International Journal of Soft Computing and Engineering (IJSCE)
Volume-1, Issue-1, March 2011

Handwritten Multiscript Numeral Recognition using Artificial Neural Networks

Stuti Asthana, Farha Haneef, Rakesh K Bhujade

***Abstract*—Handwritten Numeral recognition plays a vital role in postal automation services especially in countries like India where multiple languages and scripts are used .Because of intermixing of these languages; it is very difficult to understand the script in which the pin code is written. Objective of this paper is to resolve this problem through Multilayer feed-forward back-propagation algorithm using two hidden layer. This work has been tested on five different popular Indian scripts namely Devnagri, English, Urdu, Tamil and Telugu. Network was trained to learn its behavior by adjusting the connection strengths on every iteration. The resultant of each presented training pattern was calculated to identify the minima on the error surface for each training pattern. Experiments were performed on samples by using two hidden layers and the results revealed that as the number of hidden layers is increased, more accuracy is achieved in large number of epochs**

***Index Terms*— Numeral Recognition, Artificial Neural Network, Supervised learning, Back Propagation Algorithm.**

I. INTRODUCTION

Numeral Recognition refers to the process of translating images of hand-written, typewritten, or printed digits into a format understood by user for the purpose of editing, indexing/searching, and a reduction in storage size. The Numeral Recognition process is complicated by noisy inputs, image distortion, and differences between typefaces, sizes, and fonts. Numeral Recognition becomes more complex when multiple scripts were used during pin code or phone number writing because it is not possible for a single person to have an idea of every scripting of numeric digits. For solving this problem, artificial neural networks are commonly used. The software developed in this project converts numeric image written in five different scripts (Devnagri, English, Urdu, Tamil and Telugu) in to English so that it can be understood easily. Previously, different methods were proposed [1],[2],[3] for identifying multiscript numerals but still accuracy percentage is not satisfactory. The software used in this project uses two hidden layer so as to get more precise results. Experiments were performed on different multi scripting samples and 96.53% accuracy is achieved.

Manuscript received Feb 22, 2011.

Stuti Asthana is M.Tech Scholar from RKDF Institute of Science and Technology, Bhopal, India , e-mail: stutiasthana@gmail.com.
Farha Haneef is M.Tech Coordinator in RKDF Institute of Science and Technology, Bhopal, India.e-mail: haneef_farha@yahoo.com.
Rakesh K Bhujade is with Technocrats Institute of Technology, Bhopal , India., e-mail: rakesh.bhujade@ gmail.com.

At present postal sorting machines are available in several countries like USA, UK, Canada, Japan, France, Germany etc. There are only a few works on Indian postal system [4], [5] and at present no postal automation machine is available for India

Indian pin-code (postal code) is a six-digit number and system development towards Indian postal automation is more difficult and challenging than that of other country because of its multi-lingual and multi-script behaviour. In India there are 22 official languages and 11 scripts are used to write these languages. Fig. 1 shows a system that can substitute for the manual sorting letters.

.

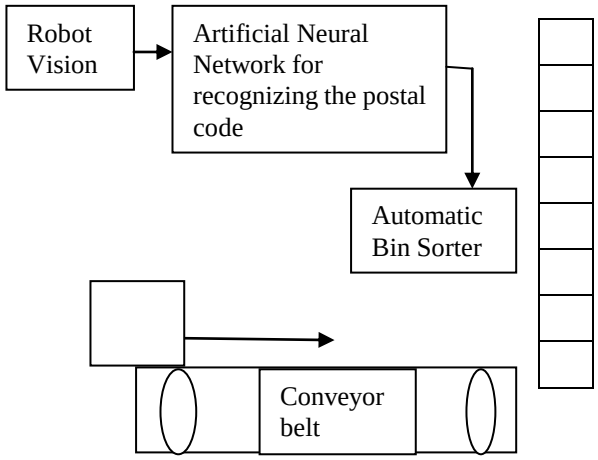

Fig 1: Automatic letter sorting system on the basis of pin code

Rest of the paper is organized as follows. Overview of Devnagri, English, Urdu, Tamil and Telugu scripts in section 2, phases in recognition of pin code numerals in section 3,and experimental results and comparative study in section 4 .

II. OVERVIEW OF TAMIL, TELUGU, URDU, DEVNAGRI AND ENGLISH SCRIPTS REVIEW STAGE

Tamil is a Dravidian language, and one of the oldest in the world. It is the official language of the Indian state of Tamil Nadu; it also has an official status in Sri Lanka, Malaysia and Singapore. The Tamil script has 10 numerals, 12 vowels, 18 consonants and five grantha letters.

Telugu is a Dravidian language and has the third most popular script in India. It is the official language of the Indian state of Andhra Pradesh. There are 10 numerals, 18 vowels, 36 consonants, and three dual symbols. Urdu digits mostly followed in Pakistan Generally, in Urdu both Urdu numerals

and old Arabic (used for English, Latin) are followed but not at the same time. A sample of Tamil, Telugu, Urdu, Devanagari and English handwritten numerals from the dataset are shown from figures 2 to 6 respectively.

Fig 2:Samples of handwritten Tamil numerals from 0 to 9

Fig 3: Samples of handwritten Telugu numerals from 0 to 9

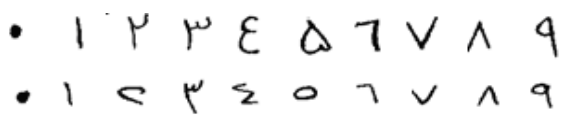

Fig 4: Samples of handwritten Urdu numerals from 0 to 9

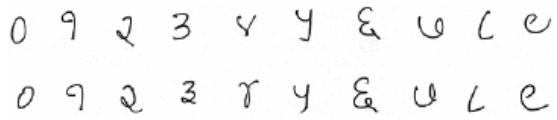

Fig 5: Samples of handwritten Devnagri numerals from 0 to 9

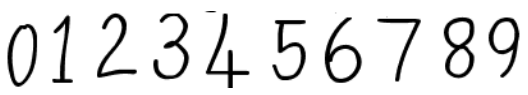

Fig 6: Samples of English numerals from 0 to 9

III. PROPOSED METHODOLOGY FOR RECOGNITION OF HANDWRITTEN PIN CODE NUMERALS

The original postcard pin code is scanned into the computer and saved as an image. This software breaks the image into sub-images, each containing a single character. The sub-images are then translated from an image format into a binary format, where each 0 and 1 represents an individual pixel of the sub-image. The binary data is then fed into a neural network that has been trained to make the association between the character image data and a numeric value that corresponds to the character. The output from the neural network is then translated into text and saved as a file.

Initially the scanned pin code number is encapsulated in a minimum possible rectangle called glyphs. When analyzed, these glyphs are made of black and white pixels comprising the shape of the numerals [6]. Here, black pixel denotes 1 and the white background denotes 0.These combination of zero and one is then fetched as a input data in the neural network where the activation function is implemented on every neurons, so as to process input data. Finally, a Unicode is generated by the output neurons which resemble the designated character.

A. Algorithm Description

The MLP learning algorithm for two hidden layer is outlined below.

1. Initialize the network, with all weights set to random numbers between –1 and +1.
2. Present the first training pattern, and obtain the output.
3. Compare the network output with the target output.
4. Propagate the error backwards.

(a) Correct the output layer of weights using the following formula.

$$W_{h2o} = W_{h2o} + (\eta \delta_o O_{h2})$$

where W_{h2o} is the weight connecting hidden unit h_2 with output unit O, η is the learning rate, O_{h2} is the output at hidden unit h_2. δ_o is error given by the following.

$$\delta_o = O_o(1- O_o)(T_o - O_o)$$

(b) Correct the input weights using the following formula.

$$W_{h1h2} = W_{h1h2} + (\eta\, \delta_{h2} O_{h1})$$

where W_{h1h2} is the weight connecting two hidden layers with , Oh1 is the input at node of the second hidden layer, η is the learning rate. δ_{h2} is calculated as follows.

$$\delta_{h2} = O_{h2}(1- O_{h2})\Sigma\, (\delta_o W_{h2o})$$

(b) Correct the input weights using the following formula.

$$W_{ih1} = W_{ih1} + (\eta\, \delta_{h1} O_i)$$

Where W_{ih1} is the weight connecting node i of the input layer with node of the first hidden layer,

O_i is the input at node of the hidden layer, η is the learning rate. δh_1 is calculated as follows.

$$\delta_{h1} = O_{h1}(1- O_{h1})\, \Sigma\, (\delta_o W_{h1h2})$$

5. Calculate the error, by taking the average difference between the target and the output vector.
6. Repeat from 2 for each pattern in the training set to complete one epoch.
7. Repeat from step 2 for a specified number of epochs, or until the error ceases to change.

B. Activation Function

If a multilayer perceptron consists of 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 any number of layers can be reduced to the standard two-layer input-output model. What makes a multilayer perceptron different is that each neuron uses a nonlinear activation function which was

developed to model the frequency of action potentials, or firing, of biological neurons in the brain. This function is modeled in several ways, but must always be normalizable and differentiable [8],[9].

The two main activation functions used in current applications are both sigmoid, and more specialized activation functions include radial basis functions which are used in another class of supervised neural network models.

Most common activation functions are the logistic and hyperbolic tangent sigmoid functions.

The project uses hyperbolic tangent function:

$$f(x)=\{2/(1+e^{-\lambda x})\}\ \&\ \text{derivative: } f'(x)=f(x)(1-f(x))$$

IV. Essential Components for Implementation

1. Formation of the network and weight initialization routine
2. Pixel analysis of images for symbol detection
3. Loading routines for training input images and corresponding desired output characters in special files named character trainer sets (*.cts)
4. Loading and saving routines for trained network (weight values)
5. Character to binary Unicode and vice versa conversion routines
6. Error, output and weight calculation routines

A. Network Formation

The MLP Network implemented for the purpose of this project is composed of 4 layers, one input, two hidden and one output.

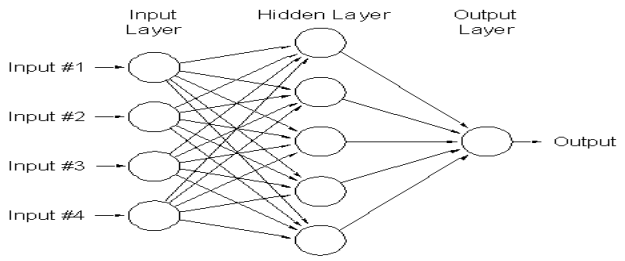

Fig 7: Typical feed forward network with single hidden layer

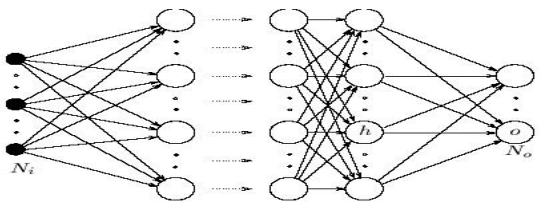

Fig 8: Typical feed forward network with multiple hidden layers

The input layer constitutes of 150 neurons which receive pixel binary data from a 10x15 symbol pixel matrix.

The size of this matrix was decided taking into consideration the average height and width of character image that can be mapped without introducing any significant pixel noise.

The two hidden layers constitute of 250 neurons each whose number is decided on the basis of optimal results on a trial and error basis.

The output layer is composed of 16 neurons corresponding to the 16-bits of Unicode encoding.

To initialize the weights a random function was used to assign an initial random number which lies between two preset integers named weight_bias. The weight bias is selected from trial and error observation to correspond to average weights for quick convergence.

B. Determining Pincode Line

1. Start at the first x and first y pixel of the image pixel (0, 0). Set number of lines to 0
2. Scan up to the width of the image on the same y-component of the image
   a. if a black pixel is detected register y as top of the first line
   b. if not continue to the next pixel
   c. if no black pixel found up to the width increment y and reset x to scan the next horizontal line
3. Start at the top of the line found and first x-component pixel(0,line_top)
4. Scan up to the width of the image on the same y-component of the image
   a. If no black pixel is detected register y-1 as bottom of the first line. Increment number of lines
   b. if a black pixel is detected increment y and reset x to scan the next horizontal line
5. Start below the bottom of the last line found and repeat steps 1-4 to detect subsequent lines.
6. If bottom of image (image height) is reached stop.

C. Detecting individual symbol

Detection of individual symbols involves scanning character lines for orthogonally separable images composed of black pixels [6, 7].

1.Start at the first character line top and first x-component
2.Scan up to image width on the same y-component
a. if black pixel is detected register y as top of the first line
b.if not continue to the next pixel
3.Start at the top of the character found and first x-component, pixel(0,character_top)
4.Scan up to the line bottom on the same x-component
a.if black pixel found register x as the left of the symbol
b.if not continue to the next pixel
c.if no black pixels are found increment x and reset y to scan the next vertical line
5.Start at the left of the symbol found and top of the current line, pixel(character_left, line_top)
6.Scan up to the width of the image on the same x-component
a.if no black characters are found register x-1 as right of the symbol
b.if a black pixel is found increment x and reset y to scan the next vertical line
7. Start at the bottom of the current line and left of the symbol, pixel(character_left,line_bottom)
8. Scan up to the right of the character on the same y-component

a. If a black pixel is found register y as the bottom of the character
b. If no black pixels are found decrement y and reset x to scan the next vertical line

D. Symbol image matrix mapping

The next step is to map the symbol image into a corresponding two dimensional binary matrix. An important issue to consider here will be deciding the size of the matrix. If all the pixels of the symbol are mapped into the matrix, one would definitely be able to acquire all the distinguishing pixel features of the symbol and minimize overlap with other symbols[10].However this strategy would imply maintaining and processing a very large matrix (up to 1500 elements for a 100x150 pixel image). Hence a reasonable tradeoff is needed in order to minimize processing time which will not significantly affect the separability of the patterns. The project employed a sampling strategy which would map the symbol image into a 10x15 binary matrix with only 150 elements. Since the height and width of individual images vary, an adaptive sampling algorithm was implemented.

E. Training

Once the network has been initialized and the training input space prepared the network is ready to be trained. Some issues that need to be addressed upon training the network are:

What should be used for the values of:

Learning rate ,Sigmoid slope, and Weight bias.

For the purpose of this project the parameters used are:

- Learning rate =150
- Sigmoid Slope = 0.014
- Number of Epochs = 300
- Mean error threshold value = 0.0002

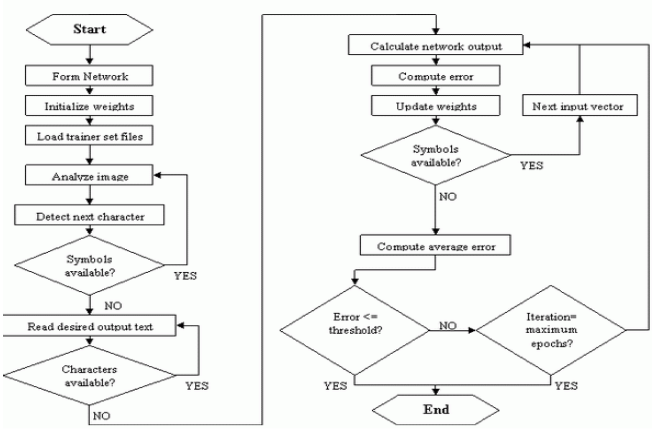

Fig 9: Flow chart for symbol recognition

V. Result Analysis

Experiments were performed on five different samples having different handwriting. Two different approaches were used, one with single hidden layer and another with two hidden layer. Rest of all the parameters is kept same. We obtained 96.53% average recognition rate using double hidden layer and 94% using single hidden layer for five scripts i.e. Devnagri, English, Urdu, Tamil and Telugu.

The system is simulated using feed forward neural network in which the characters are resized into m x n matrixes.

The network having one hidden layer used for experiment-1 and the network having two hidden layer used for experiment-2, their outcomes are as follows:

Table 1 : Result Analysis

S. no	SAMPLE NO.	Hidden Layer =1	Hidden Layer =2
		Accuracy	Accuracy
1.	Sample1(Telugu+ Tamil+English)	64.3%	98.6%
2.	Sample2(Tamil + English)	75.4%	92.2%
3.	Sample3(Devnagri+Tamil)	80.2%	94.7%
4.	Sample4(Devnagri+ Urdu)	71.9%	98.4%
5.	Sample5(Telugu+English)	77.4%	93.3%
6.	Sample6(Urdu+Devnagri)	79.4%	97.0%
7.	Sample7(English+Devnagri)	84%	98%
8.	Sample8(Tamil+Telugu)	84.1%	98.1%
9.	Sample9(Urdu+English)	73.3%	96.3%
10	Sample10(English+Devnagri+Urdu)	84.7%	98.7%

A. Error estimation

For both experiments with one and two hidden layers, it is evident that the error is reduced when two hidden layers are used in the network. In other words, we can say that with the increase in the number of hidden layers, there is an increase in probability of converging the network before the number of training epochs reaches it maximum allowed count.

B. Number of Epochs

The results of the learning process of the network in terms of the number of training iterations, small number of epochs are sufficient to train a network when we use one hidden layer. As the number of hidden layers is made two, the number of epochs required to train the network also increases.

The proposed method for the typewritten character recognition using the back propagation approach, showed the remarkable enhancement in the performance when two hidden layers were used.

The recognition accuracy is best in experiments where MLP with two hidden layers was used.

Number of hidden layers is proportional to the number of epochs. This means that as the number of hidden layers is increased; the training process of the network slows down because of the increase in the number of epochs.

If the accuracy of the results is a critical factor for an character recognition application, then the network having many hidden layers should be used but if training time is a critical factor then the network having single hidden layer (with sufficient number of hidden units) should be used. Nevertheless, more work needs to be done especially on the test for handwritten characters. The proposed work can be carried out to recognize typewritten characters.

VI . References

[1] R. Plamondon and S. N. Srihari, "On-line and off- line handwritten character recognition: A comprehensive survey", IEEE. Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 1, pp. 63-84, 2000.

[2] Liana M. Lorigo and Venu Govindaraju, "Offline Arabic handwriting recognition: A survey", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 5, pp. 712-724, 2006.
[3] G.G. Rajaput and Mallikarjun Hangarge, "Recognition of isolated handwritten Kannada numerals based on image fusion method: ", PReMI07, LNCS.4815, pp.153-160, 2007.
[4] U. Pal, K. Roy and F. Kimura, "A Lexicon driven method for unconstrained Bangla handwritten word recognition", In Proc. 10th IWFHR, pp. 601-606, 2006.
[5] K. Roy, "On the development of an optical character recognition system for Indian postal automation", Ph.D. Thesis, Jadavpur University, 2008.
[6] Muller et al. An Introduction to Backpropagation Learning Algorithms, IEEE Transactions on Neural Networks, Vol. 12, No. 2, March 2009
[7] Dong Xiao Ni,Proceedings of Students/Faculty Research Day,CSIS , Pace University,.Application of Neural Network to character recognition, May 2007.
[8] Reza Gharoie Ahangar ,Farajpur Ahangar," Farsi Character Recognition using Artificial Neural Network" IJCSIS International Journal of Computer Science and Information Security,Vol.4, no.1&2,2009
[9] Le Cun, Y., Bottou, L., and Ha_ner, P. (2008). Gradient-based learning applied to document recognition, Proceedings of the IEEE, 86(11), 2278-2324.
[10] Altun, H., Curtis,KM,1997An improved neural network learning. Proceeding of the Fourth IEEE International Conference on Electronics, Circuits and Systems, VOL.1,Page29-33.