Middle-East Journal of Scientific Research 20 (11): 1506-1510, 2014
ISSN 1990-9233
© IDOSI Publications, 2014
DOI: 10.5829/idosi.mejsr.2014.20.11.924

Neural Network Architecture for Recognition of Cursive Handwriting

[1]T. Saravanan, [2]G. Saritha and [3]V. Srinivasan

[1,3]Department of ETC, Bharath University, Chennai, India
[2]Research Scholar, Sathyabama University, Chennai, India

Abstract: Handwriting recognition has been a problem that computers are not efficient at. This is obviously due to the varying letter styles that exist. Today, efficient handwriting recognition is limited to ones, using hardware like light pens wherein the strokes are directly detected and the character is recognized. But to convert a handwritten document to digital text, one has to extract the characters and then recognize the extracted character. But the problem with this approach is that there are not many algorithms that could efficiently extract characters from a sentence. Therefore, character recognition using software is still not as efficient as it could be. In this paper, a design for software which could do this job of translating handwritten text to digital text, is proposed. Implementation details of this software is proposed and for this implementation, a new Neural Network Architecture, which is a modified version of the conventional Back Propagation Network is also proposed. Also, the details regarding the automatic preprocessing that are essential along with the shortcomings are also discussed in this paper.

Key words: Back Propagation Technique % Look Ahead Pointer % Hand Written Text

INTRODUCTION

Many character recognition systems appear to suffer from maladies in that they can perform one segment of the overall task well but are unable to fully duplicate the richness of the human's character recognition ality. Newer models exploit the same feedback and interaction between independent systems as is present within the visual cortex and provide the diversified processing power needed in order to function in a more robust manner. The multiple-layered system which makes up any robust handwriting recognizer has progressed greatly from the days when character recognition meant reading printed numerals of a fixed-size OCR-A font [1].

However, only recently have the successes within the field approached the level of a truly practical handwriting recognizer. Performance of single-algorithm system drops precipitously as the quality of input decreases. In such situations, a handwriting recognizer is not considered that reliable [2].

The Scenario: We should implement a completely computerized way of doing everything. Everything from rations, electricity, grocery, to taxes should be kept track of by computers. This would drastically reduce the speed of processing documents and manpower needed. This has to happen at some point of time, if not immediately, if India has to compete with the developed countries in any means. This is so because for over half a century, every detail has been recorded manually and if we are to switch to a digital means, all these need to be converted into the digital media too. This means more work. Manually feeding all these details into a digital media would take mammoth manpower and time which cannot be practically accomplished. So a solution for this problem would be to computerize this transformation too, by developing software that would recognize handwritten documents and transform it into digital files or databases [3].

Our Model: The basic design of our software includes a buffer which is of equal size as the input bitmap file. The whole of the file is first scanned into this buffer. All the processing involved in the recognition process is performed on the data present in this buffer. The innovation or the difference in the recognition process is the way these pixels are handled.

Corresponding Author: T. Saravanan, Department of ETC, Bharath University, Chennai, India.

In the already existing software, first a raster scan is performed on the bitmap and each character is recognized making them not suitable for recognizing running letters. To avoid this, characters are not segmented before recognizing them [4].

We accomplish the recognition process as and when the scanning is done. We use two simultaneous scans, a vertical and horizontal one; feed them to two different neural networks.

The key in this is that when a horizontal scan is performed, the pixels are fed into a neural network which is trained to identify the bases of various letters. Once the base of a letter is identified, the characters, in the order in which they appear in the sentence, are transferred to a second buffer where the sentences are placed. This is where the vertical scan comes into play [5].

The sentence buffer is scanned vertically and the pixels are fed into the second neural network which identifies the characters based on the pixel pattern. So when each of the pixels is read, it is fed into the neural network which recognizes it at real time. But the output of this is not determined until the character is decided for sure [6].

The Problem: The problem here is that during such a real time operation, letters like 'h' and 'b' could be recognized as 'l' before the complete character could be scanned. So there is a need for an extra constraint which would make sure that this kind of real time processing does not end up in wrong results. The constraint is brought in to the network such that the recognition of the character is based on the character recognized till the previous column of pixels and the current column. That is if a character (c1) has been recognized till the previous column of pixels and if the next few columns of pixels do not take it closer to any other character in terms of the hamming distance, then the corresponding character is recognized. If the next few columns do take it closer to some other character then, the column till which the character (c1) was recognized is kept track of and then if it starts to deviate from that character then it is still recognized as c1 and the next character starts from the column next to c1. On the contrary, if it takes the recognition extremely close to some other character (c2), then the character is recognized as c2 and not c1. For example, 'l' could be c1 and 'b' could be c2. In that case, c2 is recognized. If some other character like 't' were in the following pixels, it might take the letter closer to 'b' but starts to deviate there after then it is recognized as l and some other character. Thus the recognition could be performed at real time. Therefore the recognition could be extended to scripts written in cursive form [7].

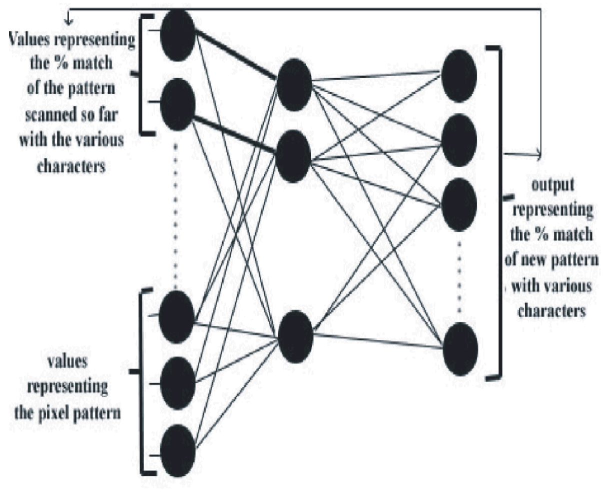
Implementation Using Back Propagation Networks: The idea that we proposed can be easily implemented using a conventional back propagation network. The network as usual can have one or two hidden layers. The number of output units is equal to the number of characters present in the language that is being recognized.

The number of units in the input layer is equal to the sum of the number of units in the output layer and the maximum height of the character. The input to the network consists of two different sets of data.

One set represents the pattern of the column of pixels that are currently being analyzed. The other set is basically the output that is being fed back into the input layer. The output of the network would represent the percentage match of the pattern so far scanned with the various characters. So when this information is fed back as a part of the input and the other part being the information on the pattern, then the output depends on both the previously recognized patterns and the current pattern [8].

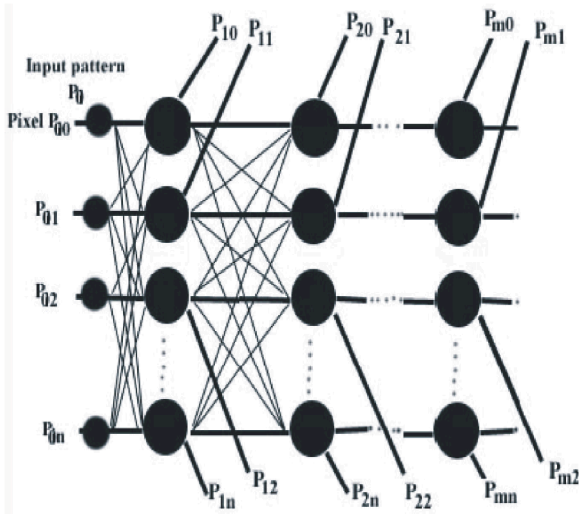
Based on this it is decided whether the pattern matches more with the characters or not. Thus when the percentage match exceeds a threshold, then the character is recognized.

But the problem with this implementation is that we would have to create a way to link the values representing the percentage match with the patterns and the characters that they stand for. Instead of this, we refined the neural network so as to give a new architecture that would perfectly suit the problem in hand.


Values representing the % match of the pattern scanned so far with the various characters
values representing the pixel pattern
output representing the % match of new pattern with various characters

The Custom-Made Neural Network Architecture: The proposed idea for recognizing the handwritten document demand a network where in the state transition during the analysis of each column of pixels could be represented. So we propose a new architecture of neural network that would suit this need. The network that we propose would accept inputs in each layer. The network would have more hidden layers than the maximum number of pixels forming the width of the characters [9].

The no. of hidden layers = maximum width of character + look ahead length.



But these hidden layers do not behave exactly as hidden layers in the fact that they take inputs too. The output of each layer would depend on the input it receives from the previous layer as well as the input representing the pixel pattern.

The Architecture of the Proposed Network Is Given below: Here P_{mn} is the array corresponding to the pixels of the sentence. m is the number of layers and n is the maximum height of the character.

The first layer of the network takes the input as the pattern of pixels in the first column. The output is fed into the second layer as usual. But the difference is that the pattern in the second column is fed as the input to the second column.

Thus all the layers except the first accept two inputs, one from the output of the previous layer and the pattern of the pixel from the corresponding pattern. Thus the output of any of these layers would converge to the value corresponding to a character if the pattern recognized so far corresponds to that character [10].

On that case, a marker is set to mark the column of pixels before which the character was recognized, then the look ahead is performed. In case, the look ahead yields in a better match, then the marker is updated and the process repeated. If the look ahead does not yield a better match, then the previously recognized character is taken into account and the scanning is done from where the marker points to. The training of this network could be done using delta rule much the same way as in the case of back propagation network, except for the fact that the input for each layer (pixel patterns) would be taken into account only during the feed forward.

The Training: The training of our neural network would be done in a different fashion as compared to the other already existing models. In our architecture of neural network, we are literally treating each layer as both input as well as output layer. This means that during the training we'd have to establish a mechanism to take care of the changing character length. We solve this technique too by using the approach described before.

The training used here is a supervised one. The training set includes the bitmap of the pattern that corresponds to a character and the value of the character that needs to be generated during the pattern recognition. The training itself is controlled by a program.

The program's modules split the input pattern of a character into columns of pixels and input each column to the corresponding layer, synchronizing it with the feed forward. It is the job of this trainer module to identify the height and width of the pixels. Based on this info the number of units and the layers in the network is chosen. During the training, the input pattern happens to be the pattern of the character that is split into columns of pixels and is fed into the corresponding layers so that the feed forward is synchronized with the input feed. The training is done using the delta rule as

$$NET_{Pj}^{h} = \sum_{i=1}^{N} W_{ji}^{h} \, x_{pi} + q_{j}^{h}$$

$$w(t+1)_i = w(t)_i + 2 \, {}_k x_{ki}$$

But the problem with our architecture is that when training for characters of smaller length like 'I', 'l' etc, only a part of the network is trained and the rest of the weights are left as such. Therefore the network would not perform as expected if trained randomly. Therefore this order is also determined by the training module. This effect could be explained using the following diagrams.

The above network shows the nodes involved in the recognition of character ‘w’. This shows that all the weights are adjusted during the training. But the diagram shown below is for the character ‘l’ wherein only the first layer is involved. So if we are to train the character ‘l’ after training the network for ‘w’ then some weights of the network is changed independent of the others which help in recognizing the characters.

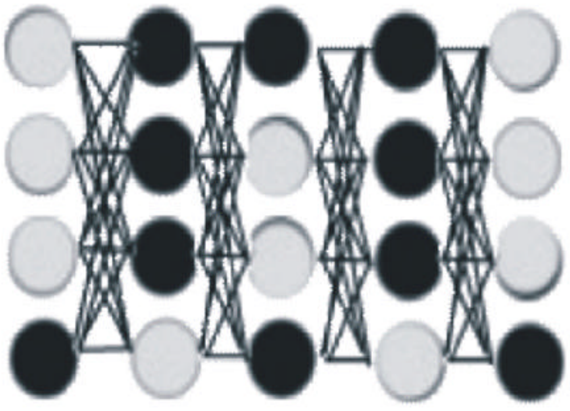
We overcome this problem simply by training the characters of shorter width first and the longest at the end, thereby eliminating the scenario where in there would be independent change. The other option that we had to overcome this was to maintain individual personality files for each character and running a sequential processing on these files during the addition of each column of pixels. But this method would be extremely time consuming.

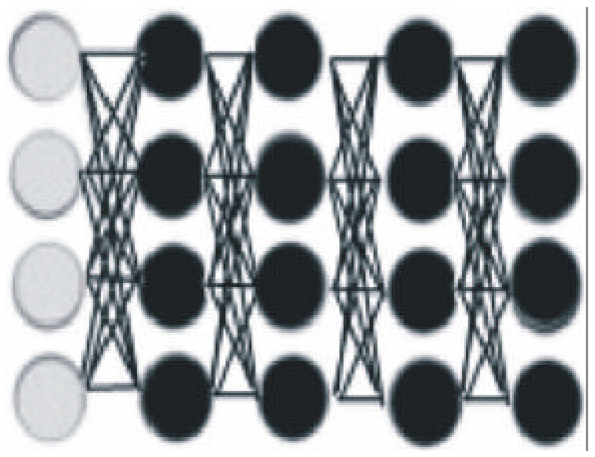
Recognition Algorithm:

```
C   repeat for j = 0,1,2,…, height of character
    Layer[0].output[j]=0;
C   layer_index = 1;
C   repeat for pix_index = 0,1,2,…, length of sentence
C   read pix_pattern_buffer [pix_index] [] into
    pix_pattern_vector []
C   Repeat for j = 0,1,2,…, height of character

    layer[layer_index].input[j] = func(layer[i-1].output[j],
pix_pattern_vector[j]

C   Propogate output to logical layer given by
    layer_index.
C   If check(layer_index) == TRUE
C   Look_ahead_ptr[char_match_arr_index] = pix_index
C   Char_match[char_match_arr_index] = character
    recognized. o Match_found = TRUE
```





```
C   Char_match_arr_index = char_match_arr_index + 1
C   repeat while ( LookAhead() == TRUE and
    look_ahead_cnt < 3) o
    Look_ahead_ptr[char_match_arr_index] = pix_index
C   Char_match[char_match_arr_index] = character
    recognized.
C   Char_match_arr_index = char_match_arr_index + 1 o
    best_match_index = findmax(charmatch)
C   string_concat(string_output,
    Char_match[best_match_index]
C   Pix_index = Look_ahead_ptr[best_match_index] o
    layer_index = 1;
C   else

C   Layer_index = layer_index + 1
C   Pix_index = pix_index + 1

9write string_output to output file;
```

The Look Ahead: As already discussed, when we go for a recognition based on the current and previous columns of pixels, chances are that letters like ‘b’, ’P’, ‘K’ etc would be recognized as ‘l’ and some other character. To avoid this we need to look ahead of the column of pixels till which a character is recognized. That is if the hamming distance of the pattern so far recognized is close enough to a character to be recognized and from the next pixel if it starts deviating, then we would keep track of the column of pixels (p1) where the distance was minimum. From there on we would look ahead if the closeness increases with some other character. If it does, then the second character is given as the output and the column mark p1’s value is changed to current column. If that is not the case, then the first character is given as the output and the next pattern is considered from p1.

The Preprocessing: The preprocessing required for the recognition is pretty simple, the whole of the bmp is resized by stretching the bmp as is necessary. This is done so that the characters to be recognized are in standard size. If only this is the case would the number of states per character would be the same as the ones given during training phase.

Advantages of Our Model:

- Recognition of running letters and cursive letters.
- Possible to identify all font styles
- Accurate recognition
- User specific training as in speech recognizers.
- Trade off speed-using converging to local minimum on the error surface.

CONCLUSION

Thus, it is inferred the challenges like failing to recognize a character or misrecognition can be avoided by splitting the bitmap file into characters only after they are recognized as predefined set of characters. The perfection of scanning increases by using the look ahead pointer since it scans the character till maximum matching percentage. Hence the efficiency is higher than in conventional models. Also the speed of scanning and recognition increases tremendously. Number of pixel columns that needs to be looked ahead before the network could deliver the output can be predefined the algorithm. Another factor of concern with this algorithm, when the proposed neural network is used, is the time constraint. But with faster processors coming up, this should not be much of a concern, at least not the major concern. Provided these challenges are taken care of, we believe that our algorithm could prove to be a highly effective means of recognizing characters with a high degree of precision.

REFERENCE

1. Zhang, J. and T. Tan, 2002. Brief review of invariant texture analysis methods, Pattern Recognition, 35(3): 735-747.
2. Campbell, R.J. and P.J. Flynn, 2001. A Survey of free-form object representation techniques Computer Vision and Image Understanding, 81(2).
3. Suri, J.S., J.S. Kecheng Liu, S. Singh and S.N. Laxminarayanan, 2002. Shape recovery Algorithms using level sets in 2-D/3-D medical imagery IEEE Transactions on Information Technology in Biomedicine.
4. Gavrila, D.M., 1999. The visual analysis of human movement: a survey, Computer Vision and Image Understanding, 73(1).
5. Vijayaragavan, S.P., B. Karthik, T.V.U. Kiran Kumar and M. Sundar Raj, 2013. Analysis of Chaotic DC-DC Converter Using Wavelet Transform, Middle-East Journal of Scientific Research, ISSN:1990-9233, 16(12): 1813-1819.
6. Vijayaragavan, S.P., B. Karthik, T.V.U. Kiran Kumar and M. Sundar Raj, 2013. Robotic Surveillance For Patient Care In Hospitals, Middle-East Journal of Scientific Research, ISSN:1990-9233, 6(12): 1820-1824.
7. Vijayaragavan, S.P., T.V.U. Kiran Kumar and M. Sundar Raj, 2013. Study of effect of MAI and its reduction in an OCDMA system, Middle-East Journal of Scientific Research, ISSN:1990-9233, 16(12): 1807-1812.
8. Gopalakrishnan, K., M. Prem Jeya Kumar, J. Sundeep Aanand and R. Udayakumar, 2013. Thermal Properties of Doped Azopolyester and its Application, Indian Journal of Science and Technology, ISSN: 0974-6846, 6(6): 4722-4725.
9. Gopalkrishnan, K., J. Sundeep Aanad and R. Udayakumar, 2013. Structural Properties Doped azopolyester and its characteristics, Middle-East Journal of Scientific Research, ISSN:1990-9233, 15(12): 1773-1778.
10. Gopalkrishnan, K., J. Sundeep Aanad and R. Udayakumar, 2013. Synthesis of Doped azopolyester using solution casting technique, Middle-East Journal of Scientific Research, ISSN:1990-9233, 15(12): 1813-1816.