A

Thesis

on

AN ANALYSIS OF NEURAL NETWORK CLASSIFIER FOR HANDWRITTEN CHARACTER RECOGNITION

Submitted in partial fulfillment of the requirement
for the award of the degree of

MASTER OF TECHNOLOGY
(INFORMATION SYSTEM)

Submitted by
VEERENDRA SINGH
(15/IS/09)

Under the Guidance of
SEBA SUSAN
(Asst. Professor)
Dept. of Information Technology

DELHI TECHNOLOGICAL UNIVERSITY
(DEPARTMENT OF INFORMATION TECHNOLOGY)
BAWANA ROAD, DELHI-110042

SESSION: 2009-11

CERTIFICATE

This is to certify that, the thesis titled as "**AN ANALYSIS OF NEURAL NETWORK CLASSIFIER FOR HANDWRITTEN CHARACTER RECOGNITION**" is being submitted by **VEERENDRA SINGH,** Roll No. **15/IS/09**, in partial fulfillment for the award of **"Master of Technology Degree in Information System"** in Delhi Technological University, Delhi; is the original work carried out by him under the guidance and supervision. The matter contained in this report has not been submitted elsewhere for reward for any other degree.

I wish him good luck for his future.

SEBA SUSAN
(Asst. Professor)
Dept. of Information Technology
Delhi Technological University
Bawana Road, Delhi- 110042

ACKNOWLEDGEMENT

Fist and foremost I would like to pay my gratitude to my parents for their constant encouragement and support in every endeavor of my life.

I would like to express my hearty gratitude and thanks to my project guide **Mrs. SEBA SUSAN**, Assistant Professor, Department of Information Technology, Delhi Technological University, Delhi. For continuous inspiration, encouragement and guidance in every stage of preparation of this project report.

I am extremely thankful to **Professor O. P. VERMA** , H.O.D., Department of Information Technology, Delhi Technological University, Delhi, for the support provided by him during the entire duration of degree course and especially in this thesis.

I am thankful to all teaching and non-teaching staff at Delhi Technological University, and my fellows, who have helped me directly or indirectly in completion of this thesis report.

VEERENDRA SINGH
ROLL NO. 15/IS/09
Master of Technology
Information System

ABSTRACT

The recognition of handwritten numerals or characters has applications in the field of pattern recognition, document processing and analysis. However, the recognition of handwritten numerals or characters is a challenging task for machine because the writing styles vary from person to person. Handwritten numeral or characters recognition is in general a benchmark problem of Pattern Recognition and Artificial Intelligence. Compared to the problem of scanned numeral recognition, the problem of handwritten numeral recognition is compounded due to variations in shapes and sizes of handwritten English numerals. This makes the recognition of handwritten numerals or characters an intensive area of research in pattern recognition. Considering all these, the problem of handwritten numeral recognition is addressed under the present work in respect to handwritten English numerals. This dissertation presents a new approach for handwritten numeral recognition method using the MLP (Multi Layer Perceptron) neural network for classification purpose. To achieve the desired task of recognition, a numeral is first enclosed inside a bounding box whose size is normalized to 42 x 30 pixels. The numeral bounding box is partitioned into 9 sub boxes and features are extracted using bounding box approach is given as input to the MLP classifier. Many images with different fonts and styles have been processed in MATLAB environment using the aforementioned classifier. It is observed from the results that the proposed technique successfully classifies handwritten numerals used in our experiment. The proposed scheme proves its utilization for recognition handwritten English numerals represented with the said feature set. This approach of recognition of handwritten English numerals by MPL classifier can also be extended to include handwritten characters of English alphabet.

CONTENT

<u>List of Figures</u>

List of Tables

CHAPTER 1

INTRODUCTION

The recognition of handwritten numeral is an intensive branch area of research in pattern recognition. Handwriting has infinite variety of style because it varies from person to person, so it is very difficult to recognize numerals for a machine. Although the researches have been going on in this field for last few decades and many algorithms have been developed to recognize the handwritten numeral with high accuracy but, the ultimate goal is still out of reach. As variation in style of handwriting depends on human thought, so recognition of unconstrained handwritten numerals becomes very difficult. An overview of the handwritten numeral recognition, the need of the new approach and the literature on the concerned topic is emphasized in this chapter.

1.1 Overview of Handwritten Numeral Recognition

This provides a huge scope of development in the field of handwritten character recognition. Any development in the field of handwritten numeral or character recognition will be able to excel the communication between men and machine. The automatic recognition and verification of handwritten legal and courtesy amounts in English language present on bank cheques is used to separate numerals to recognize and courtesy the handwritten numeral of English language which are written on the bank cheque in Indian banks [1].

A multilayer perceptron neural network (MLPNN) is very helpful tool to recognize unconstraint handwriting and extract important features from numerals. The performance of handwritten numeral recognition is largely dependent on the normalization of numeral shapes. In addition to transforming the numeral image to a standard size so as to give a representation of fixed dimensionality for classification, it is hoped that the deformation of characters can be restored so as to reduce the within-class shape variation. The conventional linear normalization is not sufficient to restore this kind of deformation. To improve the recognition performance, many methods have been proposed for normalizing handwritten numerals. The ability to identify handwritten numbers in an automated or semi automated manner has led to the development of an entirely different field of research known as the optical character recognition (OCR). Handwritten input usually occurs as connected or partially connected strings of characters, and the first stage in the recognition process usually involves “chopping” the strings into locally separate entities. This process is known as segmentation. Recognition is then performed using these entities. Since segmentation is a hard problem within itself, the assumption of pre-segmented inputs eases the task and; schemes have to be developed for the isolated input characters.

Recognizing numerals is a problem that at first seems simple, but it is a very difficult task to program a computer to do it. The difficulties and complexity of this task lie in the fact that a computer program must be able to recognize handwritten digits produced by different people, using different instruments. The system has to deal with widely different sizes and slants, with different shapes and widths of the strokes. Handwritten numeral recognition has been an especially active topic in the field of pattern classification and learning. Many approaches and methods have been proposed for pre-processing, feature extraction, classification and/or learning of handwritten numeral images. Researchers have also compiled widely used standard image databases to evaluate the performance of these approaches and methods. The most important and widely used databases are CENPARMI, CEDAR, MNIST, and the United States Postal Service (USPS) database. Many domains require recognition of digit strings, as opposed to individual digits. Some examples are automated sorting of mail by postal code, automated reading of checks, tax returns, data entry and many other applications. In these domains, handwritten digits rarely appear isolated. Instead they appear as part of a string of digits where some digits may touch and/or overlap. In many of these real world applications, the images are processed by human operators. However, automation may improve production and cut costs. For this to happen, performance of an automated system must compare favorably to human performance. Such comparison is an essential component in determining whether the problem has been solved or not. An automated system cannot be fully integrated into real world applications until the performance gap between humans and machines is sufficiently minimal.

In hand printed numeral recognition, the numerals are assumed to be written legibly allowing smaller variation in the shape of a numeral. Handwritten numeral recognition is always the research focus in the field of image process and pattern recognition. The numeral varieties in size, shape, slant and the writing style make the research harder. The numeral character recognition is the most challenging field, because the big research and development effort that has gone into it has not solved all commercial and intellectual problems. Handwritten numeral character recognition is an important step in many document processing applications. Digital document processing is gaining popularity for application to office and library automation, bank and postal services, publishing houses and communication technology. The complexity of the problem is greatly increased by noise in the data and infinite variability of handwriting as a result of mood of writer and nature of writing. Recognition of handwritten digits has been popular topic of research for many years. The recognition of handwritten numeral character has been considerable interest to researchers working on OCR.

Offline numeral recognition has many practical applications. Among the most quoted practical applications of offline numeral recognition is the reading of postal zip codes in addresses written or typed on envelopes. The benefits of implementing such systems in post offices are enormous. Such systems would make possible automatic sorting and routing of the millions of mails that flow through the postal system every day, reducing the human workload and speeding up the whole process. However, despite extensive research, the current techniques do not produce results that meet the desired accuracy. A trusted and reliable numeral recognition system needs to have very high accuracy. The recognition of handwritten numerals has many applications such as office automation, postal sorting, bank cheque recognition, etc. Since handwritten numerals and characters are used in highly sensitive areas like finance and administration, the recognition system should be accurate, fast and easy to implement The earlier systems grouped under optical character recognition (OCR) could recognize only the printed or handwritten numerals of fixed size and fonts. But, 100 percent recognition rate is beyond the reach of these systems as reported in the literature. Therefore, the present study aims at producing an accurate system targeting 100 percent recognition in face of varied size, shapes and fonts.

MLP-NN is one of the handwritten numeral recognition methodology based on simplified structural classification, by using a much smaller set of primitive types. The major advantage of MLP classifier approach stems from its robustness, easy to implement and relatively high recognition rate. Moreover, this method is insensitive to barbs and gaps that arise at the preprocessing stage. If the MLP-NN gets well trained after applying training input, it guarantees high recognition rates. From the results obtained in course of the study presented in this thesis we can emphatically advocate the use of MLP for document processing.

In this thesis features are extracted from numerals images using the proposed bounding box approach. In this method, the captured image is converted into binary image and then the image is partitioned into a 3x3 number of sub images called boxes. The features consist of number of black pixels from each box. Each sub box is divided by total no of pixels. Here, MLP neural network technique is used for recognition and recognition rates are found to be around 90 percent using MLP neural network. The methods are independent of font, size and with minor changes in preprocessing.

1.2 LITERATURE SURVEY

The handwritten numeral recognition is big research area so problems have been studied using neural networks, structural classification, In recent times, the researchers find the better result by combining two or more approaches in hand written numerals recognition such as feature extraction, classification, fuzzy membership function and many other features etc [6].The recognition of handwritten Hindi and English numerals by representing them in the form of exponential membership functions, which serve as a fuzzy model has been done in [2,7,8,10]. The box approach is used for feature extraction which makes the recognition rate very high [2,7,8]. Fuzzy logic is applied for feature extraction, classification for handwritten numerals [11]. Integration of the statistical and structural information for unconstrained handwritten Numerals [9] is done using HMMs and the structural information are modeled by singletons and relationships between macro-states. The zoning based system is designed to extract features which calculate densities of each object pixels in each zone [5]. The structural classification of the handwritten numeral recognition method is by using three types of feature extraction methods: primitive segment, tree-like classifier and fuzzy memberships. The tree-like classifier used to extract the feature points, primitives from the image and fuzzy memberships is applied to the classified numeral’s image [3].

The handwritten numeral recognition problem has been studied from syntactic recognition, neural networks, structural classification, coding, fuzzy logic, etc. Recently, there is a trend of combining two or more recognition methods to obtain better recognition rate [2, 12, 13, 14]. Among the above methodologies, structural classification is the most frequently used, with feature extraction as its base. [15] decomposed images of handwritten numerals into a set of 15 branching features, which are of the following three categories: (1) straight lines: horizontal, vertical, positive slope, and negative slope; (2) circles: plain circle and circle on the left, on the right, above, and below; and (3) open arcs: C-like, D-like, A-like, V-like, S-like, and Z-like. Their decomposition was based on the detection of the following feature points: tips, corners, and junctions. To increase recognition rate, [16] increased the number of primitives of the above categories, and used multi-stage feature aggregation to describe each image using fuzzy rules. The increase of the number of primitives makes the computation very complex. For each primitive, [17] collected local information of curvature, moving direction, length, etc. to form a primitive code of 11 elements. A global code of 5 elements was then deduced from the primitive codes to reflect global topological information, like the number of primitives. Based on the global codes, matching rules are designed for 103 prototypes and 26 subclasses. A neural network was employed to overcome their time-consuming design process.[35] divided each image into 16 equal-sized partitions, and then extracted 7 features from each partition to form a global vector of 112 features, which were then sent to a neural network for classification. As some features are dependent on the others, they erased some of them to obtain better classification results. [2] presents the recognition of handwritten numerals by representing them in the form of exponential membership functions, which serve as a fuzzy model. Their parameters were derived from features consisting of normalized distances obtained by dividing each image into a number of boxes.

Most of the Indian scripts are distinguished by the presence of matras (or, character modifiers) in addition to main characters as against the English script that has no matras. Therefore, algorithms developed for them are not directly applicable to Indian scripts. Many OCRs for Indian scripts have been reported [19-23]. However, none of these has attempted the handwritten Hindi text consisting of composite characters that involve both the main characters and matras. Printed Devanagari character recognition is attempted using Kohenen neural network (KNN) and other types of neural networks [19, 23, 24]. These results are extended to Bangla [25], which also has the header line like Hindi. Structural features like concavities and intersections are used as features. A similar approach is tried for Gujarati in [22] with limited success. Reasonable results are reported for Gurumukhi script [23]. Preliminary results are also available in the literature on recognition of two popular scripts in South India—Tamil and Kannada [21]. Sinha et al. [25, 26] have reported various aspect ratios (ARs) for Devanagari script recognition. Sethi and Chatterjee [27] have described Devanagari numeral recognition using the structural approach. The primitives used are horizontal and vertical line segments, right and left slants. A decision tree is employed to perform the analysis based on the presence/ absence of these primitives and their interconnection. A similar strategy is applied to the constrained hand-printed Devanagari characters in [28]. Neural network approach for isolated characters is also reported in [29]. MLP neural networks have been recognized as a powerful tool for pattern classification problems, but a number of researchers have also suggested that straightforward neural-network approaches to pattern recognition are largely inadequate for difficult problems such as handwritten numeral recognition. The three sophisticated neural network classifiers to solve complex pattern recognition problems: (1) multiple multilayer perceptron (MLP) classifier (2) hidden Markov model (HMM)/MLP hybrid classifier and (3) structure adaptive self-organizing map (SOM) classifier[12,31,37]. A method for recognizing handwritten Hindi numerals based on the structural descriptors of a numeral's shape is given in [30]. Density, segment and moment features are used as inputs to the classifiers with feed-forward superstructure of the Kohonen modules in [21]. The density features consist of density of the normalized samples. The moment features consist of moments of orders 2, 3 and 4 of the numeral. The feature vector for the segments consists of: (1) normalized length of the segment; (2)

average directivity of the segment; (3) average directivity strength of the segment; (4) average directivity change of the segment. These segments of the numeral are computed based on [31]. The three individual classifiers are then integrated using meta-pi network. Zernike moments are used as features and neural network as classifier on the hand printed Devanagari characters in [32]. A contour-following based algorithm for extracting features is presented in [33]. For classification of the encoded patterns by the nearest neighbor (NN) classifiers, an iterative clustering algorithm is developed to obtain a reduced, but efficient number of prototypes. A new feature extraction technique is presented in [34] and applied to the printed Hindi numerals. Classification is done using the neural networks.

Handwritten English numeral recognition using fuzzy logic is first attempted by Siy and Chen [35]. The handwritten numeral is decomposed into straight lines, portions of a circle and circles. A more flexible scheme is proposed in [36] where the decomposition is based on the detection of a set of feature points: terminal, intersection and bend points. The perturbations due to writing habits and instruments are taken into account in the recognition of off-line handwritten English numerals in [37]. Handwritten numeral recognition using self-organizing maps (SOM) and fuzzy rules is presented in [38]. During the learning phase prototypes are produced, which together with the corresponding variances, are used to determine fuzzy regions and membership functions. In the recognition stage, a fuzzy rule based classifier is employed to classify an input pattern. An unsure pattern is re-entered into the SOM classifier. A technique in [39] generates fuzzy rules, based on ID-3 approach by optimizing the defuzzification parameters with the help of a two-layer perceptron. This technique overcomes the difficulties in a conventional syntactic approach for handwritten character recognition including the problems of choosing a starting or reference point scaling and learning by machines. A new scheme for offline recognition of totally unconstrained handwritten characters using a simple multi-layer cluster neural network trained with the back propagation algorithm is presented in [40] and it is shown that the use of genetic algorithms avoids the problem of finding local minima that arise from training the multi-layer cluster neural network with gradient descent technique. Modified Hough transform method is used in [41] to extract features, which are fed as the input to a linear classifier (discriminate analysis) and a non-linear classifier (NN). Two neural architectures along with two different feature extraction techniques are investigated in [42] along with a novel technique for character feature extraction. A new technique for local decision combination is presented in [43]. The technique uses a genetic algorithm to determine the optimal weight vector to balance the local decision in the combination process. All these methods demonstrate their effectiveness on CEDAR database.

The benchmarking of the state of the art techniques for the digit recognition from different databases, viz., CEDAR, MNIST and CENPARMI is undertaken in [44]. Eight classifiers, viz., *k*-NN (*k*-neural networks), MLP (multi layer perceptron), RBF (radial basis function), PC (polynomial classifier), two variants of SVC (support vector classifier), LVQ (learning vector quantization) and LQDF (learning quadratic discriminate function) are employed. The 10 features include : 4-orientation and 8-direction chain code features, 4-orientation plus 133D and 8-direction chain code features plus 233D with 11 crossing counts and 22 concavity measurements, 4-orientation and 8-direction gradient features, 4-orientation and 8-direction gradient features from pseudo gray scale images, 4-orientation Kirsch features and 4-orientation PDC features. The 10 features and eight classifiers are combined to give 80 accuracies to the test dataset of three databases. The recognition rates are found to be higher than the best in the literature.

In [45] two types of 256-dimensional feature vectors: Contour based gradient distribution (CGD) and directional distance distributions (DDD) are extracted. Using non-parametric method the features are evaluated in terms of their class separation and recognition capabilities. From GCD and DDD two types of new features are derived: Class-common and Class-dependent. Using the separating power of the class-common features and discriminating power of the class-dependent features, modular network architectures are trained to classify the numerals. In [27], statistical information is modeled by microstates using HMMs and structural information is modeled by singletons and relationships between macrostates where each macrostate is a collection of individual microstates. The orientations that constitute the statistical information are encoded into discrete codebooks and distributions of locations that constitute the structural information are modeled by the joint Gaussian distribution functions. The recognition process is measured by the degree of matching. The state-duration adapted transition probability gives this degree as state-duration indicates the number of feature segments modeled by the state. The back-propagation neural network is used in [46] for the recognition of handwritten characters. In that, feature extraction is done using three different approaches, namely, ring, sector and hybrid. The features consist of normalized vector distances and angles. The hybrid approach, which combines the ring and sector approaches, is found to yield the best results. The same features are adopted in [8]. We follow the methodology in [46] for feature extraction in the present work. But the recognition approach in [8] which is solely based on the membership function is extended to include entropy and the membership functions are modified by way of the structural parameters necessitated because of large variations in the means and variances of samples. In order to avoid this problem; consistent samples have been used in [8].

CHAPTER 2

MULTILAYER PRECEPTRON NEURAL NETWORK (MLP-NN)

The multilayer perceptron (MLP) is a hierarchical structure of several perceptrons, and overcomes the shortcomings of the single-layer networks. The MLP consists of three or more layers (an input layer and an output layer with one or more hidden layers) of neurons with nonlinear activation function. The hidden neurons make enable the multilayer network to learn complex tasks by extracting progressively more meaningful information from the input. The MLP is capable of learning a rich variety of nonlinear decision surfaces. Nonlinear functions can be represented by multilayer perceptrons with units that use nonlinear activation functions. Multiple layers of cascaded linear units still produce only linear mappings.

"A neural network with one or more layers of nodes between the input and the output nodes is called multilayer perceptron neural network."

The multilayer network MLP has a highly connected topology since every input is connected to all neurons in the first hidden layer; every unit in the hidden layers is connected to all neurons in the next layer, and so on. The input signals propagate through the neural network in a forward direction on a layer-by- layer basis that is why they are often called feed forward multilayer networks. Two kinds of signals pass through these networks:

(1) Function signals: The inputs are propagated through the hidden neurons and processed by their activation functions emerge as outputs;

(2) Error signals: The error at the output neuron is propagated backward layer-by-layer through the network so that each node returns its error back to the nodes in the previous hidden layer.

2.1 Definition of activation function

The training algorithm for multilayer networks requires differentiable, continuous nonlinear activation functions. If a multilayer perceptron has a linear activation function in all neurons, that is, a simple on-off mechanism to determine whether or not a neuron fires, then it is easily proved with linear algebra that number of layers can be reduced to the standard two-layer input-output model. What makes a MLP different is that each neuron uses a nonlinear activation function which was developed to model the frequency of action potentials or firing of neurons in the MLP. This function is modeled in several ways, but must always be normalized and differentiable.

The two main activation functions used in current applications are both sigmoid, and are described by

$\Phi(y_i) = \tanh(v_i)$

$\Phi(y_i) = (1+e^{-v_i})^{-1}$

in which the former function is a hyperbolic tangent which ranges from -1 to 1, and the latter is equivalent in shape but ranges from 0 to 1. Here y_i is the output of the ith neuron and v_i is the weighted sum of the input synapses. More specialized activation functions include radial basis functions which are used in another class of supervised neural network models. But the hyperbolic tangent is preferred because it makes the training easier than others.

2.2 Structure of MLP

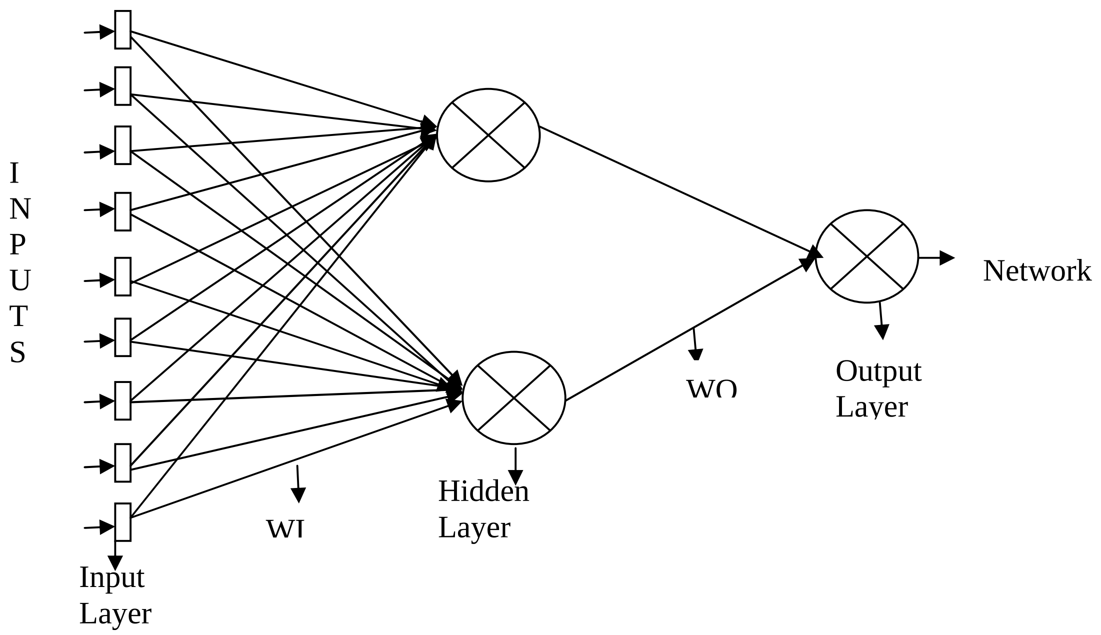


Fig2.1 Structure of Multilayer perceptron

$$WI = \begin{Bmatrix} WI_{1,1} WI_{1,2} WI_{1,3} \\ WI_{2,1} WI_{2,2} WI_{2,3} \\ WI_{3,1} WI_{3,2} WI_{3,3} \end{Bmatrix} \qquad WO = \begin{Bmatrix} WO1 \\ WO2 \end{Bmatrix}$$

The structure of MLP is shown in the Fig2.1 with one input layer, one hidden layer and one output layer. In this each input of input layer is directly connected to each neuron of hidden layer and output of hidden layer is directly connected to output layer of the neural network. Each neuron has activation function to convert the output in nonlinear form. Input is applied to the hidden layer is 9 dimension and WI is input weights are also 9 dimension. Then output of the hidden neuron works as input to output layer neuron which is 2 dimensions and WO is output weights for output layer neuron is also 2 dimensions. Inputs are applied and weights are initialized to the network is shown in Fig2.1 after this training is done in each iteration and network output is calculated. This output is matched with target output and error is calculated from above formula. This process is running till error is minimized.

Input Layer: A vector of predictor variable values (I1 I2 …. I9) is presented to the input layer. The input layer standardizes these values so that the range of each variable is -1 to 1. The input layer distributes the values to each of the neurons in the hidden layer. In addition to the weights are initialize to each of the hidden layers; the weights are multiplied by input and added to the sum going into the neuron.

Hidden Layer: Arriving at a neuron in the hidden layer, the value from each input neuron is multiplied by a weight, and the resulting weighted values are added together producing a combined value. The weighted sum is fed into a transfer function, which produce output. The outputs from the hidden layer are distributed to the output layer.

Output Layer: Arriving at a neuron in the output layer, the value from each hidden layer neuron is multiplied with output weight, and the resulting weighted values are added together producing a combined value. The weighted sum is fed into a transfer function, which produce output that is the outputs of the network.

2.3 Training of MLP:

The main objective of the training of the data in MLP to find set of weights that will produce the output of the neural network to match the desired output to close the target output of neural network to minimize the network error. This totally depends on the designing and training.

The training algorithms follow this cycle to refine the weights values:

(1) Run a set of predictor variable values through the network using a tentative set of weights.

(2) Compute the difference between the predicted target value and the required output value for this case.

(3) Average the error information over the entire set of training cases.

(4) Propagate the error backward through the network and compute the gradient (vector of derivatives) of the change in error with respect to changes in weight values.

(5) Make adjustments to the weights to reduce the error.

There are several issues involved in designing and training a MLP:

(a)Selecting the Number of Hidden Layers

For nearly all problems, one hidden layer is sufficient. Two hidden layers are required for modeling data with discontinuities such as a saw tooth wave pattern. Using two hidden layers rarely improves the model, and it may introduce a greater risk of converging to a local minima.

(b)Deciding how many neurons to use in the hidden layers

One of the most important concerns is that how many neurons should be in the hidden layer(s) in MLP. If an inadequate number of neurons are used, the network will be unable to model complex data, and the results fitting will be poor. If too many neurons are used, the training time may become excessively long, and, worse, the network may over fit the data. When over fitting occurs, the network begins to model random noise in the data. The result is that the model fits the training data extremely well, but it poorly generalizes the new, unseen data. Validation must be used to test for this. An automated feature is used to find the optimal number of neurons in the hidden layer. You specify the minimum and maximum number of neurons you want it to test, and it will build models using varying numbers of neurons and measure the quality using either cross validation or hold-out data not used for training. This is a highly effective method for finding the optimal number of neurons, but it is computationally expensive, because many models must be built, and each model has to be validated. If you have a multiprocessor computer to use multiple CPU’s during the process, this method may be adopted. The automated search for the optimal number of neurons only searches the first hidden layer. If you select a model with two hidden layers, you must manually specify the number of neurons in the second hidden layer.

2.4. Backpropagation

The backpropagation training algorithm was first described by Rumelhart and McClelland in 1986; it was the first practical method for training neural networks. Backpropagation is feedback neural network used to minimize error of the neural network that assigns the weights to the hidden layer neurons. After each iteration of training input the error is calculated from networks output and desired output in the neural network. After that weights are adjust automatically for new iteration until the desired output is achieved.

Back Propagation Weights Updating Rule

MLP is shown in Fig2.1, with one layer of hidden neurons and one output neuron. When an input vector is propagated through the network, for the current set of weights the network output is P_{red}. The objective of supervised training is to adjust the weights automatically so that the difference between the network output P_{red} and the required output R_{eq} is reduced. This requires an algorithm that reduces the absolute error, which is the same as reducing the squared error.

$$\text{Network Error (E)}= P_{red} - R_{eq} \qquad (1)$$

The algorithm should adjust the weights automatically such that E2 is minimized. Backpropagation is such an algorithm that performs a gradient descent minimization of E2. In order to minimize E2, its sensitivity to each of the weights must be calculated. We see that what effect changing each of the weights will have on E2. If this is known then the weights can be adjusted in the direction that reduces the absolute error. The notation for the following description of the back-propagation rule is based on the diagram below.

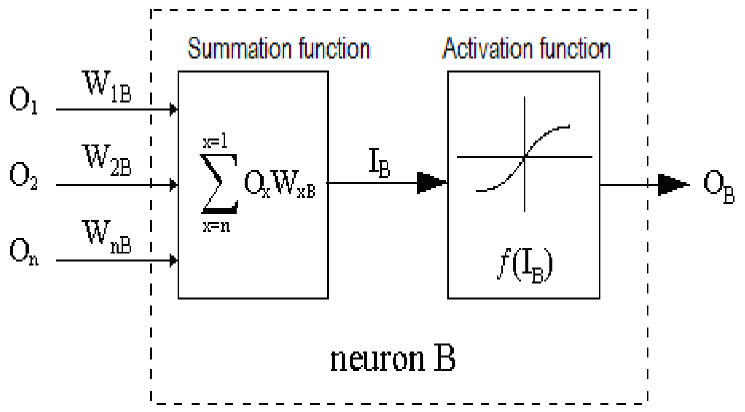


Fig.2.2 Architecture of a Neuron

The dashed line represents a neuron B, which can be either a hidden or the output neuron. The outputs of n neurons ($O_1 \ldots O_n$) in the preceding layer provide the inputs to neuron B. If neuron B is in the hidden layer then this is simply the input vector. These outputs are multiplied by the respective weights ($W_{1B} \ldots W_{nB}$), where W_{nB} is the weight connecting neuron n to neuron B. The summation function adds together all these products to provide the input I_B that is processed by the activation function f (.) of neuron B. f (I_B) is the output O_B of neuron B.

For the purpose of this illustration, let neuron 1 be called neuron A and then consider the weight W_{AB} connecting the two neurons.

The approximation used for the weight change is given by the delta rule:

$$W_{AB(new)} = W_{AB(old)} - \eta \frac{\partial E^2}{\partial W_{AB}} \qquad (2)$$

Where η is the learning rate parameter, which determines the rate of learning, and

$$\frac{\partial E^2}{\partial W_{AB}}$$

is the sensitivity of the error E^2 to the weight W_{AB} and determines the direction of search in weight space for the new weight $W_{AB(new)}$ as illustrated in the figure below.

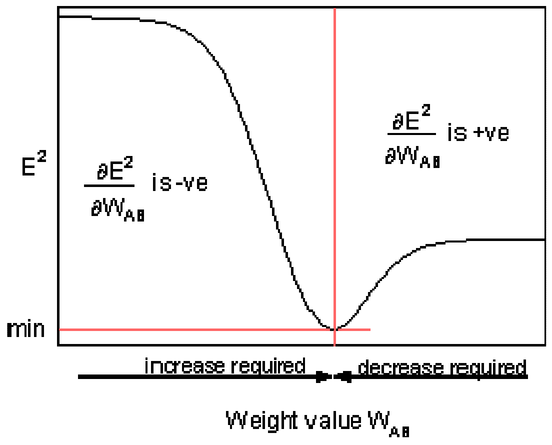


Fig 2.3 In order to minimize E^2 the delta rule gives the direction of weight change required

From the chain rule,

$$\frac{\partial E^2}{\partial W_{AB}} = \frac{\partial E^2}{\partial I_B}\frac{\partial I_B}{\partial W_{AB}} \tag{3}$$

and, $$\frac{\partial I_B}{\partial W_{AB}} = \frac{\partial \sum_{X-x}^{X-1} O_X W_{XB}}{\partial W_{AB}} = \frac{\partial (O_A W_{AB})}{\partial W_{AB}} + \frac{\partial \sum_{X-x}^{X-2} O_X W_{XB}}{\partial W_{AB}} = O_A \tag{4}$$

Since the rest of the inputs to neuron B have no dependency on the weight W_{AB}. Thus from equation (3) and (4), equation (2) becomes,

$$W_{AB(new)} = W_{AB(old)} - \eta \frac{\partial E^2}{\partial I_B} O_A \tag{5}$$

And the weight change of W_{AB} depends on the sensitivity of the squared error (E^2) to the input (I_B) of neuron B and on the input signal O_A. There are two possible situations:

1. B is the output neuron;
2. B is a hidden neuron.

Considering the first case:

Since B is the output neuron, the change in the squared error due to an adjustment of W_{AB} is simply the change in the squared error of the output of B:

$$\partial E^2 = \partial(\Pr ed - \operatorname{Re} q)^2$$

$$\frac{\partial E^2}{\partial I_B} = 2(\Pr ed - \operatorname{Re} q)\frac{\partial \Pr ed}{\partial I_B}$$

$$= 2E\frac{\partial f(I_B)}{\partial I_B}$$

$$= 2Ef'(I_B) \qquad (6)$$

Combining equation (5) with (6) we get the rule for modifying the weights, when neuron B is an output neuron i.e.

$$W_{AB(new)=}W_{AB(old)} - \eta O_A 2Ef'(I_B) \qquad (7)$$

If the output activation functions, f (**.**), is the logistic function then:

$$f(x) = \frac{1}{1+e^{-x}} = (1+e^{-x})^{-1} \qquad (8)$$

Differentiating (8) by its argument x;

$$f'(x) = -1(1+e^{-x})^{-2} - 1(e^{-x}) = \frac{e^{-x}}{(1+e^{-x})^{-2}} \qquad (9)$$

But, $$f(x) = \frac{1}{1+e^{-x}}$$

Where $$e^{-x} = \frac{(1-f(x))}{f(x)} \qquad (10)$$

Inserting (10) into (9) gives:

$$f'(x) = \frac{(1 - f(x))}{f(x)} \Big/ \frac{1}{(f(x))^2}$$

$$= f(x) \times (1 - f(x)) \qquad (11)$$

Similarly for the tanh function,

$$f'(x) = (1 - f(x)^2)$$

or for the linear (identity) function,

$$f'(x) = 1$$

This gives:

$$W_{AB(new)} = W_{AB(old)} - \eta O_A 2 E O_B (1 - O_B) \qquad \text{(logistic)}$$

$$W_{AB(new)} = W_{AB(old)} - \eta O_A 2 E (1 - O_B{}^2) \qquad \text{(tanh)}$$

$$W_{AB(new)} = W_{AB(old)} - \eta O_A 2 E \qquad \text{(linear)}$$

Considering the second case: B is a hidden neuron**.**

$$\frac{\partial E^2}{\partial I_B} = \frac{\partial E^2}{\partial I_o} \frac{\partial I_o}{\partial O_B} \frac{\partial O_B}{\partial I_B} \qquad (12)$$

Where the subscript, o, represents the output neuron.

$$\frac{\partial O_B}{\partial I_B} = \frac{\partial f(I_B)}{\partial I_B} = f'(I_B) \qquad (13)$$

$$\frac{\partial I_o}{\partial O_B} = \frac{\partial \sum_P O_P W_{PO}}{\partial O_B} \qquad (14)$$

Where p is an index that ranges over all the neurons including neuron B, that provides input signals to the output neuron. Expanding the right hand side of equation (14),

$$\frac{\partial \sum_{P} O_P W_{P^o}}{\partial O_B} = \frac{\partial O_B W_{BO}}{\partial O_B} + \frac{\partial \sum_{P}^{P \neq B} O_P W_{P^o}}{\partial O_B} = W_{BO} \qquad (15)$$

since the weights of the other neurons ,W_{pO} (p!=B) have no dependency on O_B. Inserting (13) and (15) into (12),

$$\frac{\partial E^2}{\partial I_B} = \frac{\partial E^2}{\partial I_o} W_{BO} f\,'(I_B) \qquad (16)$$

Thus $\partial E^2 / \partial I_B$ is now expressed as a function of $\partial E^2 / \partial I_O$, calculated as in (6). The complete rule for modifying the weight W_{AB} between a neuron A sending a signal to a neuron B is,

$$W_{AB(new)} = W_{AB(old)} - \eta \frac{\partial E^2}{\partial I_B} O_A \qquad (17)$$

Where,

$$\frac{\partial E^2}{\partial I_B} = 2Ef_o\,'(I_B)$$ -I_B is the output neuron

$$\frac{\partial E^2}{\partial I_B} = \frac{\partial E^2}{\partial I_0} W_{B0} f_k\,'(I_B)$$ -I_B is the output neuron

where $f_o(.)$ and $f_h(.)$are the output and hidden activation functions respectively.

Network Output = [tanh (I^T .WI)] . WO

Let,

HID = The outputs of the hidden neurons

ERROR = (Network Output - Required Output)

LR (Learning rate) = 0.2(constant)

Error threshold = 0.0001

linear output weights updating

$$WO = WO - (LR \times ERROR \times HID) \qquad (18)$$

Linear input weights updating

$$WI = WI - \{ LR \times [ERROR \times WO \times (1- HID^2)] . I^T \}^T \qquad (19)$$

Equations 18 and 19 show that the weights change is an input signal multiplied by a local gradient. This gives a direction that also has magnitude dependent on the magnitude of the error. If the direction is taken with no magnitude then all changes will be of equal size which will depend on the learning rate.

The algorithm above is a simplified version in that there is only one output neuron. In the original algorithm more than one output is allowed and the gradient descent minimizes the total squared error of all the outputs. With only one output this reduces to minimizing the error.

CHAPTER 3

FEATURE EXTRACTION METHOD

This section describes the proposed method used for handwritten numeral recognition. Feature extraction is the crucial phase in numeral identification as each numeral is unique in its own way, thus distinguishing itself from other numerals. Hence, it is very important to extract features a way that the recognition of different numerals becomes easier on the basis of the individual features of each numerals [2].

3.1 Bounding box approach

For extracting features, we use the Bounding Box- approach. This approach requires the spatial division of the numeral image. The major advantage of this approach stems from its robustness to small variations, ease of implementation and relatively high recognition rate. The choice of box size and number of boxes is discussed in Ref [2]. Each character image is divided into 9 sub boxes so that the portions of a numeral will be in some of these boxes.

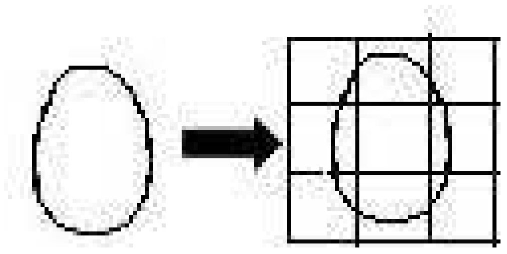


Fig3.1 Partitions of the numeral image in 9 sub boxes

English numeral 0 is enclosed in the 3×3 grid. However, all boxes are considered for analysis in a sequential order. By calculating no of black pixels in each sub box and each sub box is divided by total no of black pixels as shown in Table3.1.

0.1497	0.1020	0.1361
0.1088	0	0.1156
0.1633	0.0816	0.1429

Table3.1 A 9 dimension feature of numeral image in Fig3.1

3.2 Processing of numeral image

Step1. The numeral images are captured by the camera.

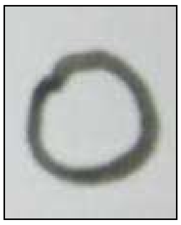


Fig3.2 Captured image

Step2. Binarization of the captured image: Binarization is carried out before the character recognition phase. Ideally an input character should have two tones, i.e., black and white pixels (commonly represented by 1 and 0, respectively). Image binarization converts an image of up to 256 gray levels into a two-tone image.

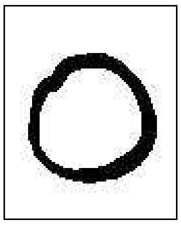


Fig3.3 Binarized image

Step2. Thicken operation is applied on the image.

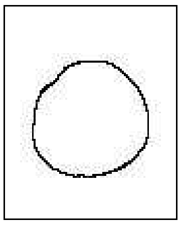


Fig3.4 Thick image

Step3. Crop the image into the fitted window.

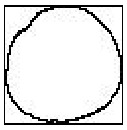


Fig3.5 Cropped image

Step4. Resize this image into 42 x 30 pixels.

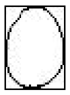


Fig3.6 Resized image

Step5. Partition of this image into 3 x 3 boxes each of size 14*10.

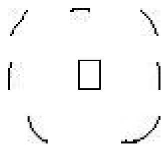


Fig3.7 The 9 sub partitioned images of Fig3.7

Step6. Compute the total no of black pixels in each sub box and divide the total black pixels in the image. This fraction is applied as the training input to the MLP classifier.

Step7. Test input is applied after training and the percentage of correct classification is calculated.

CHAPTER 4

EXPERIMENTAL RESULTS AND DISCUSSION

In the present chapter a setup for the image processing is developed and the results of the algorithm on the images are presented.

4.1 EXPERIMENTAL SETUP

Experiment was performed on image of numerals captured by a camera (resolution 10 mp and 3x optical zoom) to examine the efficiency of proposed algorithm. To implement proposed algorithm, system configured with Intel processor 2.63 GHz and 1 Gigabyte of RAM used and matlab2009 tool used.

The following parameters were set to the MLP neural network as shown in table4.1. Offset -0.05 was also added to HID for better results. The thicken operation is applied on the numeral images from MATLAB command (as img= bwmorph(image, ‘thicken’,Inf)). The images are resized into 42x30 and partitioned into 9 sub boxes as shown in Fig3.7

(a). Weight initialization

A 9 weights are initialize to the MLP neural network.

```
W1 = [0.3;0.2;0.4;0.5;0.8;0.4;0.5;0.7;0.1];
W2 = [0.1;0.9;0.7;0.6;0.5;0.1;0.2;0.6;0.4];
WI   = [W1,W2];
WO = [0.5;0.2]
```

No of input layer	1
No of hidden layer	1
No of output layer	1
No neurons of input layer	9
No neurons of hidden layer	2
No neurons of output layer	1
LR	0.02
ϵ	0.0001

Table 4.1 Parameters which are used in experiment

4.2 PROPOSED ALGORITHM RESULTS

The training images of 0 and 7 are shown in Fig4.1 & Fig4.2 respectively. According to proposed algorithm features are extracted from the training images is shown Table 4.2 & Table 4.3 for 0 and 7 respectively. As well as the test images of 0 and 7 are shown in Fig4.3 & Fig4.4 respectively and features are extracted for test images are shown in Table4.4 & Table4.5 for 0 & 7 respectively.

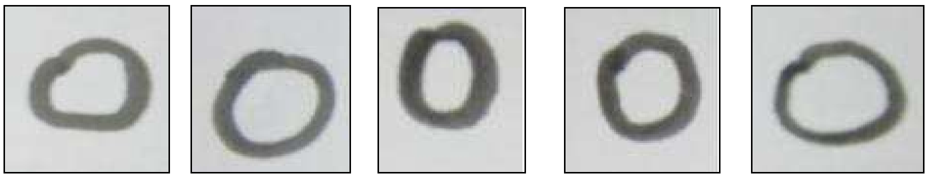


Fig 4.1 Training images of 0 are used in our experiment

0.1858	0.1062	0. 1327
0. 0973	0	0.0619
0. 1681	0. 1062	0. 1416

(a)

0.1360	0.1120	0.1280
0..1280	0	0.1120
0.1440	0.1040	0.1360

(b)

0.1215	0. 1028	0. 1682
0. 0841	0	0. 1308
0. 1495	0. 1028	0. 1402

(c)

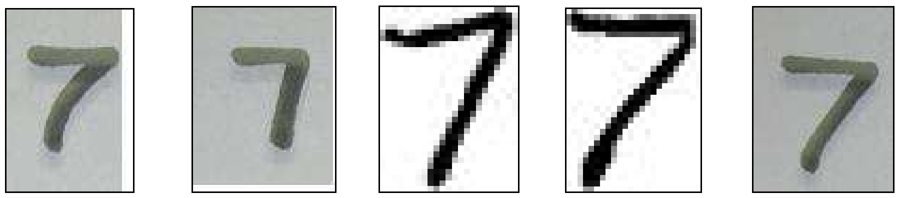
0.1226	0.1132	0.1509
0.1321	0	0.1321
0.1038	0.0943	0.1509

(d)

0. 1613	0. 1075	0. 1398
0. 0645	0	0. 1075
0. 1398	0. 1075	0. 1720

(e)

Table 4.2 A 9 dimension feature of 0 numeral image used in training for figure 4.1(a), (b), (c), (d) and (e)



(f) (g) (h) (i) (j)

Fig 4.2 Training images of 7 are used in our experiment

0.0125	0.1500	0.3750
0.0625	0.2550	0
0.1250	0	0

(f)

0.2065	0.1630	0.2283
0	0.1848	0.0326
0.1630	0.0217	0

(g)

0.1420	0.1111	0.3210
0	0.0926	0.1235
0	0.2099	0

(h)

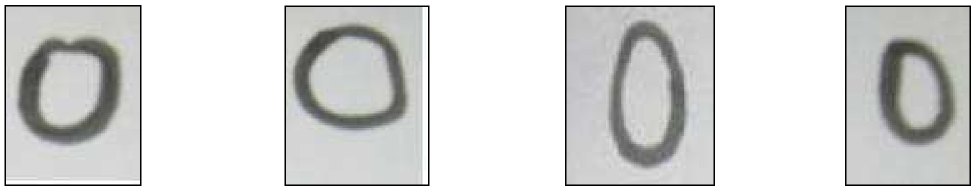
0.1370	0.1438	0.2603
0	0.1575	0.0890
0.1438	0.0685	0

(i)

0.2462	0.1385	0.1538
0	0.2462	0
0.2000	0.0154	0

(j)

Table 4.3 A 9 dimension feature for 7 numeral image used in training for figure 4.2 (f), (g), (h), (i) and (j)



(k) (l) (m) (n)

Fig 4.3 Testing images of 0 are used in our experiment

0.1570	0.1157	0.1570
0.0909	0	0.0909
0.1405	0.1074	0.1405

(k)

0.1293	0.0948	0.1379
0.1207	0	0.1293
0.1638	0.0776	0.1466

(l)

0.1545	0.1000	0.1455
0.1000	0	0.0818
0.1364	0.1000	0.1818

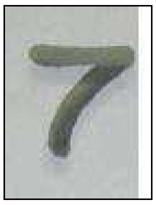
(m)

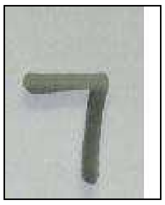
0.1655	0.0828	0.1310
0.0966	0	0.1034
0.1793	0.0897	0.1517

(n)

Table 4.4 Black Pixels count of 0 numeral images in each sub box for test input

(o)


(p)


(q)

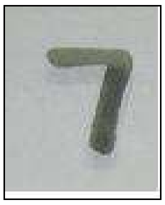

(r)

Fig 4.4 Testing images of 7 are used in our experiment

0.1068	0.1068	0.3981
0.0194	0.2039	0
0.1650	0	0

(q)

0.1429	0.2755	0.2449
0.1122	0.1539	0
0.00714	0	0

(r)

0.1875	0.2411	0.3750
0	0	0.0714
0	0	0.1250

(s)

0.1075	0.1075	0.3226
0	0	0.2366
0	0.0538	0.1720

(t)

Table 4.5 Black Pixels count of numeral image 7 in each sub box for test input

After getting black pixels in the image in each box, the training input image is taken and input weights are initialized for training to MLP neural network in 1st iteration after these weights are updated automatically. For our experiment, we select the numerals 0 and 7 to which we assign the labels +10 and -10 respectively. After getting MLP NN well trained the 4 test inputs are applied for testing our numeral recognition system so that we get 0% and 100% efficiency for 0 and 7 numerals respectively through our proposed method as shown in table 4.6.

Numerals	No of Training inputs	No of Testing inputs	%age recognit ion
0	5	4	80%
7	5	4	100%

Table 4.6 Classification result of 0 Vs 7

Test image	Calculated Label
1	-0.6307
2	0.0648
3	0.6737
4	0.6720

Table 4.7 Test result for 0

Test image	Calculated Label
1	-12.5444
2	-10.5964
3	-9.4794
4	-6.0092

Table 4.8 Test result for 7

As observed from classification results in table4.1, our proposed method is very effective to recognize handwritten numerals. In our experiment, we have achieved classification between two distinct numerals 0 & 7 with successful results. Test results are shown in Table 4.7 & Table 4.8 for 0 and 7 respectively.

CHAPTER 5

MAIN CONCLUSION AND FUTURE SCOPE

In this chapter the final concluding statement along with the future scope in the presented field are discussed.

5.1 CONCLUSION

In this thesis a novel approach for handwritten numeral or characters recognition using MLP neural network classifier and bounding box approach which was implemented successfully. We have obtained maximum efficiency using the proposed technique with a mean correct classification of 90% for distinct numerals. The recognition of handwritten numerals has many applications such as office automation, postal sorting, bank cheque recognition, automatic pin code recognition, collecting data from filled in forms etc. Though there are some commercially available softwares, mainly for printed character recognition of some languages, But the success yet to be extended for hand written characters. Such technique is to facilitate smoother interaction between man and machine. Since handwritten numerals and characters are used in highly sensitive areas like finance and administrations the recognition system should be accurate, fast and easy to implement. The various writing styles of characters introduce large variations. In MLP Neuarl Network classifier the “thicken” operation (MATLAB function) is performed on the captured images from our database and to extract normalized features using the proposed bounding box approach. The major advantage of MLP classifier approach is robustness, easy to implement and relatively high recognition rate. Moreover, this method is insensitive to barbs and gaps that arise at the preprocessing stage. We can emphatically advocate the use of MLP Neural Network for document processing from this study. The automatic recognition of digits on scanned images has wide commercial importance.

5.2 FUTURE WORK

Our future work is dedicated to improving the recognition rate of indistinct unconstrained numerals using improved feature extraction techniques. This approach of recognition of handwritten English numerals by MPL classifier can also be extended to include handwritten characters of English alphabet.

References

[1] A.K.Telele, Sanjay & M.E.Rane “Automatic Recognition and Verification of Handwritten Legal and Courtesy Amounts in English Language Present on Bank Cheques” International Journal of Computer Applications (0975 – 8887) Vol. 21(8) (2011) 35-41.

[2] Hanmandlu, M., & Murthy, O. V. R. “Fuzzy model based recognition of handwritten numerals” Pattern Recognition, 40(6) (2007) 1840–1854.

[3] C. Jou &H.C. Lee “Handwritten numeral recognition based on simplified structural classification and fuzzy memberships” Expert Systems with Applications 36 (2009) 11858–11863.

[4] Sung-Bae Cho “Neural-Network Classifiers for Recognizing Totally Unconstrained Handwritten Numerals” IEEE Transaction on Neural Networks, VOL. 8, NO. 1, Jan 1997.

[5] D. Sharma & D. Gupta “Isolated Handwritten Digit Recognition using Adaptive Unsupervised Incremental Learning Technique” International Journal of Computer Applications (0975 – 8887) Vol. 7– No.4, Sep. 2010.

[6] M.J.K. Singh,R. Dhir & R. Rani “Performance Comparison of Devanagari Handwritten Numerals Recognition” International Journal of Computer Applications (0975 – 8887) VOL. 22(.1), May 2011 1-6.

[7] M. Hanmandlu, M.H.M. Yusof, M. Vamsi Krishna “Off-line signature verification and forgery detection using fuzzy modeling” Pattern Recognition 38 (3) (2005) PP.341–356.

[8] M. Hanmandlu, K. R. Murali Mohan, S. Chakraborty, S. Goyal, and D. R. Choudhury, “Unconstrained handwritten character recognition based on fuzzy logic” Pattern Recognition Vol. 36(3) (2003) 603-623.

[9] J. Cai & Zhi-Qiang Liu “Integration of Structural and Statistical Information for Unconstrained Handwritten Numeral Recognition” IEEE Transaction on Pattern Analysis and Machine Intelligence, VOL. 21(3) (1999) 263-270.

[10] Y. Perwej & A. Chaturvedi “Neural Networks for Handwritten English Alphabet Recognition” International Journal of Computer Applications (0975 – 8887) Vol.20 (7), April 2011 1-5.

[11] A. Malaviya., & L. Peters,”Fuzzy feature description of handwriting patterns” Pattern Recognition, 30(10) (1997) 1591–1604.

[12]. Zhang, P., & Chen, L. (2002). A novel feature extraction method and hybrid tree classification for handwritten numeral recognition. Pattern Recognition Letters, 23(1–3), 45–56.

[13]. Wang, J., & Yan, H. (2000). A hybrid method for unconstrained handwritten numeral recognition by combining structural and neural ‘gas’ classifiers. Pattern Recognition Letters, 21(6–7), 625–635.

[14]. Cao, J., Ahmadi, M., & Shridhar, M. (1995). Recognition of handwritten numerals with multiple feature and multistage classifier. Pattern Recognition, 28(2), 153–160.

[15]. Siy, P., & Chen, C. S. (1974). Fuzzy logic for handwritten numeral character recognition. IEEE Transactions on Systems, Man and Cybernetics, 520–574.

[16]. Malaviya, A., & Peters, L. (1997). Fuzzy feature description of handwriting patterns. Pattern Recognition, 30(10), 1591–1604.

[17]. Hu, J., Yu, D., & Yan, H. (1996). Algorithm for stroke width compensation of handwritten characters. Electronics Letters, 32(24), 2221–2222.

[18]. Mayora-Ibarra, O., & Curatelli, F. (1998). Handwritten digit recognition by means of a holographic associative memory. Expert Systems with Applications, 15(3), 399–403.

[19] S. Khedekar, V. Ramanaprasad, S. Setlur, V. Govindaraju, "Text—image separation in devanagari documents", Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR'03), Edinburgh, Scotland, 3–6 August, 2003, pp. 1265–1269.

[20] R. Bajaj, L. Dey and S. Chaudhury, "Devnagari numeral recognition by combining decision of multiple connectionist classifiers", Sadhana 27 (2002) (1), pp. 59–72.

[21] S. Antanani, L. Agnihotri, "Gujarati character recognition", in: Proceedings of Fifth IEEE International Conference on Document Analysis and Recognition (ICDAR'99), Bangalore, India, 20–22 September, 1999, pp. 418–421.

[22] V. Bansal, R.M.K. Sinha, A Devanagari OCR and a brief review of OCR research for Indian scripts, Proceedings of STRANS01, IIT Kanpur, Kanpur, India, 2001.

[23] B.B. Chaudhuri, U. Pal, An OCR system to read two Indian language scripts: Bangla and Devanagari (Hindi), in: Proceedings of Fourth IEEE International Conference on Document Analysis and Recognition (ICDAR'97), Ulm, Germany, August 18–20, 1997, pp. 1011–1015.

[24] S.S. Marwah, S.K. Mullick, R.M.K. Sinha, "Character recognition of devanagari characters using a hierarchial binary decision tree classifier", IEEE International Conference on Systems, Man and Cybernetics, October 1994.

[25] R.M.K. Sinha and H.N. Mahabala, "Machine recognition of Devanagari script", IEEE Trans. Syst. Man Cybern. 9 (1979) (8), pp. 435–441.

[26] I.K. Sethi and B. Chatterjee, “Machine recognition of handprinted Devanagari numerals”, J. Inst. Electron. Telecommun. Eng. 22 (1976), pp. 532–535.

[27] I.K. Sethi and B. Chatterjee, “Machine recognition of constrained handprinted Devanagari”, Pattern Recognition 9 (1977) (2), pp. 69–75.

[28] J.C. Sant, S.K. Mullick, “Handwritten Devanagari script recognition using CTNNSE algorithm”, International Conference on Application of Information Technology in South Asia language—AKSHARA’94, New Delhi, India, 25–26 February.

[29] A. Elnagar and S. Harous, “Recognition of handwritten Hindu numerals using structural descriptors”, J. Exp. Theor. Artif. Intell. 15 (2003) (3), pp. 299–314.

[30] Y. Suganuma, “Learning structures of visual patterns from single instances”, Artif. Intell. 50 (1991) (1), pp. 1–36.

[31] S. Kumar, C. Singh, “A study of zernike moments and its use in Devanagari handwritten character recognition”, Proceedings of the International Conference on Cognition and Recognition (ICCR’05), Mysore, India, 22–23 December, 2005, pp. 514–520.

[32] Y.B. Mahdy, E. Moumen, T. El-Melegy, “Encoding patterns for efficient classification by nearest neighbor classifiers and neural networks with application to handwritten Hindi numeral recognition”, Proceedings of Third International Conference on Signal Processing (ICSP’96), Beijing, China, 14–18 October, 1996, pp. 1362–1365.

[33] H.Y.Y. Sanossian, “Feature extraction technique for Hindi numerals”, Proceedings of the IEEE Workshop on Neural Networks for Signal Processing VIII, Cambridge, U.K, 31 August–3 September, 1998, pp. 524–530.

[34] P. Siy and C.S. Chen, Fuzzy logic for handwritten numeral character recognition, IEEE Trans. Syst. Man Cybern. 4 (1974), pp. 570–575.

[35] G. Baptista and K.M. Kulkarni, A high accuracy algorithm for the recognition of handwritten numerals, Pattern Recognition 21 (1998) (4), pp. 287–291.

[36] T.M. Ha and H. Bunke, "Offline handwritten numeral recognition by perturbation method", IEEE Trans. Pattern Anal. Mach. Intell. 19 (1997) (5), pp. 535–539.

[37] Z. Chi, J. Wu and H. Yan, "Handwritten numeral recognition using self organizing feature maps and fuzzy rules", Pattern Recognition 28 (1995) (1), pp. 59–66.

[38] Z. Chi and H. Yan, "ID-3 derived fuzzy rules and optimized defuzzification for handwritten numeral recognition", IEEE Trans. Fuzzy Systems 4 (2000) (1), pp. 24–31.

[39] S.W. Lee, "Offline recognition of totally unconstrained handwritten numerals using multi-layer cluster neural networks", IEEE Trans. Pattern Anal. Mach. Intell. 18 (1996) (6), pp. 648–652.

[40] S. Singh, Hewitt, "Cursive digit and character recognition on CEDAR database", Proceedings 15th International Conference on Pattern Recognition, Barcelona, vol. 2, IEEE Press, New York, 3–8 September, 2000, pp. 569–572.

[41] M. Blumenstein, Griffith, B. Verma, Basli, "A novel feature extraction technique for the recognition of segmented handwritten characters", Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR' 03), Edinburgh, Scotland, 3–6 August, 2003, pp. 137–141.

[42] G. Dimauro, S. Impedovo, R. Modugno, G. Pirlo, "Numeral recognition by weighting local decisions", Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR'03), Edinburgh, Scotland, 3–6 August, 2003, pp. 1070–1074.

[43] C.L. Liu, K. Nakashima, H. Sako and H. Fujisawa, "Handwritten digit recognition: benchmarking of state-of-the-art techniques", Pattern Recognition 36 (2003) (10), pp. 2271–2285.

[44] I.I.S. Oh, J.S. Lee and C.Y. Suen, “Analysis of class separation and combination of class-dependent features for handwriting recognition”, IEEE Trans. Pattern Anal. Mach. Intell. 21 (1999) (10), pp. 1089–1094.

[45] Z.H. Cai and Z.Q. Liu, “Integration of structural and statistical information for unconstrained handwritten numeral recognition”, IEEE Trans. Pattern Anal. Mach. Intell. 21 (1999) (3), pp. 263–270.

[46] M. Hanmandlu, K.R. Murali Mohan, H. Kumar, “Neural-based handwritten character recognition”, in: Proceedings of Fifth IEEE International Conference on Document Analysis and Recognition (ICDAR’99), Bangalore, India, 20–22 September, 1999, pp. 241–244.