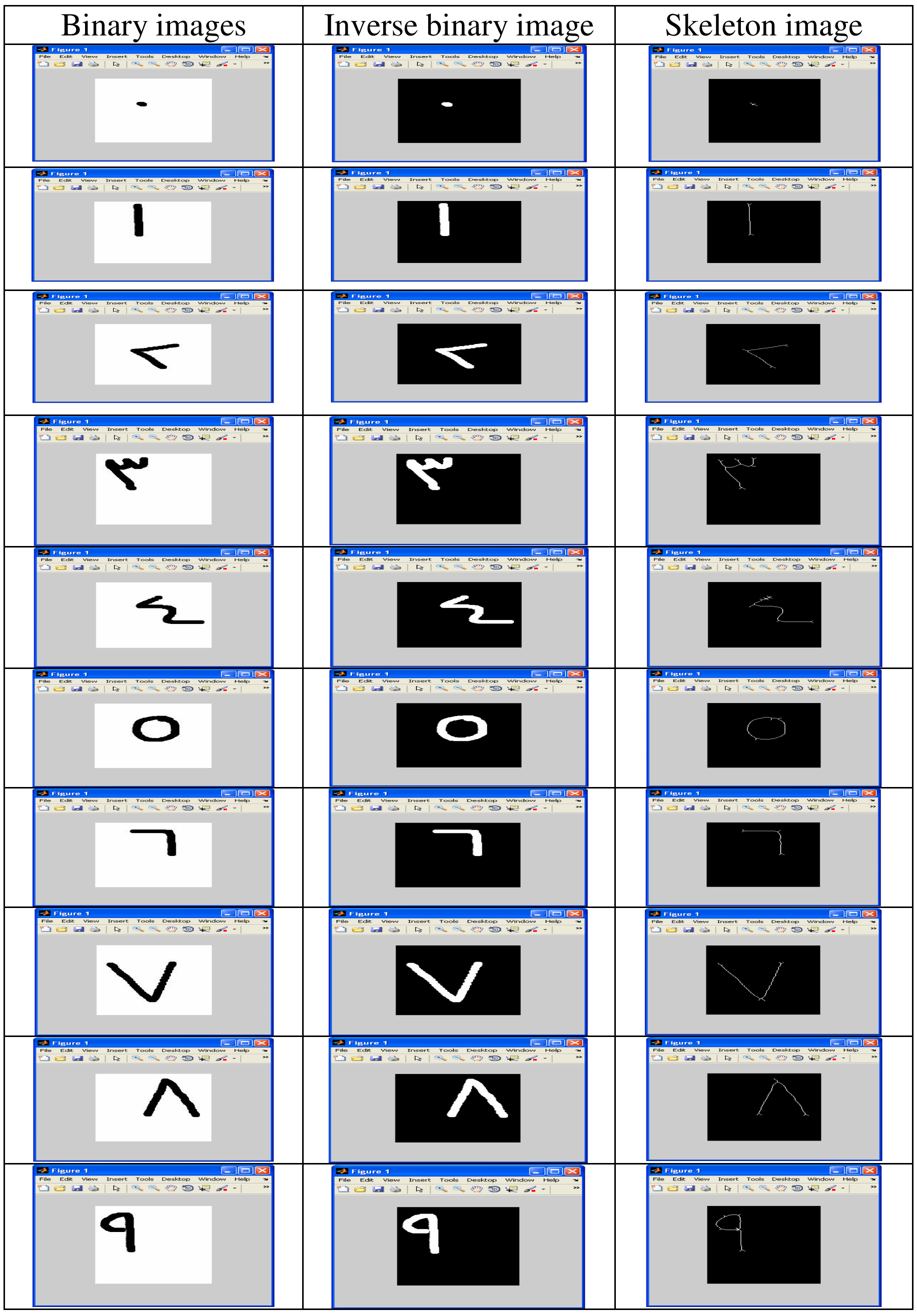


Figure 3.12: The results for the Hindi digits after preprocessing

3.4 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 (Ouchtati, et al., 2007).

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 the 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:

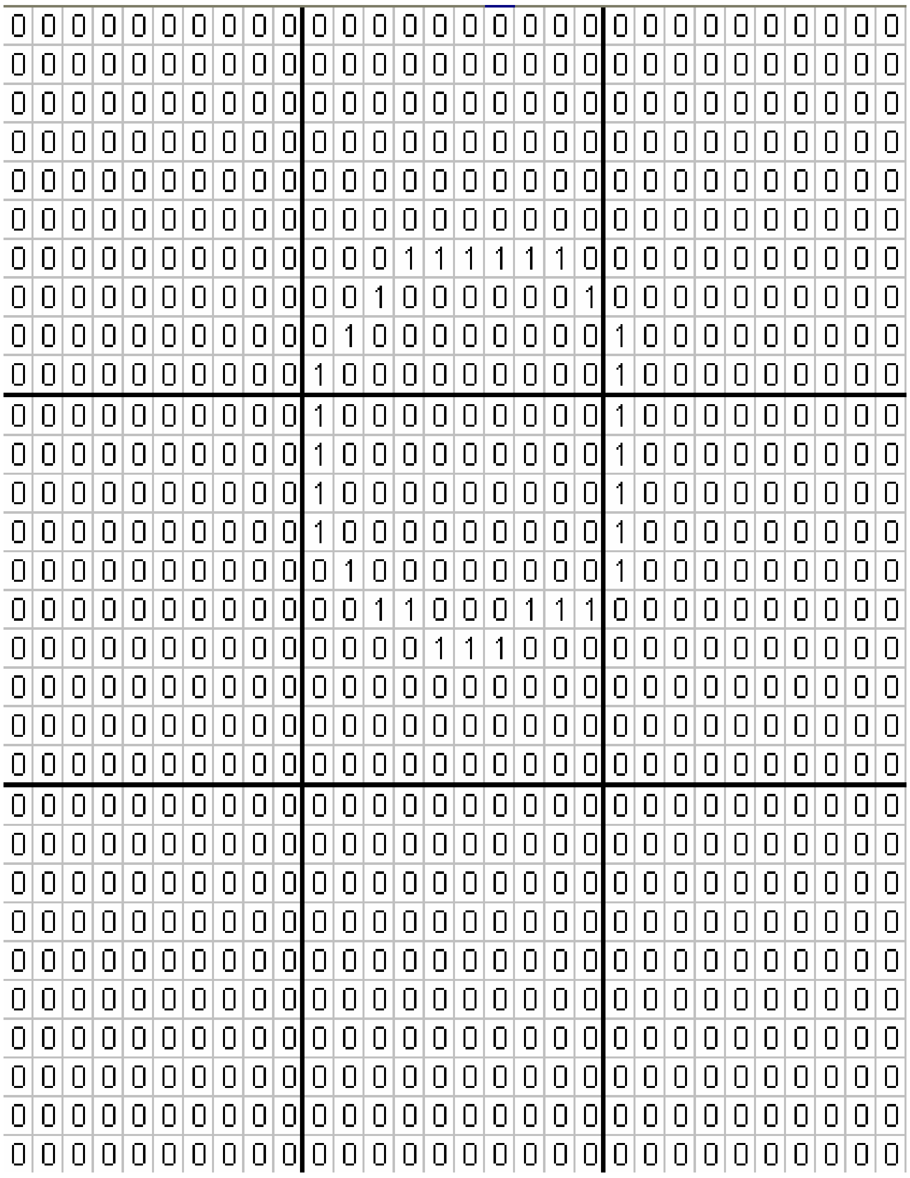
1. Store each image in a matrix with size 30x30
2. The system divides the image into blocks where each block has a size of 10x10 by dividing the image horizontally into 3 blocks and vertically into 3 blocks as shown in Figure 3.13.
3. For each block, store the following two features of each digit in a one dimension array called feature vector:

    1) 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.

2) The standard deviation, see the standard deviation equation 3.2.

$$s = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots.. 3.2.$$



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	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	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	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	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	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	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	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
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	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
0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	1	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	1	1	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	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	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	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	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	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	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	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	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	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	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	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	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	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

Figure 3.13: Blocks resulting from dividing an image

Now the vector contains 18 elements as shown in Table 3.2.

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

Table 3.2: Number khamsa_5 (18) feature vector

3.5 Digit recognition Stage

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.

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. Typically many such input/target pairs are needed to train a network.

We present an overview of neural-network properties, and then examine an application that illustrates them using the Matlab Neural Network Toolbox. Matlab (Matrix Laboratory) is a technical computing environment that provides a high-performance, interactive system for numerical computation and visualization.

A Feedforward neural network model for digit classification

All neural networks have an input layer of neurons to read the input data and an output layer. But non-linear recognition tasks such as handwritten character recognition cannot be accomplished by simple neural networks which have only input and output layers.

Multilayer Feedforward Networks distinguishes itself by the presence of one or more hidden layers, whose computation nodes are correspondingly called hidden neurons or hidden units. The function of hidden neurons is to intervene between the external input and the network output in some useful manner. By adding one or more hidden layers, the network is enabled to extract higher-order statistics. In most application a feed-forward network with a single layer of hidden units is used with a sigmoid activation function for the units.

Each layer of neurons feeds only the very next layer of neurons, and receives input only from the immediately preceding layer of neurons. Figure 3.14 shows the graphical representation of feedforward neural networks.

A feedforward neural network in our research serves as classification tools which are demonstrated in several studies as valuable classifiers for non linear function. It was used in many pattern recognition systems.

For neural networks, there are several available architectures and learning methods which may be selected depending on a given problem. In our work, we have selected a multilayer perceptron architecture called feedforward neural network using the backpropagation. We have chosen them because they are popular neural network methods used widely for several types of digit recognition problems, which are supervised learning.

3.5.1 The proposed system network architecture

The proposed feedforward neural network architecture is shown in Figure 3.14 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 digits which the program wants to recognize.

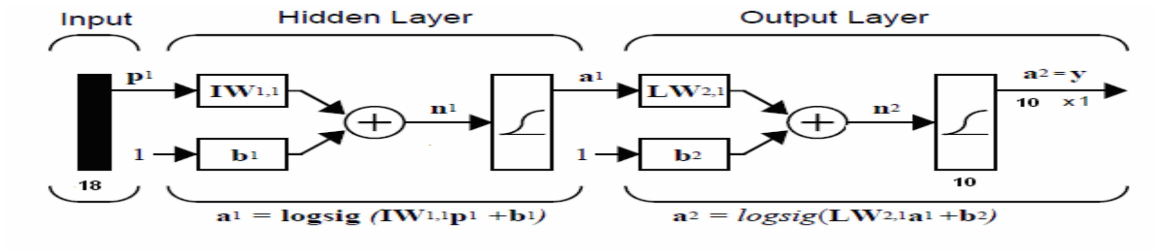


Figure 3.14: Three layers feedforward neural network

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

3.5.2 The number of nodes in each layer

1. 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 30x30, the system divided the image into blocks where each block has a size of 10x10 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.
2. 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).

The best number of hidden units depends on:

- The numbers of input and output units
- The number of training cases
- The amount of noise in the targets
- The complexity of the function or classification to be learned
- The network architecture
- The type of hidden unit activation function
- The training algorithm
- Regularization

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, et al., (1992) discuss 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.

3. 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 18x100x10 Neural Network model.

3.5.3 The activation (transfer) function

Neural network researchers have chosen from a variety of transfer functions; among the most popular are the logistic-sigmoid and tangent-sigmoid functions. Both hidden and output nodes have activation function and the function can be different. Notice that the sigmoid activation function at the output is optional. Only the activation function for the hidden must be used. The reason for using a sigmoid at the output is to force the output values to normalize between 0 and 1.

In the proposed 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 3.3.

logsig(n) = 1 / (1 + exp(-n))... 3.3

This function is illustrated in Figure 3.15.

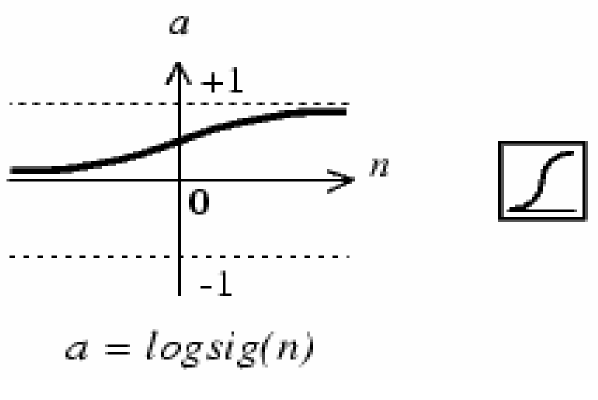


Figure 3.15: Logistic-sigmoid, range (0, 1)

3.5.4 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. Currently, there are more than a hundred different learning machines. In our system, we select 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.

In the proposed system, three layers backpropagation neural network are used which shown in Figure 3.16. The parameters x1, x2…..xi refer to input signals, y1, y2….yk refer to output signals.

This process aims to adjust the weights of the individual network connections so that learning can take place. The training algorithm adjusts their weights depending on the difference between the actual and the desired output which is called the training error.

At the end of the training process, the network stores its output as the final set of weights.

The process of training data requires a database as a background. This database should be prepared in advance. It also should consist of expected features for all types of digits that the system wants to recognize. As shown in Figure 3.16, the input signals are propagated from left to right and the error signals from right to left through the network.

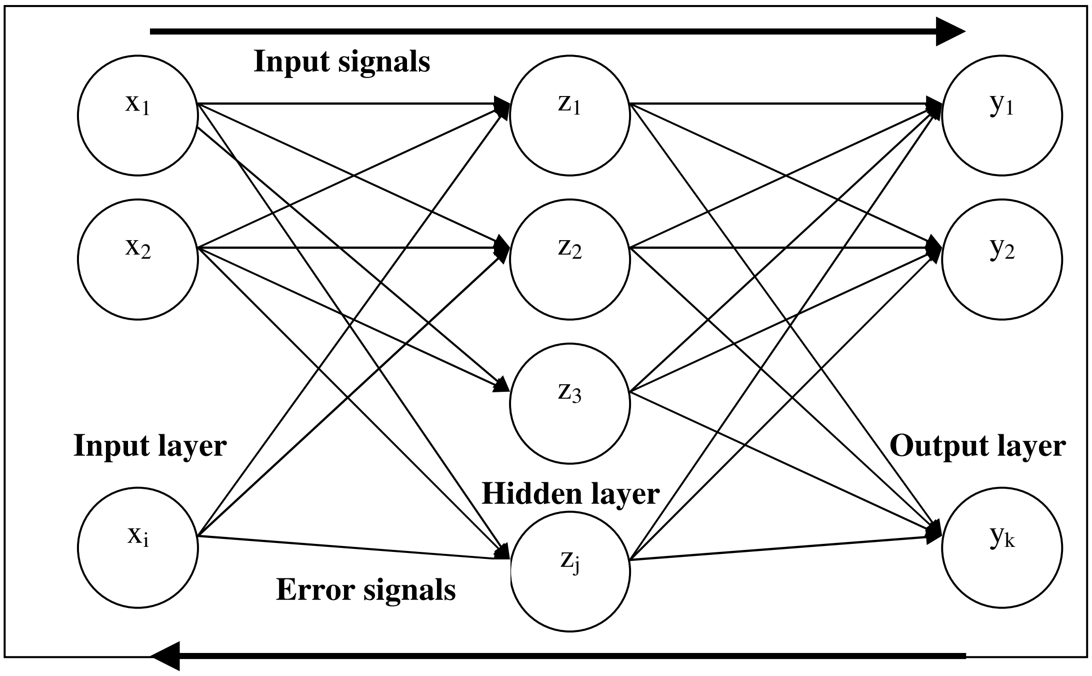


Figure 3.16: Three layers backpropagation

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.

1- While stopping, condition is false do steps 2-9.

2- For each training pair, do steps 3-8.

Feedforward:

3- Each input unit (x_i, i=1 to 18) receives an input signal xi and to all units in the layer above (the hidden units).

4- Each hidden unit (z_j, j=1 to 100) sums its weighted input signals

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

Apply its activation function to compute its output signal.

$$z_j = f\,(z\text{-}in_j)$$

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

5- Each output unit (y_k, k=1 to 10) sums its weight input signals.

$$y\text{-}in_k = w_{0k} + \sum_{j=1}^{n} z_i \, w_{ik}$$

Backpropagation of error:

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

$\partial_k \, (t_k - y_k) \, f' \, (y\text{-} in_k)$

calculate its weight correction term (used to update w_{ik} later)

$\Delta \, w_{ik} = \alpha \, \partial_k \, z_i$

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.

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

$$\partial\text{-}in_j = \sum_{k=1}^{m} \partial_k \, w_{ik}$$

multiplies by the derivative of its activation function to calculate its error information term.

$\partial_j = \partial\text{-}in_j \, f' \, (z\text{-} in_j)$

calculates its weight correction term (used to update vij 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:

8- Each output unit (y_k, k=1 to 10) updates its bias and weights (j=0, to 100)

w_{ik} (new) $=$ w_{ik} (old) $+ \Delta \, w_{ik}$

Each hidden unit (Zj, j=1 to 100) updates its bias and weights (i=0 to 18)

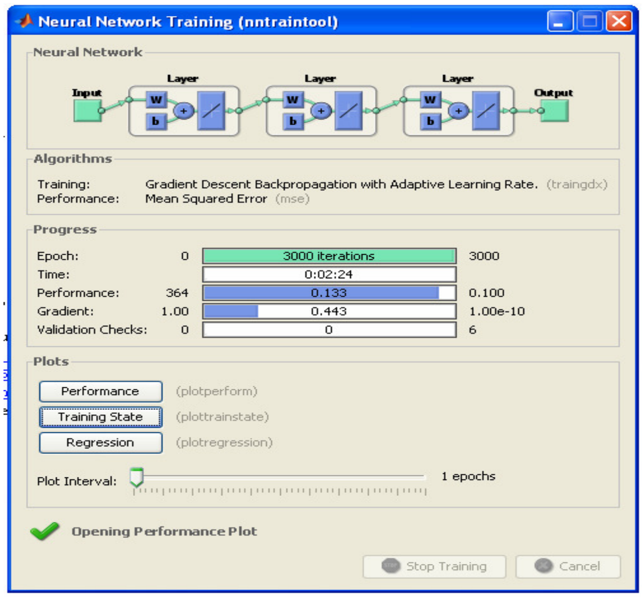
v_{ij} (new) $=$ v_{ij} (old) $+ \Delta \, v_{ij}$

9- Test stopping condition (Fausett, 1994).

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 listed below some examples of different neural networks.

First NN: Four layers network, purelin linear activation function which is a neural transfer function that calculate a layer's output from its net input for first hidden layer with 50 nodes, second hidden layer with 50 nodes and for output layer (see the equation 3.4 for this function). Mean squared error (MSE) used to judge the network performance to perform numeral recognition, see Figure 3.17.



```
Neural Network Training (nntraintool)

Neural Network
  Input   Layer   Layer   Layer   Output

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

Progress
  Epoch:              0     3000 iterations     3000
  Time:                     0:02:24
  Performance:      364     0.133               0.100
  Gradient:        1.00     0.443               1.00e-10
  Validation Checks:  0     0                   6

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

  Plot Interval:                               1 epochs

  Opening Performance Plot

                              Stop Training   Cancel
```

Figure 3.17: NN with two hidden layers using MSE and purelin

The performance of this network is shown in Figure 3.18.

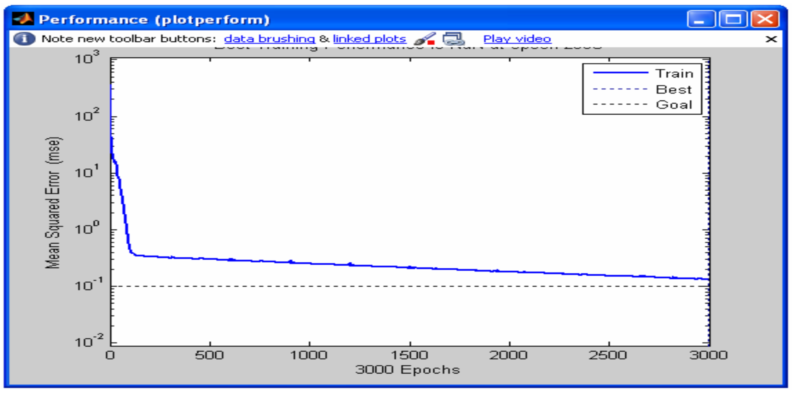
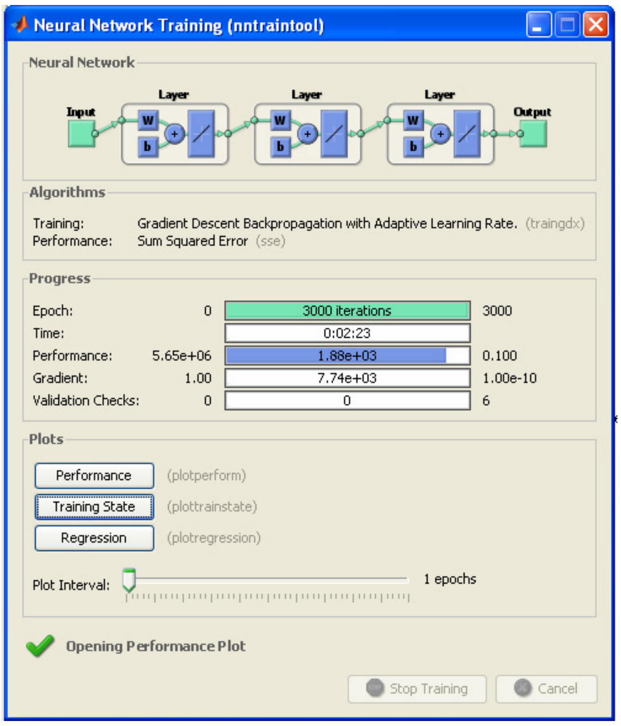


Figure 3.18: The performance of NN with two hidden layers using MSE and purelin

Purelin(n) = (W_p + b)……………………………………………… Equation 3.4

Where Wp is the weight and b is the bias.

Second NN: Four layers network, purelin activation function for first and second hidden layers 50 nodes for each, purelin for output layer. Sum squared error (SSE) used to judge the network performance to perform numeral recognition, see Figure 3.19.



Neural Network Training (nntraintool)

Neural Network

Algorithms

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

Progress

Epoch:	0	3000 iterations	3000
Time:		0:02:23	
Performance:	5.65e+06	1.88e+03	0.100
Gradient:	1.00	7.74e+03	1.00e-10
Validation Checks:	0	0	6

Plots

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

Plot Interval: 1 epochs

✓ Opening Performance Plot

Stop Training Cancel

Figure 3.19: NN with two hidden layers using SSE and purelin

The performance of this network is shown in Figure 3.20.

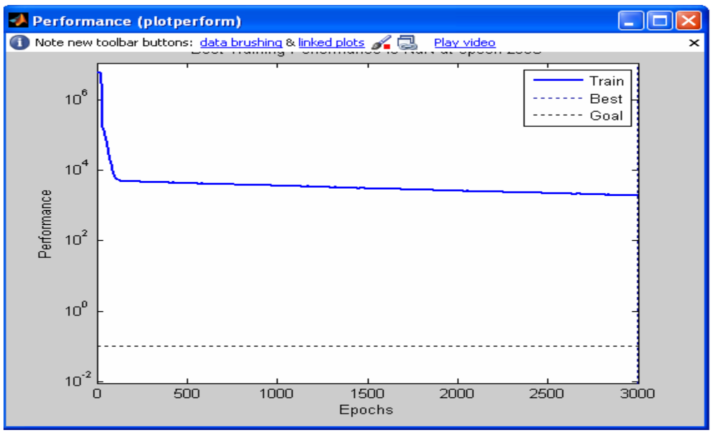
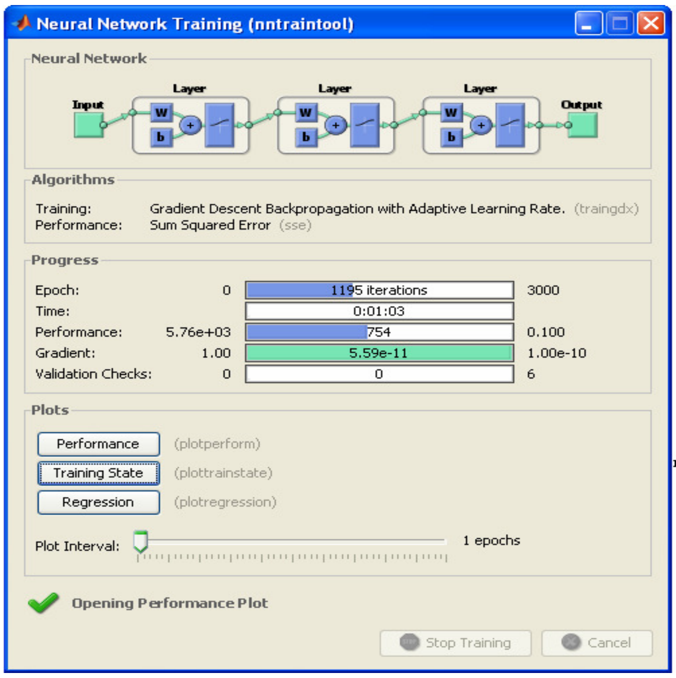


Figure 3.20: The performance of NN with two hidden layers using SSE and purelin

The linear transfer function calculates the neuron's output by simply returning the value passed to it. This neuron can be trained to learn an affine function of its inputs, or to find a linear approximation to a nonlinear function. A linear network cannot, of course, be made to perform a nonlinear computation.

Third NN: Four layers network, logsig activation function for the first and second hidden layers with 50 nodes for each, logsig for output layer. Sum squared error (SSE) used to judge the network performance to perform numeral recognition, see Figure 3.21.


Neural Network Training (nntraintool)

Neural Network

Input
Layer
W
b
Layer
W
b
Layer
W
b
Output

Algorithms

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

Progress

```
Epoch:              0         1195 iterations    3000
Time:                         0:01:03
Performance:        5.76e+03  754                0.100
Gradient:           1.00      5.59e-11           1.00e-10
Validation Checks:  0         0                  6
```

Plots

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

Plot Interval: 1 epochs

Opening Performance Plot

Figure 3.21: NN with two hidden layers using SSE and logsig

The performance of this network is shown in Figure 3.22.

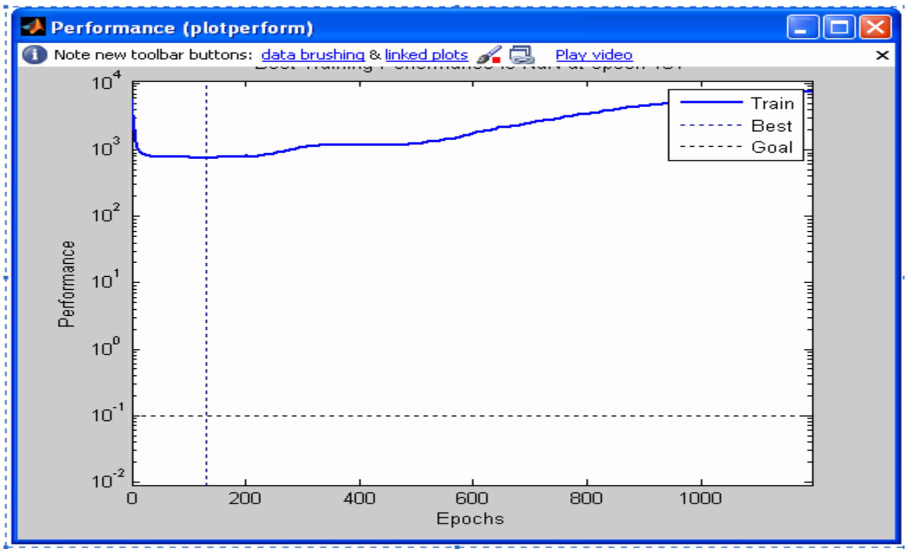
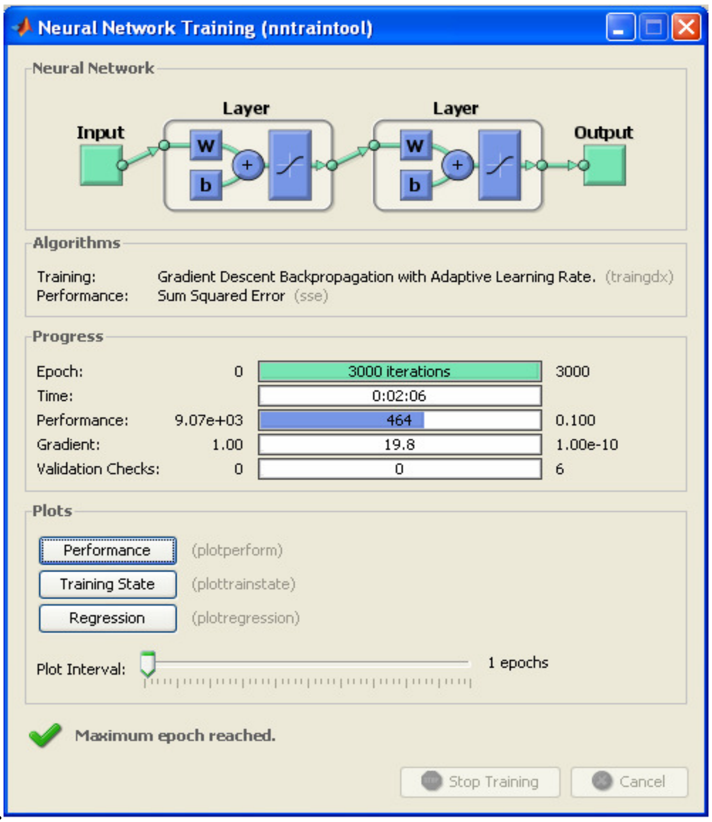


Figure 3.22: The performance of NN with two hidden layers using SSE and logsig

Forth NN: Three layers network, tansig Hyperbolic tangent sigmoid transfer function is a neural transfer function which calculate a layer's output from its net input and returns S x Q matrix of N's elements squashed into [-1 1] for hidden layer with 100 nodes and output layer see equation 3.5. Sum squared error (SSE) used to judge the network performance to perform numeral recognition, see Figure 2.23.

tansig(n) = 2/(1+exp(-2*n))-1……………………………………………… 3.5



Neural Network Training (nntraintool)

Neural Network

Algorithms

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

Progress

```
Epoch:               0      3000 iterations    3000
Time:                       0:02:06
Performance:   9.07e+03     464                0.100
Gradient:         1.00      19.8               1.00e-10
Validation Checks:   0      0                  6
```

Plots

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

Plot Interval: 1 epochs

Maximum epoch reached.

Figure 3.23: NN with one hidden layer using SSE and tansig

The performance of this network is shown in Figure 3.24.

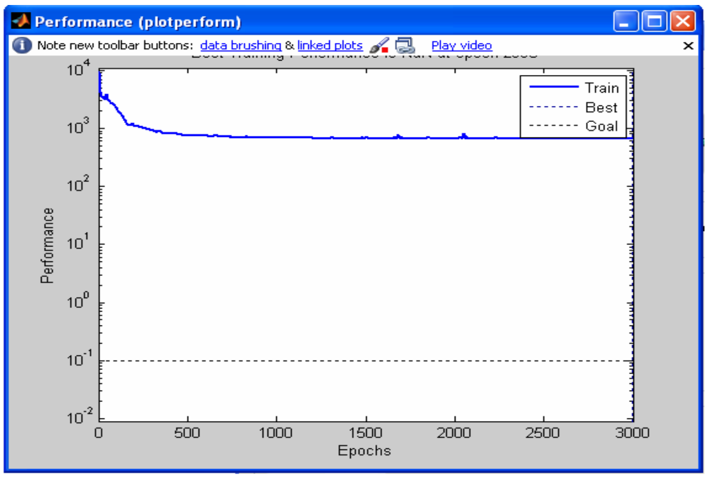
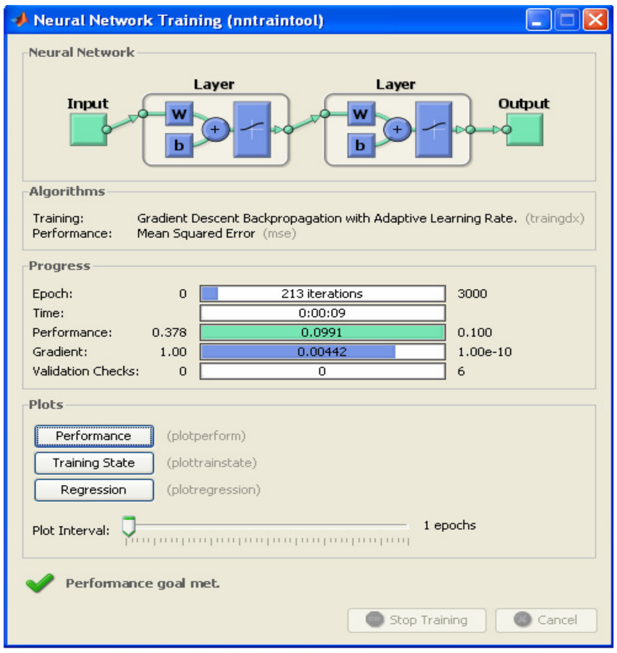


Figure 3.24: The performance of NN with one hidden layer using SSE and tansig

The last one 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, see Figure 3.25.


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       213 iterations    3000
Time:                       0:00:09
Performance:        0.378   0.0991            0.100
Gradient:           1.00    0.00442           1.00e-10
Validation Checks:  0       0                 6
```

Plots

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

Plot Interval: 1 epochs

Performance goal met.

Stop Training Cancel

Figure 3.25: NN with one hidden layer using MSE and logsig

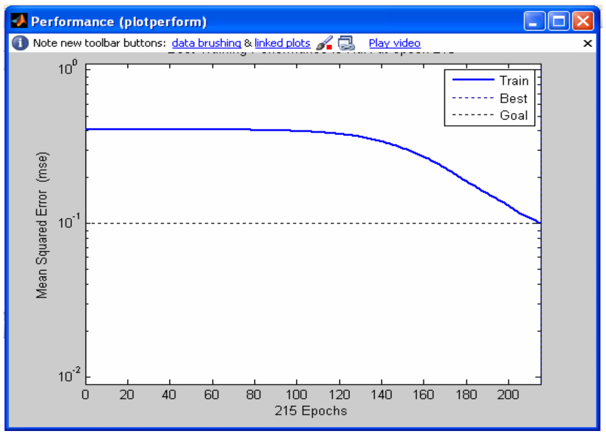


Figure 3.26: The performance NN with one hidden layer using MSE and logsig

Different Neural network with different numbers 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. The network which has the best performance and minimum error is chosen. So the proposed system has the NN with one hidden layer, using MSE network performance function, which measures the network's performance according to the mean of squared errors. And logsig activation functions for both hidden and output layers.

Chapter 4
Experimental Result and Discussion

4.1 The Experimental Results

Hindi digit Recognition using Neural Network (HDRNN) has been designed for classifying the Hindi digits. In this thesis, we present a review about digits recognition and its importance, and its applications, we also focus on important phases in recognition systems which are preprocessing and feature extraction.

HDRNN system has been designed and developed on 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.

The network with three layers has been experimented. The network has been trained on the training data set using feed forward back propagation algorithm.

There are 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. Most of the images are used to train the network because it's important to train with a large amount of digits.

The training error is an important indicator for the model performance. The backpropagation training algorithm tries to minimize this error. Figure 3.26 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. As shown in Figure 3.26, the learning process took epochs in order to train the model. 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.

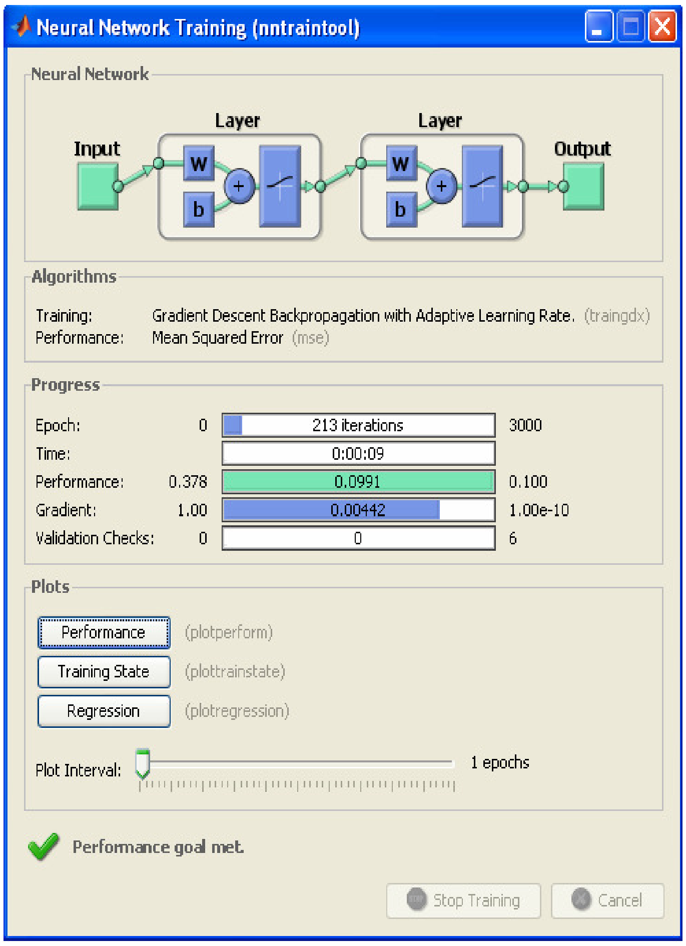
The (traingdx) function adjusts the weight and the bias of the different layers depending on the gradient in the output error. And the activation function Logistic-sigmoid, which is the most typical activation functions has a range of (0, 1) for both hidden and output nodes.

The objective is to minimize the error in the output so; the Mean Squared error (MSE) is used to judge the network performance to perform a numeral recognition.

In Figure 4.1, the training occurs according to traingdx's, its parameters shown in Table 4.1 with their values.

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

Table 4.1: The function traingdx parameters



Algorithms

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

Progress

```
Epoch:              0   213 iterations   3000
Time:                   0:00:09
Performance:    0.378   0.0991           0.100
Gradient:        1.00   0.00442          1.00e-10
Validation Checks:  0   0                6
```

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

Plot Interval: 1 epochs

Performance goal met.

Figure 4.1: Neural network training structure

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 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.

Hindi Number	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%

Table 4.2: Recognition accuracy testing results

Table 4.2 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%). these testing results were summarized for all digits in Table 4.2.

So, these results show 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 important and the hard 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 in Figure 4.2. On the other hand, in Figure 4.3, some examples of digits, which were poorly written, were not classified correctly.

Most of these samples were taken from children with the age less than 5 years old. They found some problems by using the mouse to draw the numbers.

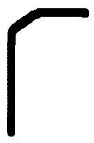
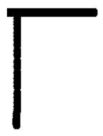
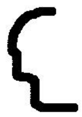
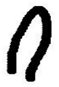
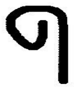
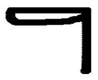
 sefer_0	 wahad_1	 ethnan_2	 ethnan_2	 thalatha_3	 arbaa_4
 khamsa_5	 seta_6	 Sabaa_7	 thamaneya_8	 Tesaa_9	 tesaa_9

Figure 4.2: Some examples of poorly written digits and correctly classified

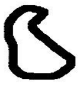
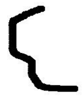
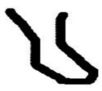
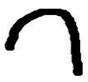
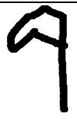
 wahad_1 Recognized as 0	 ethnan_2 Recognized as 3	 Thalatha_3 Recognized as 2	 arbaa_4 Recognized as 3	 khamsa_5 Recognized as 8
 arbaa_4 Recognized as 2	 seta_6 Recognized as 2	 sabaa_7 Recognized as 9	 thamaneya_8 Recognized as 6	 tesaa_9 Recognized as 7

Figure 4.3: Some examples of poorly written digits and badly classified

4.2 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.

The input data is an image format file containing numbers. The data feed into the system, and then passed through several preprocessing steps before it can be recognized.

The preprocessing steps are image processing methods, such as converting the grayscale image into a binary image, resizing the image and reversing the binary image. Then the binary image passed through a thinning technique to convert it into a skeleton image.

Name	**Approach**	**Features used**	**Handwritten type**	**Average recognition rate**
LeCun et al.	Backpropagation neural network	shared-weight averaging	Zip code	87%-89%
Shilbayeh et al.	FCC with eight connectivity	FCC with 8 connectivity	Hindi digits	89%
Al-Taani	Structural Features and Transition Network	Shape-based features like curve, line, dot	Arabic Digits	95%
Ahanga	Multilayer perceptron with one hidden layer	used global input	Farsi characters	80%
Mahmoud et al.	SVM and HMM classifiers	Local, intermediate, and large scales features	Hindi digits	99.83% 99.00%
Al-Omari et al.	Feed forward back propagation neural network algorithm	Feature extraction is not part of their work	Arabic digits	95%

Table 5.1: The average recognition rate for different approaches

As shown in Table 5.1, we have listed the recognition rate for Le Cun, et al., (1989) as 87%-89% with the backpropagation neural network. 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 HDRNN system had a better performance in the case of Mouse Gesture Hindi Digit comparing to Shilbayeh, et al., (2009), MGHD system used (FCC) with eight connectivity and then used a template to recognize the Hindi digit. 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 neural network seems to be better than other techniques used for recognition with high accuracy and less time.

A-Taani, (2006) 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 work. And 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 & 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.

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.

Ahanga, (2009), system has 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.

Chapter 5
Conclusion and Future work

5.1 Conclusion

Handwriting recognition has been an active research area for many years. During the recent years, many methods have been developed or improved to obtain higher recognition rate on various data.

The recognition of handwritten digits is difficult due to several reasons, including the fact that there are different writing styles of different persons. This results in digits of different sizes and strokes that vary in width and shape.

Different neural networks with different number of layers and different neurons per layer can be chosen to recognize the handwritten digits. Also, different transfer functions could be used for each layer.

Trying to train a MLP of NN is usually a tedious process which takes much time. There are many variations proposed to improve the standard backpropagation as well as other learning algorithms that do not suffer from these limitations.

The proposed system for recognizing handwritten Hindi digits has been designed and tested. The features of the Hindi digits after preprocessed throw a number of steps used as the inputs of the MLP network. The experimental result which shows that backpropagation network yields high recognition accuracy of about 91% on the sample test. Experiments have showed that NN approach is flexible and can achieve high recognition rate for the shapes of the digits represented in this study.

We can conclude that the HDRNN system reached the computer to the importance use of isolated digits recognition for different applications. This recognition starts with acquiring the image to be preprocessed through a number of steps, which is one of important phases in recognition systems. We then extract the feature from each image to produce the feature vector which is the input of the MLP network.

In a final conclusion, neural network seems to be better than other techniques used for recognition.

5.2 Future Work

Future work considers testing the method on a larger data set to improve the effectiveness of our method, since we get most of the writing variations of the digits by different users.
The proposed method will be modified to deal with Arabic handwritten characters and mathematical symbols. In addition, the next important work is to add additional constraints on the primitives. For example, the feature extraction methods can affect and guarantee accurate results. Another work is to solve the problem of bad recognition for the poorly written digits.

Finally, this system can be used in many applications such as postal codes, banking account numbers, courtesy amount in bank checks, numbers on car plates, and students ID etc.

References

Ahangar, R. G. & Ahangar, M. F. (2009): Handwritten Farsi character recognition using artificial neural network. ***International Journal of Computer Science and Information Security*,** vol. 4, No. 1 & 2.

Breiman, L. (1994): Neural networks: a review from statistical perspective, ***Comment, StatisticalScience*** ,vol. 9, No. 1., pp. 38-42.

Choudhari P. 1. , Prof. Dr. Choudhari N. S. (2006): A new approach for handwritten numeral character recognition. ***Journal on Engineering*** , vol.3 No.1, pp 13-16.

Dibeklioglu H. (2007). Digit Recognition Using HMM and MLP. (Online), available:http://www.cmpe.boun.edu.tr/~dibeklioglu/projects/dr_ann_hmm.pdf

Fausett, L. (1994). **"***Fundamentals of neural networks architectures, algorithms and applications***"**, New Jersey: Prentice Hall.

Geman S., Bienenstock E. and Doursat R. (1992). Neural networks and the bias/variance dilemma. ***Neural Computation*** vol. 4, pp.1-58.

Gołda A. (2005)**.** Principles of training multi-layer neural network using backpropagation. (Online) available: http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Al-Imo˘glu, F. & Alpaydin, E., (2001). Combining multiple representations for pen-based handwritten digit recognition. ***Turk J Elec Engin***, vol.9, no.1, Istanbul-Turkey.

Jain, A. K. & Zongker D. (1997). Representation and recognition of handwritten digits using deformable templates. ***IEEE transactions on pattern analysis and machine intelligence,*** vol. 19, no. 12 pp. 1386-1390.

Jain, G. & Ko, J. (2008). Handwritten Digits Recognition. (Online), available: http://users.cs.dal.ca/~mheywood/Reports/TKlassen.pdf

Klassen, T. (2001). Towards neural network recognition of handwritten Arabic letters. (Online), available: http://users.cs.dal.ca/~mheywood/Reports/TKlassen.pdf

Knerr, S., Personnaz, L. & Dreyfus, G. (1992). Handwritten digit recognition by neural networks with single-layer training. ***IEEE Transactions on Neural Networks***, vol. 3, 962.

Kröse, B. & Smagt, P. (1996). An introduction to Neural Networks" (Online), available: URL:http://www.fwi.uva.nl/research/neuro/ URL:http://www.op.dlr.de/FF-DR-RS/ .

LeCun, Y., Jackel, L.D., Bottou, L., Cortes, C., Denker, J.S., Drucker, H., Guyon, I., Muller, U.A., Sackinger, E., Simard, P., and Vapnik, V. (1990). "Learning algorithms for classification: a comparison on handwritten digit recognition", ***Neural Networks: The Statistical Mechanics Perspective*****,** pp. 261-276.

LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., &Jackel, L.D. (1989). Backpropagation applied to handwritten zip code recognition, ***Neural Computation*****,** vol.1, no. 4, pp. 541-551.

Lorigo, L. M. & Venu, G., (2006). **"Off-line Arabic handwritten recognition**" IEEE, vol . 28, Pages:40,1-40.

Mahmoud, S. A. & Awaida S. M. (2009). Recognition of off-line handwritten arabic (indian) numerals using multi-scale features and support vector machines vs. hidden markov models. ***The Arabian Journal for Science and Engineering***, vol. 34, Number 2B.

Mashor, M. Y., Sulaiman, S. N., (2001). Recognition of Noisy Numerals using Neural Network. ***International Journal of the Computer, the Internet and Management***, vol. 9(3), 45-52.

Mike, O., (2006). Neural Network for Recognition of Handwritten Digits. (Online) available: http://www.codeproject.com/KB/library/NeuralNetRecognition.html.

Al-Omari, S. A., Sumari, P., Al-Taweel S. A. & Husain A. J., (2009). Digital recognition using neural network. ***Journal of Computer Science,*** vol. 5 (6): 427-434.
Ouchtati, S., Bedda M. & Lachouri A. (2007). Segmentation and recognition of handwritten numeric chains**.** ***Journal of Computer Science,*** vol. 3 (4): 242-248.

Parkins, A.D. and Nandi, A.K. (2004). Genetic programming techniques for hand written digit recognition, ***Signal processing****,* vol. 84, pp.2345-2365.

Al-Rashaideh H. (2006).Preprocessing phase for Arabic word handwritten recognition. ***Том*** 6, № 1, 2006, стр. 11-19

Senior, A. W., (1992) .Off-line handwriting recognition: A review and experiments, Technical Report, Cambridge University Engineering Department.

Shilbayeh M.F, G. Raho and M. Alkateeb (2009). "An Efficient Structural Mouse Gesture Approach for Recognizing Hindi Digits". ***Asian Network for Scientific Information*** vol**.** 9(19): 3469-3479.

Stergiou, C. & Siganos, D., (1996). Introduction to Neural Networks. (Online), available: URL:http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/cs11/report\.

Al-Taani A. (2008). Recognition of on-line handwritten arabic digits using structural features and transition network", ***Informatica*** vol**.** 32 (2008) 275–281.

Teredesai, A. Ratzlaff, E. Subrahmonia J. & Govindaraju V. (2002). On-Line digit recognition using off-line features". ICVGIP 2002, Proceedings of the Third Indian Conference on Computer Vision, Graphics Image Processing, Ahmadabad, India. (Online), available: http://www.ee.iitb.ac.in/~icvgip/PAPERS/321.pdf.