ISSN 2229-5518

Handwritten Character Recognition using Neural Network

Chirag I Patel, Ripal Patel, Palak Patel

Abstract— Objective is this paper is recognize the characters in a given scanned documents and study the effects of changing the Models of ANN.Today Neural Networks are mostly used for Pattern Recognition task. The paper describes the behaviors of different Models of Neural Network used in OCR. OCR is widespread use of Neural Network. We have considered parameters like number of Hidden Layer, size of Hidden Layer and epochs. We have used Multilayer Feed Forward network with Back propagation. In Preprocessing we have applied some basic algorithms for segmentation of characters, normalizing of characters and De-skewing. We have used different Models of Neural Network and applied the test set on each to find the accuracy of the respective Neural Network.

Index Terms— Optical Character Recognition, Artificial Nueral Network, Backpropogation Network, Skew Detection.

1 INTRODUCTION

Such software's are useful when we want to convert our Hard copies into soft copies. Such software's reduces almost 80% of the conversion work while still some verification is always required.

Optical character recognition, usually abbreviated to OCR, involves computer software designed to translate images of typewritten text (usually captured by a scanner) into machine-editable text, or to translate pictures of characters into a standard encoding scheme representing them in (ASCII or Unicode). OCR began as a field of research in artificial intelligence and machine vision. Though academic research in the field continues, the focus on OCR has shifted to implementation of proven techniques [4].

2 ARTIFICIAL NUERAL 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.

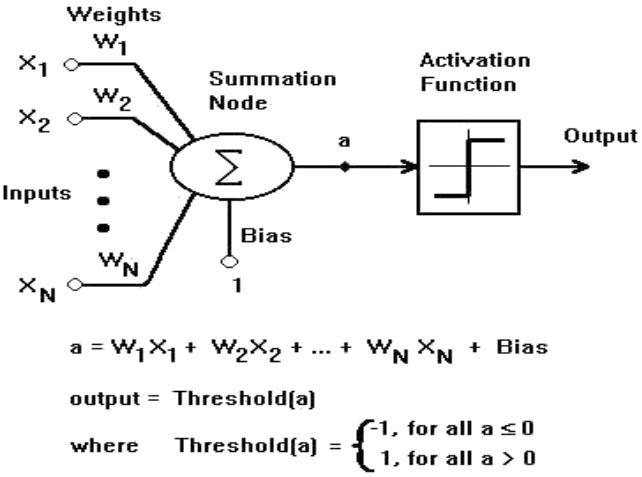


Fig. 1 A simple Neuron

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. 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 applica-

- *Chirag I Patel has been completed his M.Tech in Computer Science engineering in Nirma Institute of Technology, Ahmedabad, India, PH-91-9979541227. E-mail: chirag453@gmail.com*
- *Ripal Patel has been completed her M.E in Electronics & C ommunication engineering in Dharmsinh Desai Institute of Technology, Nadiad, India, PH-91-9998389428.E-mail: ripalpatel315@gmail.com*
- *Palak Patel is persuing her M.E in Electronics & C ommunication engineering in G.H.Patel College of Enginnering & Technology, Vallabh Vidyanagar, India, PH-91-9998389428.E-mail: ipalakec@yahoo.com*

tion 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 backpropagation neural network. It also encapsulates the Preprocessing that is required for effective.

2.1 Backpropogation

Backpropagation 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 ANALYSIS

By analyzing the OCR we have found some parameter which affects the accuracy of OCR system [1][5]. The parameters listed in these papers are skewing, slanting, thickening, cursive handwriting, joint characters. If all these parameters are taken care in the preprocessing phase then overall accuracy of the Neural Network would increase.

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 [2]. 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 Network, train it and note the output accuracy.
There are basically two main phases in our Paper:
Preprocessing 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 EdgeDetection).

5 PREPROCESSING

5.1 Character Extraction Algorithm

1. Create a TraverseList :- List of pixels which have been already traversed. This list is 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 Edgedetection Algorithm.
4. Add the List of Pixels returned by Edgedetection Algorithm to TraverseList.
5. Continue the steps 2 - 5 for all rows

5.2 Edge Detection Algorithm

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

```
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.

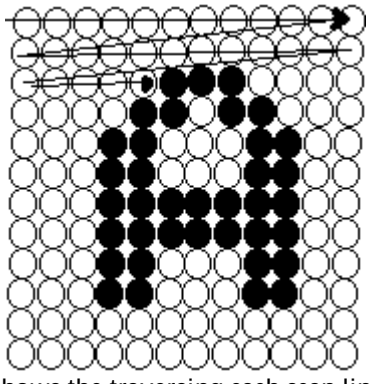
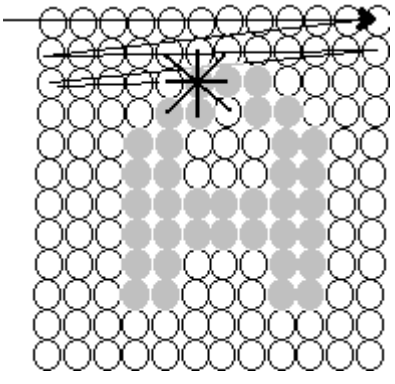


Fig 4(a) shows the traversing each scan lines.



4(b) shows the respective calls made to the all 8-neighbouring pixels.

The Edge detection is called when we hit a pixel (i.e. encounter a pixel with value 1). As per the algorithm the current position is entered in TraverseList and recursive calls are made to all 8 - neighboring pixels. Before the calls are made it is ensured that the corresponding neighboring pixels is having value 1 and is not already encountered before i.e. it should not be in the TraverseList.

5.3 Normalizing

Now as we have extracted the character we need to normalize the size of the characters. There are large variations in the sizes of each Character hence we need a method to normalize the size.

We have found a simple method to implement the normalizing. To understand this method considers an example that we have extracted a character of size 7 X 8. We want to convert it to size of 10 X 10. So we make a matrix of 70 X 80 by duplicating rows and columns. Now we divide this 70 X 80 into sub Matrix of 7 X 8. We extract each sub matrix and calculate the no. of ones in that sub matrix. If the no. of one's is greater than half the size of sub matrix we assign 1 to corresponding position in normalized matrix. Hence the output would be a 10 X 10 matrix.

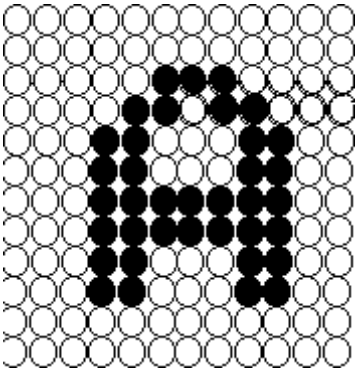


Fig 5(a) shows original representation of the character.

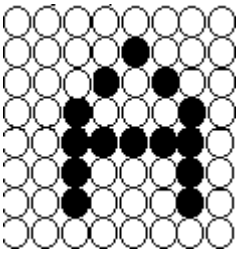


Fig 5(b) shows the Normalized Character representation after Normalizing.

The Fig 5(a) is shows a representation of character of 12 X 12 size. Using the above algorithm it is converted into a character of 8 X 8 as shown in the Fig 5(b).

5.4 Skew Detection

The Characters are often found to be skewed. This would impose problems on the efficient character recognition [3]. So to correct the effect of this skewedness we need counter rotate the image by an angle θ.

We use a very simple but effective technique for Skew Correction. We use "Line Fitting" i.e. Linear Regression to find the angle θ. Consider the Skewed character as a graph i.e. all the pixels that have value 1 are considered to be data points. Then we perform linear regression using the equation Y = M*X +C. Using the formulas for regres-

sion we calculate M= (nΣxiyi - ΣxiΣyi) / (nΣxi2-(Σxi)2). This angle is equivalent to the skewed angle so by rotating the image by opposite of this angle will remove the skew ness. This is a very crude way of removing skew ness there are other highly efficient ways of removing skew ness. But for Characters that have very low Skew angles this gets the thing done.

The Characters are often found to be skewed. This would impose problems on the efficient character recognition. So to correct the effect of this skewed ness we need counter rotate the image by an angle θ.

.We use a very simple but effective technique for Skew Correction. We use "Line Fitting" i.e. Linear Regression to find the angle θ. Consider the Skewed character as a graph i.e. all the pixels that have value 1 are considered to be data points. Then we perform linear regression using the equation Y = M*X +C. Using the formulas for regression we calculate M= (nΣxiyi - ΣxiΣyi) / (nΣxi2-(Σxi)2). This angle is equivalent to the skewed angle so by rotating the image by opposite of this angle will remove the skew ness. This is a very crude way of removing skew ness there are other highly efficient ways of removing skew ness. But for Characters that have very low Skew angles this gets the thing done.

Fig 6(a) Skewed Image

Fig 6(b) Corrected Image.

6 NUERAL NETWORK DESIGN

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

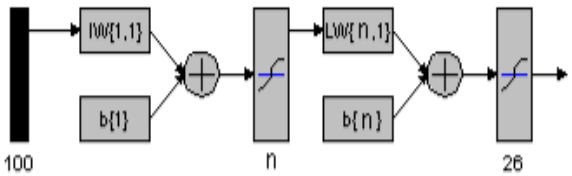


Fig 7. The general Model of ANN used.

Fig 8. Ttraining automated character extraction

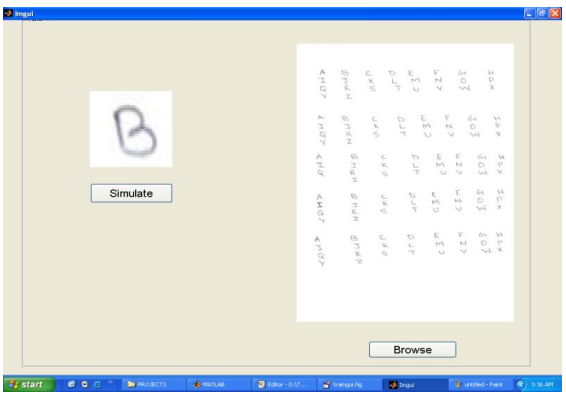


Fig 9 recognition

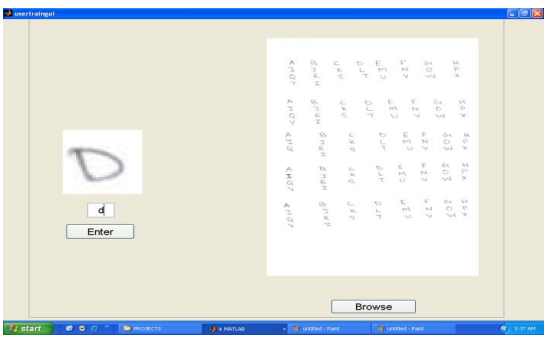


Fig 10 Training – User defined Character Extraction.

7 TEST AND RESULTS ANALYSIS

7.1 Test

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 100 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 1. Model 1

Epochs	Number of Hidden Layer	Configuration (No. of nodes in HL)	(%)
300	1	26	20
600	1	26	65
1000	1	26	82
300	1	52	25
600	1	52	69
1000	1	52	88
300	1	78	27
600	1	78	71
1000	1	78	91

Table 2. Model 2

Epochs	Number of Hidden Layer	Configuration (No. of nodes in HL)	(%)
300	2	26-52	23
600	2	26-52	67
1000	2	26-52	81
300	2	52-78	40
600	2	52-78	78
1000	2	52-78	96
300	2	26-78	27
600	2	26-78	77
1000	2	26-78	89

Table 3. Model 3

Epochs	Number of Hidden Layer	Configuration (No. of nodes in HL)	Accuracy (%)
300	3	26-52-26	31
600	3	26-52-26	65
1000	3	26-52-26	82
300	3	26-52-78	29
600	3	26-52-78	74
1000	3	26-52-78	92
300	3	78-26-78	27
600	3	78-26-78	71
1000	3	78-26-78	91

Table 4. Model 4

Epochs	Number of Hidden Layer	Configura (No. of nodes in HL)	Accuracy (%)
300	3	26-52-26	31
600	3	26-52-26	65
1000	3	26-52-26	82
300	3	26-52-78	29
600	3	26-52-78	74
1000	3	26-52-78	92
300	3	78-26-78	27
600	3	78-26-78	71
1000	3	78-26-78	91

Table 5. Model 5

Epochs	Number of Hidden Layer	Configura (No. of nodes in HL)	Accura (%)
300	4	26-52-78-104	35
600	4	26-52-78-104	79
1000	4	26-52-78-104	96
300	4	26-52-78-26	30
600	4	26-52-78-26	61
1000	4	26-52-78-26	86
300	4	78-26-52-104	43
600	4	78-26-52-104	82
1000	4	78-26-52-104	98
300	4	78-26-78-52	31
600	4	78-26-78-52	88
1000	4	78-26-78-52	94

Table 6. Model 6

Epochs	Number of Hidden Layer	Configura (No. of nodes in HL)	Accura (%)
300	4	26-52-78-104	35
600	4	26-52-78-104	79
1000	4	26-52-78-104	96
300	4	26-52-78-26	30
600	4	26-52-78-26	61
1000	4	26-52-78-26	86
300	4	78-26-52-104	43
600	4	78-26-52-104	82
1000	4	78-26-52-104	98
300	4	78-26-78-52	31
600	4	78-26-78-52	88
1000	4	78-26-78-52	94

We have used sigmoid transfer function in all the layers. We have used same dataset for training all the different Models while testing character set was changed.

7.2 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.

7.3 Additional Formatting and Style Resources

Additional information on formatting and style issues can be obtained in the IJSER Style Guide, which is posted online at: http://www.ijser.org/. Click on the appropriate topic under the Special Sections link.

8 Conclusion

The backpropagation 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 backpropagation 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. Basavaraj Patil, N. V. Subbareddy 'Neural network based system for script identification in Indian documents' in Sadhana Vol. 27, Part 1, February 2002, pp. 83–97.

[2] T. V. Ashwin, P. S. Sastry 'A font and size-independent OCR system for printed Kannada documents using support vector machines' in Sadhana Vol. 27, Part 1, February 2002, pp. 35–58.

[3] Kavallieratou, E.; Fakotakis, N.; Kokkinakis, G.,' New algorithms for skewing correction and slant removal on word-level [OCR]' in Proceedings of ICECS '99.

[4] Simmon Tanner, "Deciding whether Optical Character Recognition is Feasible".

[5] Matthew Ziegler, "Handwritten Numeral Recognition via Neural Networks with Novel Preprocessing Schemes".