Character Recognition Using Neural Networks and Convert into Text File

Anjuben Lunagariya
M.Tech. Student, Dept. Of Computer Science Engineering,
Rai University, Ahmedabad, Gujarat – 382260
(lunagaria.anju92@gmail.com)

Abstract-- Now a days there is a lot of development in smart devices which are combination of human intelligence and machines. Character recognition is one of the example of smart device and Mathematical expression recognition is belongs to such device which is developed to recognize printed mathematical symbols. This system conventional programming methods of mapping symbol images into matrices, analyzing pixel and/or vector data and trying to decide which symbol corresponds to which character would yield little or no realistic results. Clearly the needed methodology will be one that can detect 'proximity' of graphic representations to known symbols and make decisions based on this proximity.

In this paper discuses about then usefulness of neural networks, more specifically the motivations behind the development of neural networks, the outline network architectures and learning processes. We conclude with Mathematical Symbol recognition, a successful layered neural network application and try to convent into the text file. Here all character, number and mathematical formula are recognition and try to convent character into text file.

Keywords— *Optical Character Recognition; neural network; Mathematical symbol; proximity;*

I. INTRODUCTION

Recognizing Character in image file and convent into text file is a new and important field in document analysis. It is quite different from extracting mathematical expressions in image-based documents. In this paper, we propose a neural network method to detect both isolated and embedded mathematical expressions in image documents. Moreover, various features of formulas, including geometric layout, character and context content, are used to adapt to a wide range of formula types. Experimental results show satisfactory performance of the proposed method.

One of the most classical applications of the Artificial Neural Network is the Character Recognition System. This system is the base for many different types of applications in various fields, many of which we use in our daily lives. Cost effective and less time consuming, businesses, post offices, banks, security systems, and even the field of robotics employ this system as the base of their Operations. Handwritten character recognition is a difficult problem due to the great variations of writing styles, different size (length and height) and orientation angle of the characters. Handwritten Character recognition is an area of pattern recognition that has become the subject of research during the last some decades. Neural network is playing an important role in handwritten character recognition. Many reports of character recognition in English have been published but still high recognition accuracy and minimum training time of handwritten English characters using neural network is an open problem. Therefore, it is a great important to develop an automatic character recognition system for a mathematical symbol.

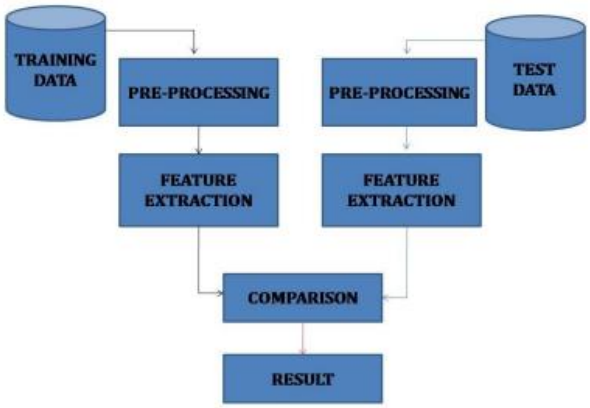

Fig 1. OCR Process

In this paper, efforts have been made to develop automatic mathematical symbol recognition system for mathematics with high recognition accuracy and minimum training and classification time. Hence the conventional programming methods of mapping symbol images into matrices, analyzing pixel and/or vector data and trying to decide which symbol corresponds to which character would yield little or no realistic results. Clearly the needed methodology will be one that can detect 'proximity' of graphic representations to known symbols and make decisions based on this proximity. To implement such proximity algorithms in the conventional programming one needs to write endless code, one for each type of possible irregularity or deviation from the assumed output either in terms of pixel or vector parameters, clearly not a realistic fare.

In this paper, one effective optical character recognition from text image using texture and topological features is proposed. For better performance, the texture and topological features of all characters of text image like corner points, features of different regions, and ratio of character area and convex area are calculated. Based on the texture and topological information, character verification is done using feature matching between the extracted character and the template of all character serves as a measure of similarity between the two. Also discuss about then usefulness of neural networks, more specifically the motivations behind the

development of neural networks, the outline network architectures and learning processes. We conclude with Mathematical Symbol recognition, a successful layered neural network application and try to convent into the text file. Here all character, number and mathematical formula are recognition.

II. Related work

The existing OCR systems show high accuracy in interpreting text portions but failed to properly process other components like graphics, half-tones, mathematical formulas and equations. Segmenting documents to text, graphics, half-tones, tables etc. have been reported in the literature by many researchers. However, segmenting math-zone is still a challenging problem. It has been observed from the existing literatures that most of the works are directed toward math-symbol or equation recognition assuming that the math-zones are already marked. Though, symbol recognition is a part of OCR activity but when it is applied to the non-segmented mixed material (text with math-zone and others) computation will be expensive and success far from satisfactory. We on the other hand contend that a better approach is to segment the math-zone from the mixed material thereby helping the future OCR activity to focus its processing only on math-symbols and equations. In this paper we propose fully automated segmentation technique extracting math-zone exploiting spatial distribution of black pixels on white background. Unlike many reported works we did not use any type of symbol recognition techniques for mathzone segmentation.

III. Problem Statement

Existing Character, mathematical symbol and mathematical formula are recognition but here to try convent into a text file to easy use and helpful. A mathematical symbol recognition identify two significant areas of weakness, that of correctly segmenting text and math lines, and precisely identifying the locations of mathematical formulae. For each of these issues we have proposed and implemented more advanced techniques, then rerun previously reported experiments and for each case reported significant improvements. Here all type of recognition but try only character to convent into the text file.

IV. IMPLEMENTATION ENVIRONMENT

1. Network Formation

The MLP Network implemented for the purpose of this project is composed of 3 layers, one input, one hidden and one output.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 hidden layer constitutes of 250 neurons 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.

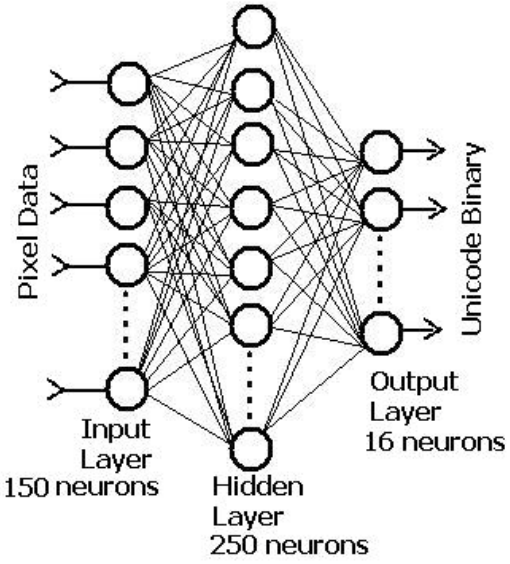

Fig 2. The Project MLP Network

2. Symbol image detection

The process of image analysis to detect character symbols by examining pixels is the core part of input set preparation in both the training and testing phase. Symbolic extents are recognized out of an input image file based on the color value of individual pixels, which for the limits of this project is assumed to be either black RGB(255,0,0,0) or white RGB(255,255,255,255). The input images are assumed to be in bitmap form of any resolution which can be mapped to an internal bitmap object in the Microsoft Visual Studio environment. The procedure also assumes the input image is composed of only characters and any other type of bounding object like a boarder line is not taken into consideration.

The procedure for analyzing images to detect characters is listed in the following algorithms:

i. Determining character lines

Enumeration of character lines in a character image (‘page’) is essential in delimiting the bounds within which the detection can proceed. Thus detecting the next character in an image does not necessarily involve scanning the whole image all over again.

Algorithm:

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.

ii. Detecting Individual symbols

Detection of individual symbols involves scanning character lines for orthogonally separable images composed of black pixels.

Algorithm:

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

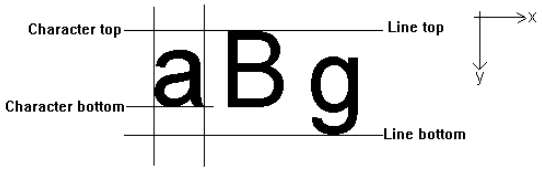

Fig 3. Line and Character boundary detection

From the procedure followed and the above figure it is obvious that the detected character bound might not be the actual bound for the character in question. This is an issue that arises with the height and bottom alignment irregularity that exists with printed alphabetic symbols. Thus a line top does not necessarily mean top of all characters and a line bottom might not mean bottom of all characters as well.

Hence a confirmation of top and bottom for the character is needed.An optional confirmation algorithm implemented in the project is:

1. start at the top of the current line and left of the character
2. scan up to the right of the character
   a. if a black pixels is detected register y as the confirmed top
   b. if not continue to the next pixel
   c. if no black pixels are found increment y and reset x to scan the next horizontal line

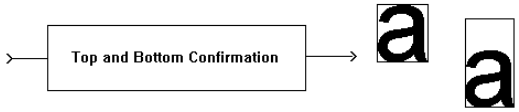

Fig 4. Confirmation of Character boundaries

3. 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. 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. The algorithm is listed below:

Algorithm:

a. For the width (initially 20 elements wide)
   1. Map the first (0,y) and last (width,y) pixel components directly to the first (0,y) and last (20,y) elements of the matrix
   2. Map the middle pixel component (width/2,y) to the 10th matrix element
   3. subdivide further divisions and map accordingly to the matrix
b. For the height (initially 30 elements high)
   1. Map the first x,(0) and last (x,height) pixel components directly to the first (x,0) and last (x,30) elements of the matrix
   2. Map the middle pixel component (x,height/2) to the 15th matrix element
   3. subdivide further divisions and map accordingly to the matrix
c. Further reduce the matrix to 10x15 by sampling by a factor of 2 on both the width and the height

.

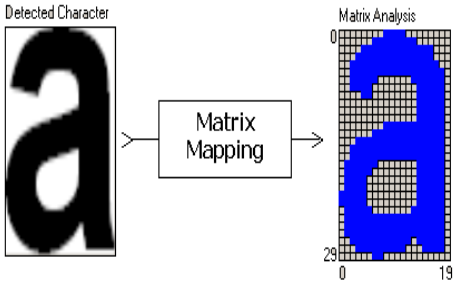

Fig. 5 Mapping symbol images onto a binary matrix

In order to be able to feed the matrix data to the network (which is of a single dimension) the matrix must first be linearized to a single dimension. This is accomplished with a simple routine with the following algorithm:

1. start with the first matrix element (0,0)
2. increment x keeping y constant up to the matrix width
   a. map each element to an element of a linear array (increment array index)
   b. if matrix width is reached reset x, increment y
3. repeat up to the matrix height (x,y)=(width, height)

Hence the linear array is our input vector for the MLP Network. In a training phase all such symbols from the trainer set image file are mapped into their own linear array and as a whole constitute an input space.

V. PROPOSED METHOD

Here I try to Using above algorithm and create a system recognize a image and convent into a text which helpful to using other data without any problem.

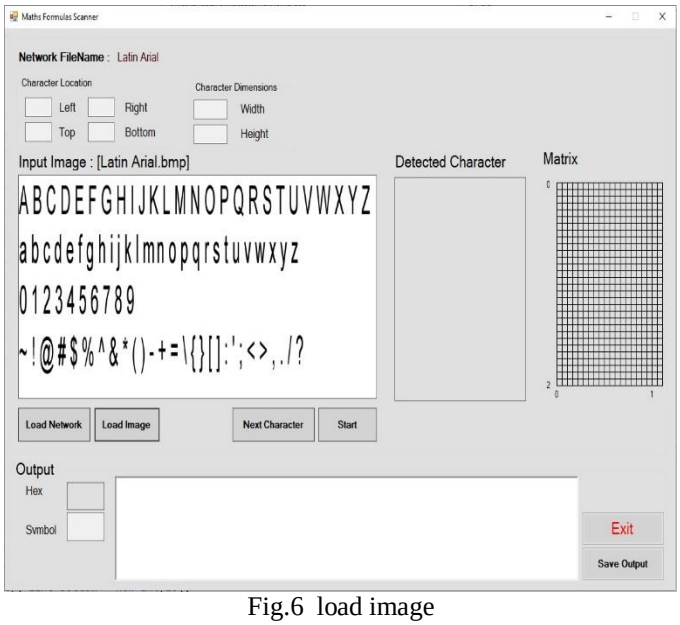

Fig.6 load image

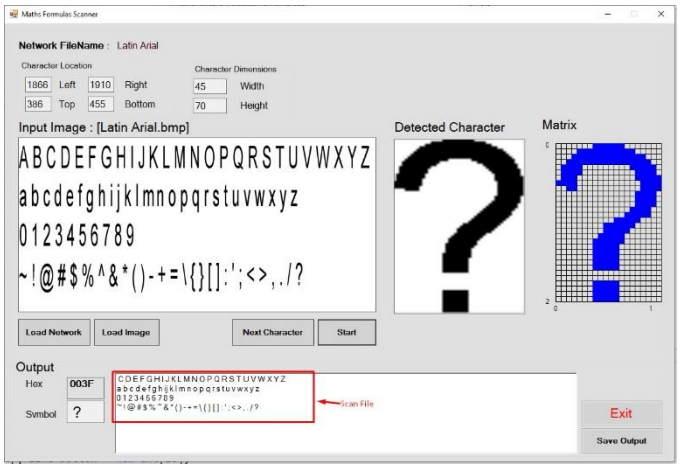

Fig.7 Scan the character and convent into a text file

VI. RESULT AND DISCUSSION

Although the results listed in the subsequent tables are from a training/testing process of symbol images created with a 72pt. font size the use of any other size is also straight forward by preparing the input/desired output set as explained. The application can be operated with symbol images as small as 20pt font size.

a. Results for variation in number of Epochs

Number of characters=90, Learning rate=150, Sigmoid slope=0.014

Font Type	300		600		800	
	No of wrong characters	% Error	No of wrong characters	% Error	No of wrong characters	% Error
Latin Arial	4	4.44	3	3.33	1	1.11
Latin Tahoma	1	1.11	0	0	0	0
Latin Times Roman	0	0	0	0	1	1.11

B. **Results for variation in number of Input characters**

Number of Epochs=100, Learning rate=150, Sigmoid slope=0.014

Font Type	20		50		90	
	No of wrong characters	% Error	No of wrong characters	% Error	No of wrong characters	% Error
Latin Arial	0	0	6	12	11	12.22
Latin Tahoma	0	0	3	6	8	8.89
Latin Times Roman	0	0	2	4	9	10

C. Results for variation in Learning rate parameter
Number of characters=90, Number of Epochs=600, moid slope=0.014

Font Type	50		100		120	
	No of wrong characters	% Error	No of wrong characters	% Error	No of wrong characters	% Error
Latin Arial	82	91.11	18	20	3	3.33
Latin Tahoma	56	62.22	11	12.22	1	1.11
Latin Times Roman	77	85.56	15	16.67	0	0

V. CONCLUSION

In conclusion, the combination of math retrieval and math recognition technologies provides rich possibilities for math-aware computer interfaces, and for intelligent search and retrieval tools for math in documents. The detection/segmentation technique utilized in this work can increase OCR accuracy in document images by allowing for a higher degree of document understanding prior to recognition. In order for mathematical regions to be properly recognized during OCR and not mangled with normal language text it is important that mathematical expression regions are detected and then properly segmented from their surroundings. The evaluation technique utilized in this work counts the true positive, false positive, true negative, and false negative pixels after detection and segmentation is carried out in order to get a highly accurate and objective understanding of performance. In this paper we have only convent text. In future utilized neural network and try to convent mathematical symbols and mathematical formula. Also try to other font style convent into a text.

REFERENCES

1. Aleksander, Igor, and Morton, Helen (1990), An Introduction to Neural Computing, Chapman and Hall, London.
2. Anzai, Yuichiro (1992), Pattern Recognition and Machine Learning, Academic Press, Englewood Cliffs, NJ.
3. Freeman, James A., and Skapura, David M. (1991), Neural Networks Algorithms, Applications, and Programming Techniques, Addison-Wesley, Reading, MA.
4. Hertz, John, Krogh, Anders, and Palmer, Richard (1991) , Introduction to the Theory of Neural Computation, Addison-Wesley, Reading, MA.
5. Rzempoluck, E. J. (1998), Neural Network Data Analysis Using Simulne, Springer, New-York.
6. Wasserman, Philip D. (1989), Neural Computing, Van Nostrand Reinhold, New York.
7. Gader, Paul, et al. (1992), "Fuzzy and Crisp Handwritten Character Recognition Using Neural Networks," Conference Proceedings of the 1992 Artificial Neural Networks in Engineering Conference, V.3, pp. 421–424.
8. Eugen Ganea, Marius Brezovan, Simona Ganea "CHARACTER RECOGNITION USING NEURAL NETWORKS" International Journal of Computer Applications Technology and Research.
9. Wei, WuFeng Li, Jun Kong, Lichang Hou, Bingdui Zhu ": A Bottom-Up OCR System for Mathematical Formulas Recognition" International Conference on Intelligent Computing.
10. Qi Xiangwei, Pan Weimin, Yusup, and Wang Yang. The study of structure analysis strategy in handwritten recognition of general mathematical expression. In IFITA '09. International Forum on Information Technology and Applications, 2009., volume 2, pages 101 –107, may 2009.
11. Richard Zanibbi, Dorothea Blostein, and James R. Cordy. Recognizing mathematical expressions using tree transformation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 24:1455–1467, 2002.
12. Erik G. Miller and Paul A. Viola. Ambiguity and constraint in mathematical expression recognition. In AAAI '98/IAAI '98: Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence, pages 784–791, Menlo Park, CA, USA, 1998. American Association for Artificial Intelligence.
13. Colby McKibbin ": Optical Character Recognition Using Artificial Neural Networks" Colorado State University-Pueblo Honors Thesis Spring 2015.
14. Christian Bartz, Haojin Yang, Christoph Meinel "STN-OCR: A single Neural Network for Text Detection and Text Recognition" Colorado State University-Pueblo Honors Thesis Spring 2015.
15. M. Adeel, H.S. Cheung, and H.S. Khiyal. Math go! Prototype of a content based mathematical formula search engine. J. Theoretical and Applied Information Technology, 4(10):1002–1012, 2008.
16. A.V. Aho, B.W. Kernighan, and P.J.Weinberger. The AWK Programming Language. Addison-Wesley, New York, 1988.
17. M. Altamimi and A.S. Youssef. An extensive math query language. In ISCA Int'l Conf. Software Engineering and Data Engineering, pages 57–63, Las Vegas, USA, 2007.
18. W. Aly, S. Uchida, and M. Suzuki. Identifying subscripts and superscripts in mathematical documents. Mathematics in Computer Science, 2(2):195–209, 2008.
19. R.H. Anderson. Syntax-Directed Recognition of HandPrinted Two-Dimensional Equations. PhD thesis, Harvard University, Cambridge, MA, 1968.
20. J. Gllavata, R. Ewerth , B. Freisleben, "A Robust Algorithm for Text Detection in Images", 3rd International Symposium on Image and Signal Processing and Analysis, 2003. ISPA 2003.
21. R.H. Anderson. Two-dimensional mathematical notation. In K.S. Fu, editor, Syntactic Pattern Recognition, Applications, pages 174–177. Springer, New York, 1977.

Anju Lunagariya is s PG Scholar at Rai University, Ahmedabad. She is especially interested in Artificial intelligence and Optical Character Recognition.