Advances in Computer Science and Information Technology (ACSIT)
Print ISSN: 2393-9907; Online ISSN: 2393-9915; Volume 2, Number 10; April-June, 2015 pp. 72-77

http://www.krishisanskriti.org/acsit.html

Handwritten Character Recognition using Neural Network

Sugam[1] and Asha Sohal[2]

[1]Master of Technology (Computer Science and Engineering) from K.R. Mangalam University Sohna Road Gurgaon
[2]K.R.Mangalam University (Sohna Road Gurgaon)
E-mail: [1]sugam35yadav@gmail.com, [2]asha.shrawat@gmail.com

Abstract—*Hand written character recognition is a challenging task in the field of research on image processing, Artificial Intelligence as well as machine vision since the handwriting varies from person to person. It has numerous application of handwritten text in reading bank cheques, zip code recognition and in removing the problem of handling document manually has, made it necessary to acquire digitally formatted data. In this paper we focus on recognition of English character in a given scan text document with the help of neural network. The first step is image acquisition which acquires the scanned image, rendering image suitable for segmentation where image is decompose into sub image. Feature extraction improves recognition rate and misclassification. We use Character Extraction and Edge detection algorithm for training the neural network to classify and recognize hand written characters. A Multilayer feed forward neural network is created and trained through Back-Propagation algorithm. After the training, testing is done to match the pattern with test data. Result for various convergence objective of neural network are obtained and analyzed.*

Keyword*: Artificial Intelligence, English Character, Neural Network, Back-Propagation, Multilayer feed-forward.*

1. INTRODUCTION

Character recognition is one of the most successful applications of neural network technology. In character recognition, printed documents are transformed into ASCII files for the purpose of editing, compact storage, fast retrieval through the use of computer[]. The recognition of character in a document becomes difficult due to noise, distortion, various character fonts and size, writing styles as handwriting of different persons is different. Handwritten character recognition can be differentiated into two categories i.e. Online Handwritten character recognition and Offline Handwritten character Recognition. On-line handwritten character recognition deals with automatic conversion of characters that may be written using a special digitizer; tablet PC in which a sensor picks up the pen-tip movements and also the pen-up/pen-down switching. Off-line handwritten character recognition deals with a data set, which is obtained from a scanned handwritten document.

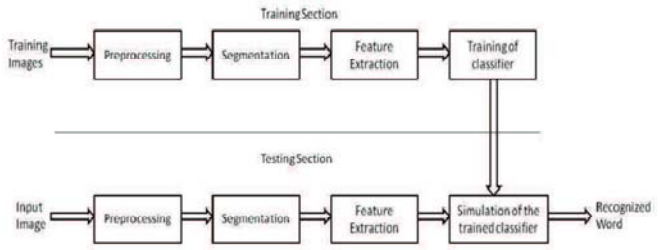

Fig. 1: Generic Character Recognition Model

The recognition of characters from scanned images of documents has been a problem that has received much attention in the fields of image processing, pattern recognition and artificial intelligence. Classical methods in pattern recognition do not as such suffice for the recognition of visual characters due to the following reasons:

1. The 'same' characters differ in sizes, shapes and styles from person to person and even from time to time with the same person.
2. Like any image, visual characters are subject to spoilage due to noise.
3. There are no hard-and-fast rules that define the appearance of a visual character. Hence rules need to be heuristically deduced from samples. As such, the human system of vision is excellent in the sense of the following qualities**:**

I. The human brain is adaptive to minor changes and errors in visual patterns. Thus we are able to read the handwritings of many people despite different styles of writing.

II. The human vision system learns from experience, Hence we are able to grasp newer styles and scripts with amazingly high speed.

III. The human vision system is immune to most variations of size, aspect ratio, color, location and orientation of visual characters.

In contrast to limitations of classical computing, Artificial Neural Networks (ANNs), that were first developed in the mid 1900's serve for the emulation of human thinking in computation to a meager, yet appreciable extent.

2. ARTIFICIAL NEURAL NETWORK

Pattern recognition is extremely difficult to automate. Animals recognize various objects and make sense out of large amount of visual information, apparently requiring very little effort. Simulating the task performed by animals to recognize to the extent allowed by physical limitations will be enormously profitable for the system. This necessitates study and simulation of Artificial Neural Network. In Neural Network, each node perform some simple computation and each connection conveys a signal from one node to another labeled by a number called the "connection strength" or weight indicating the extent to which signal is amplified or diminished by the connection.

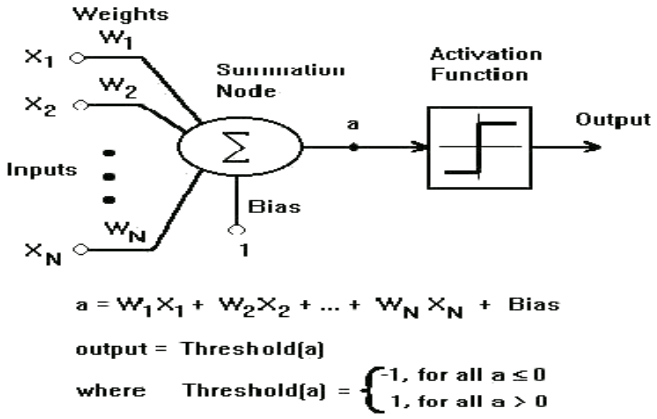

Fig. 2.1: Neural Network Model

Different choices for weight results in different functions are being evaluated by the network. If in a given network whose weight are initial random and given that we know the task to be accomplished by the network , a learning algorithm must be used to determine the values of the weight that will achieve the desired task. Learning Algorithm qualifies the computing system to be called Artificial Neural Network[2]. The node function was predetermined to apply specific function on inputs imposing a fundamental limitation on the capabilities of the network.

Typical pattern recognition systems are designed using two pass. The first pass is a feature extractor that finds features within the data which are specific to the task being solved (e.g. finding bars of pixels within an image for character recognition). The second pass is the classifier, which is more general purpose and can be trained using a neural network and sample data sets. Clearly, the feature extractor typically requires the most design effort, since it usually must be hand-crafted based on what the application is trying to achieve.

One of the main contributions of neural networks to pattern recognition has been to provide an alternative to this design: properly designed multi-layer networks can learn complex mappings in high-dimensional spaces without requiring complicated hand-crafted feature extractors. Thus, rather than building complex feature detection algorithms, this paper focuses on implementing a standard back-propagation neural network. It also encapsulates the Preprocessing that is required for effective.

2.1 Back-Propagation

Back-propagation was created by generalizing the Widrow-Hoff learning rule to multiple-layer networks and nonlinear differentiable transfer functions. Input vectors and the corresponding target vectors are used to train a network until it can approximate a function, associate input vectors with specific output vectors, or classify input vectors in an appropriate way as defined by you. Networks with biases, a sigmoid layer, and a linear output layer are capable of approximating any function with a finite number of discontinuities.

3. THE PROPOSED MODEL

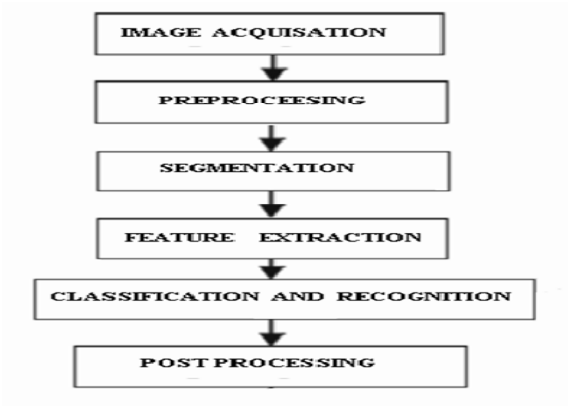

Fig. 3.1: General Model

3.1 Image Acquisition

In Image acquisition, the recognition system acquires a scanned image as an input image. The image should have a specific format such as JPEG, BMP, etc. This image is acquired through a scanner, digital camera or any other suitable digital input device [6, 7].

3.2 Pre-Processing

The pre-processing is a series of operations performed on scanned input image. It essentially enhances the image rendering it suitable for segmentation. The role of pre-processing is to segment the interesting pattern from the background. Generally, noise filtering, smoothing and normalization should be done in this step. The pre-processing also defines a compact representation of the pattern.

Binarization process converts a gray scale image into a binary image. Dilation of edges in the binarized image is done using sobel technique [10].

3.2.1 Character Extraction Algorithm

1. Create a Traverse List: - List of pixels which have been already traversed. This listis initially empty.
2. Scan row Pixel-by-Pixel.
3. Whenever we get a black pixel check whether the pixel is already in the traverse list, if it is simply ignore and move on else apply Edge detection Algorithm.
4. Add the List of Pixels returned by Edgedetection Algorithm to Traverse List.
5. Continue the steps 2 - 5 for all rows.

3.2.2 Edge Detection Algorithm

The Edge Detection Algorithm has a list called traverse list. It is the list of pixel already traversed by the algorithm The Edge Detection Algorithm has a list called traverse list. It is the list of pixel already traversed by the algorithm.

EdgeDetection(x,y,TraverseList);

1) Add the current pixel to TraverseList. The current

position of pixel is (x,y).

2) NewTraverseList= TraverseList + current position

(x,y).

If pixel at (x-1,y-1) then

Check if it is not in TraverseList.

Edgedetection(x-1,y-1,NewTraverseList);

endif

If pixel at (x-1,y) then

Check if it is not in TraverseList.

Edgedetection(x-1,y,NewTraverseList);

endif

If pixel at (x-1,y) then

Check if it is not in TraverseList.

Edgedetection(x-1,y+1,NewTraverseList);

endif

If pixel at (x,y-1) then

Check if it is not in TraverseList.

Edgedetection(x,y-1,NewTraverseList);

Endif

If pixel at (x,y+1) then

Check if it is not in TraverseList.

Edgedetection(x,y+1,NewTraverseList);

Endif

If pixel at (x+1,y-1) then

Check if it is not in TraverseList.

Edgedetection(x+1,y-1,NewTraverseList);

endif

If pixel at (x+1,y) then

Check if it is not in TraverseList.

Edgedetection(x+1,y,NewTraverseList);

endif

If pixel at (x+1,y+1) then

Check if it is not in TraverseList.

Edgedetection(x+1,y+1,NewTraverseList);

endif

3) return;

The EdgeDetection algorithm terminates when it has

covered all the pixels of the character as every pixel's position would be in TraverseList so any further call to EdgeDetection is prevented.

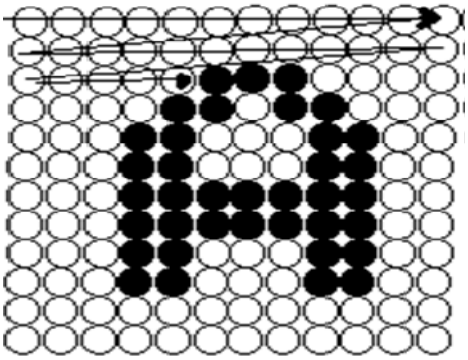

Fig. 3.2.1: Shows the traversing each scan lines.

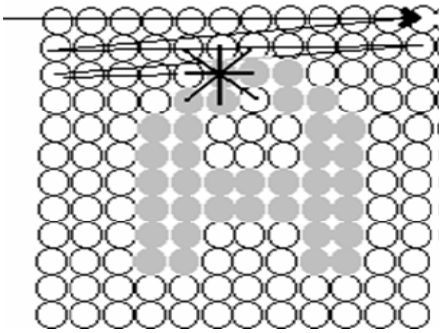

Fig. 3.2.2: Shows the respective calls made to the all 8-neighbouring pixels.

3.2 Segmentation

In the segmentation stage, an image of sequence of characters is decomposed into sub-images of individual character [9]. In the proposed system, the pre-processed input image is segmented into isolated characters by assigning a number to each character using a labelling process. This labelling provides information about number of characters in the image. Each individual character is uniformly resized into pixels.

3.4 Feature Extraction

In this stage, the features of the characters that are crucial for classifying them at recognition stage are extracted. This is an important stage as its effective functioning improves the recognition rate and reduces the misclassification. Diagonal feature extraction scheme for recognizing off-line handwritten characters is proposed in this work [8, 9]. Every character image is divided into equal zones, each of size 10x10 pixels. The features are extracted from each zone pixels by moving along the diagonals of its respective 10x10 pixels.

3.5 Classification and Recognition

The classification stage is the decision making part of the recognition system [3]. A feed forward back propagation neural network is used in this work for classifying and recognizing the handwritten characters. The pixels derived from the resized character in the segmentation stage form the input to the classifier. The neural classifier consists of two hidden layers besides an input layer and an output layer. The total number of neurons in the output layer is 26 as the proposed system is designed to recognize English alphabets [6].

3.5.1 Neural Network Design

For training and simulating purposes we have scanned certain documents. We have 2 types of documents train documents and test documents.

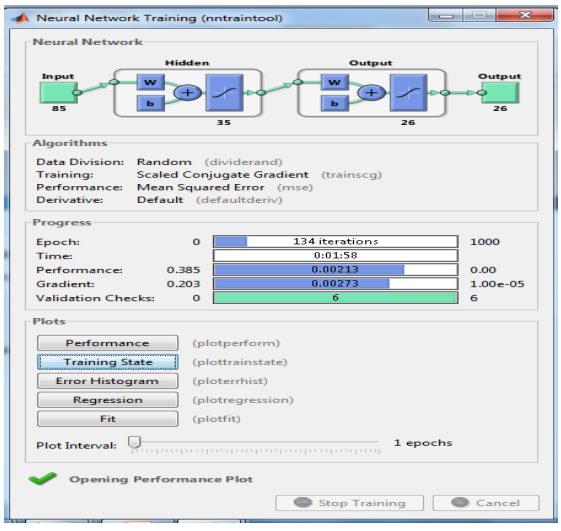

Fig. 3.5.1: Neural Network Model

The train documents are the images of the documents which we want to use for training. Similarly test documents are the images of documents which we want to use for test. According to the characters in the documents we train the neural network and apply the test documents. We have different Models of Neural Network. Hence we record certain parameters like training time, accuracy etc. to find the effectiveness of the Neural Network. We have selected an image size of 10 X 10 as an input to the Neural Network. Hence we have taken a neural network that has 100 inputs. We are performing the test on only Capital characters so the outputs of the Neural Networks are 26. The no. of nodes of input layer are 100 and the no. of node of output layer are 26. The no. of hidden layer and the size of hidden layer vary.

3.6 Post- processing

Post-processing stage is the final stage of the proposed recognition system. It prints the corresponding recognized characters in the structured text form by calculating equivalent ASCII value using recognition index of the test samples.

4. DESIGN AND IMPLEMENTATION

Initially we are making the Algorithm of Character Extraction. We are using MATLAB as tool for implementing the algorithm. Then we design neural network, we need to have a Neural Network that would give the optimum results [11]. There is no specific way of finding the correct model of Neural Network. It could only be found by trial and error method. Take different models of Neural Net- work, train it and note the output accuracy. There are basically two main phases in our Paper: Pre-processing and Character Recognition. In first phase we have are preprocessing the given scanned document for separating the Characters from it and normalizing each characters. Initially we specify an input image file, which is opened for reading and preprocessing. The image would be in RGB format (usually) so we convert it into binary format. To do this, it converts the input image to grayscale format (if it is not already an intensity image), and then uses threshold to convert this grayscale image to binary *i.e.* all the pixels above certain threshold as 1 and below it as 0. Firstly we needed a method to extract a given character from the document. For this purpose we modified the graphics 8-way connected algorithm (which we call as Edge Detection).

5. TEST RESULT

This section shows some implementation results. The training variables involved in the tests were: the number of cycles, the size of the hidden layer, and the number of hidden layer. The dataset consisted of A-Z typed characters of different size and type. Thus the input layer consisted of more than neurons, and the output layer 26 neurons (one for each character). Ideally, we'd like our training and testing data to consist of thousands of samples, but this not feasible since this data was created from scratch.

Table 5.1: Model 1

Epochs	Hiddenlayers	Config	Classification%
43	10	85-10-26	27.8
104	20	85-20-26	77.4
148	30	85-30-26	93.8
172	35	85-35-26	94.3
117	39	85-39-26	93.3
53	45	85-45-26	12.5
60	50	85-50-26	83.7
76	55	85-55-26	78.3
71	60	85-60-26	18
110	65	85-65-26	49.8
112	70	85-70-26	49.2

Table5.2: Model 2

Epochs	Hidden layers	Config	Classification %
47	10	108-10-26	10
189	20	108-20-26	87.4
144	30	108-30-26	82
137	35	108-35-26	82.2
148	39	108-39-26	94.5
109	45	108-45-26	76.8
113	50	108-50-26	86.6
98	55	108-55-26	55.2
94	60	108-60-26	68.3
102	65	108-65-26	89.1
130	70	108-70-26	91.1

6. RESULT ANALYSIS

From the results, the following observations are made:

- A small number of nodes in the hidden layer (eg. 26) lower the accuracy.
- A large number of neurons in the hidden layer help in increasing the accuracy; however there is probably some upper limit to this which is dependent on the data being used. Additionally, high neuron counts in the hidden layers increase training time significantly.
- As number of hidden layer increases the accuracy increases initially and then saturates at certain rate probably due to the data used in training.
- Mostly Accuracy is increased by increasing the number of cycles.
- Accuracy could also be increased by increasing the training set.

7. CONCLUSION

The back-propagation neural network discussed and implemented in this paper can also be used for almost any general image recognition applications such as face detection and fingerprint detection. The implementation of the fully connected back-propagation network gave reasonable results toward recognizing characters. The most notable is the fact that it cannot handle major variations in translation, rotation, or scale. While a few pre-processing steps can be implemented in order to account for these variances, as we did. In general they are difficult to solve completely.

REFERENCES

[1] S. Mori, C. Y. Suen and K. Kamamoto, "*Historical review of OCR research and development*", Proc. of IEEE, vol. 80, **(1992)** July, pp. 1029-1058.

[2] N. Arica and F. Yarman-Vural, "*An Overview of Character Recognition Focused on Off-line Handwriting*", *IEEE Transactions on Systems, Man, and Cybernetic*s, Part C: Applications and Reviews, vol. 31, no. 2, **(2001)**, pp. 216 - 233.

[3] A. Rajavelu, M. T. Musavi and M. V. Shirvaikar, "*ANeural Network Approach to Character Recognition*", *Neural Networks*, vol. 2, **(1989)**, pp. 387-393.

[4] 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, **(2000)**, pp. 63-84.

[5] U. Bhattacharya and B. B. Chaudhuri, "*Handwritten numeral databases of Indian scripts and multistage recognition of mixed numerals*", *IEEE Transaction on Pattern analysis and machine intelligence*, vol. 31, no. 3, **(2009)**, pp. 444-457.

[6] M. Hanmandlu, K. R. M. Mohan and H. Kumar, "*Neural-based Handwritten character recognition*", *in Proceedings of Fifth IEEE International Conference on Document Analysis* and *Recognition, ICDAR*'99, Bangalore, India, **(1999)**, pp. 241-244.

[7] T. V. Ashwin and P. S. Sastry, "*A font and size-independent OCR system for printed Kannada documents using support vector machines*", in Sadhana, vol. 27, Part 1, **(2002)** February, pp. 35–58.

[8] S. V. Rajashekararadhya and P. Vanajaranjan, "*Efficient zone based feature extraction algorithm for handwritten numeral recognition of four popular south-Indian scripts*", *Journal of Theoretical and Applied Information Technology*, *JATIT*, vol. 4, no. 12, **(2008)**, pp. 1171-1181.

[9] R. G. Casey and E. Lecolinet, "*A Survey of Methods and Strategies in Character Segmentation*", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 7, **(1996)** July, pp. 690-706.

[10] R. C. Gonzalez, R. E. woods and S. L. Eddins, "*Digital Image Processing using MATLAB*", *Pearson Education, Dorling Kindersley*, South Asia, **(2004)**.

[11] T. V. Ashwin and P. S. Sastry, "*A font and size-independent OCR system for printed Kannada documents using support vectormachines*", in Sadhana, vol. 27, Part 1, **(2002)** February, pp. 35–58.

[12] M. Blumenstein and B. Verma, "*Neural-based solutions for the segmentation and recognition of difficult handwritten words from a benchmark database*", *in Proceedings of the Fifth International Conference on Document Analysis and Recognition, ICDAR* '99, **(1999)** September, pp. 281 –284.

[13] Y. Tay, M. Khalid, R. Yusof and C. Viard-Gaudin, "*Offline cursive handwriting recognition system based on hybrid markov model and neural networks*", *in Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation 2003*, vol. 3, **(2003)** July, pp. 1190 – 1195.

[14] G. Kim, V. Govindaraju and S. Srihari, *"A segmentation and recognition strategy for handwritten phrases", in Proceedings of the 13th Interna-tional Conference on Pattern Recognition 1996*, vol. 4, **(1996)** August, pp. 510–514.

[15] Y. Y. Chung and M. T. Wong, *"Handwritten character recognition by fourier descriptors and neural network", in Proceedings of IEEE Region 10 Annual Conference on Speech and Image Technologies for Computing and Telecommunications, TENCON* '97, vol. 1, **(1997)** December, pp. 391 –394