Unicode Mizo Character Recognition System using Multilayer Neural Network Model

J. Hussain, Lalthlamuana

Abstract - The current investigation presents an algorithm and software to detect and recognize pre-printed mizo character symbol images. Four types of mizo fonts were under investigation namely – Arial, Tohoma, Cambria, and New Times Roman. The approach involves scanning the document, preprocessing, segmentation, feature extraction, classification & recognition and post processing. The multilayer perceptron neural network is used for classification and recognition algorithm which is simple and easy to implement for better results. In this work, Unicode encoding technique is applied for recognition of mizo characters as the ASCII code cannot represent all the mizo characters especially the characters with circumflex and dot at the bottom. The experimental results are quite satisfactory for implementation of mizo character recognition system.

Keyword: Character Recognition, Neural Network, Multi-Layer Perceptron, and Unicode.

I. INTRODUCTION

Character Recognition is the electronic translation of images of handwritten, typewritten or printed text (usually captured by a scanner) into machine editable text. All OCR systems include an optical scanner for reading text, and sophisticated software for analyzing images. In OCR processing, the scanned image or bitmap is analyzed for light and dark areas in order to identify each alphabetic letter or numerical digit. When a character is recognized, it is generally converted into ASCII code for english language and Unicode for other languages [8]. In this study, Unicode encoding technique is applied for recognition of mizo characters as the ASCII code cannot represent all the mizo characters. Unicode is an expedition of Unicode Consortium to encode every possible languages but ASCII only used for frequent American English encoding. ASCII only supports 128 characters while Unicode supports much more than 109000 characters.

With recent advancements in neural networks, many recognition tasks have shown good capabilities in performing character recognition. There are many algorithms based on ANN to achieve OCR. In this paper, it has been achieved to recognize mizo letters using Multi-Layer Perceptron (MLP) neural network model.

II. PROPERTIES OF MIZO SCRIPT

Basic mizo character set comprises of 25 alphabets, 6 vowels, and 10 numerical. The mizo script are mostly derives from latin script and hence similar in nature except special characters incorporated in mizo script such as Â, â, Ê, ê, Î, î, Ô, ô, Û, û, Ṭ, and ṭ.

Manuscript received on May 2014.

Associate Prof. Jamal Hussain, Department of Mathematics & Computer Science, Mizoram University, Aizawl - 796001, Mizoram, India.

Mr. Lalthlamuana, Department of Mathematics & Computer Science, Mizoram University, Aizawl - 796001, Mizoram, India.

These special characters with their circumflex and dot at the bottom are not available in english script. Therefore, mizo fonts have been developed using unicode standard to enable to generate all the mizo characters. In mizo script, there are compound characters such as “AW”, “CH”, and “NG”. These compound characters are treated as a single character in mizo script. But for the purpose of recognition, the character ‘C’ and ‘H’ are treated as separate character. In this work, the mizo characters are divided into four classes such as:

Capital letter: A AW B CH D E F G NG H I J K L M N O P R S T Ṭ U V Z

Small letter: a aw b ch d e f g ng h i j k l m n o p r s t ṭ u v z

Numerals: 0 1 2 3 4 5 6 7 8 9

Vowels: Â â ÂW âw Ê ê Î î Ô ô Û û

The special characters with circumflex and dot at the bottom have separate meanings and different pronunciation than the characters without circumflex and dot at the bottom. Hence, the special characters are very important characters in mizo script while writing, reading and speaking.

III. METHODOLOGY

In this work, the entire process can be broken down into optical scanning the document, preprocessing, segmentation, feature extraction, and passing into MLP neural network for training and simulation. These steps are visualized in fig 1.

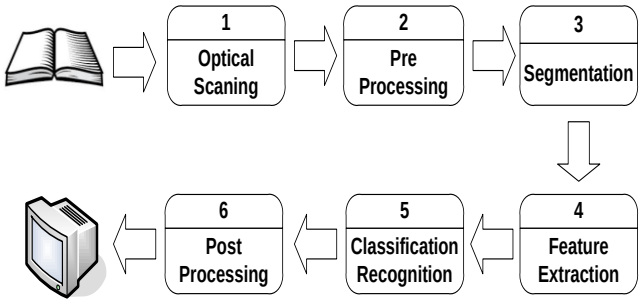

Fig 1: Implementation methodology of Mizo character recognition system

A. Optical Scanning

The proposed recognition system acquires a scanned image as an input image. The image should have specific format such as jpg, bmp, etc. This image is acquired through a scanner, digital camera or any other suitable digital input device [5]. The size of the input image is as specified by the user and can be of any length but is inherently restricted by the scope of the vision and by the scanner software length.

B. Preprocessing

Preprocessing consists of number preliminary processing steps to make the raw data usable for segmentation. The scanned image is first converted into grayscale image. The gray scale image is again converted into binary image known as binarization. Binarization separates the foreground (text) and background information [1]. The most common method for binarization is to select a proper threshold for the intensity of the image and then convert all the intensity values above the threshold to one intensity value ("white"), and all intensity values below the threshold to the other chosen intensity ("black").

After determining the threshold value, each pixel in the image is compared with the threshold value. If the value of the pixel is less than the threshold, reset the pixel to one. Otherwise, reset the pixel to zero as in Equation 1:

$$P(x,y) = \begin{cases} 1 : P(x,y) < \text{threshold value} \\ 0 : P(x,y) > \text{threshold value} \end{cases} \quad (1)$$

Where, P(x, y) is the pixel of the image and the threshold value 255 is the value between the dominant and the maximum value. After applying the binarization algorithm on the digital image, we obtain a binary image consisting of two values 1 as black and 0 as white.

C. Segmentation

After pre-processing, the noise free image is passes to the segmentation phase, where the image will be decomposed into individual character. Segmentation is an integral part of any text based recognition system. It assures efficiency of classification and recognition. Accuracy of character recognition heavily depends upon segmentation phase. Incorrect segmentation leads to incorrect recognition. Segmentation phase include segmentation of character lines and segmentation of individual character. It is important to obtain complete segmented character without any noise to ensure quality feature extraction [9].

(1) Segmentation of character lines:

The character line in a character image 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.

(2) Segmentation of Individual Character:

This involves scanning character lines for orthogonally separable images and divided into characters and saved in an array. Again the main assumption is that no merge between characters and no break points in the single character [2].

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

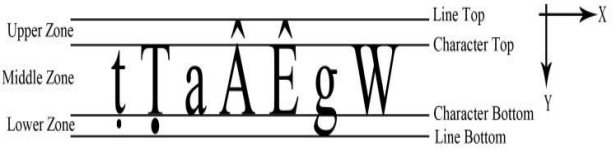

Fig. 2: Line and Character boundary detection

D. Feature Extraction

In feature extraction, the character images are represented by a set of numerical features. These features will be used by the classifier to classify the data. The numerical features of the images could be height of the character, width of character, and pixels in the various regions [7]. In this study, the individual character image is represented by two dimensional binary matrixes. All the pixels of the character are mapped into the matrix to acquire all the distinguishing pixel features of the character and minimize overlap with other characters. However this strategy would imply maintaining and processing a very large matrix (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 character 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.

Algorithm:

1. Width of Character
a. Map the first (0,y) and last (width,y) pixel component of the matrix
b. Map the middle pixel component (width/2,y) of the matrix
c. subdivide further divisions and map accordingly to the matrix
2. Height of Character
a. Map the first x,(0) and last (x,height) pixel components of the matrix
b. Map the middle pixel component (x,height/2) of the matrix
c. subdivide further divisions and map accordingly to the matrix
3. Further reduce the matrix to 10x15 by sampling both the width and the height

In order to be able to feed the matrix data to the neural network, the matrix must first be linearized to a single dimension. This 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)

These linear arrays are used as an input vector for the Multi-Layer Perceptron (MLP) neural network for classification.

E. Classification and Recognition

Classification is done using the features extracted in the previous step, which corresponds to each character glyph. These features can be analyzed using the set of rules and labeled as belonging to different classes. There are three types of classifiers such as (i) A typical rule based classifier; (ii) Neural Network based classifier; (iii) Support vector machine based classifier.

In this work, MLP network is chosen for classification because of its simplicity and ease of implementation. The Multi-Layer Perceptron (MLP) neural Network is perhaps the most popular network architecture in use today [11]. 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 [5]. 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.

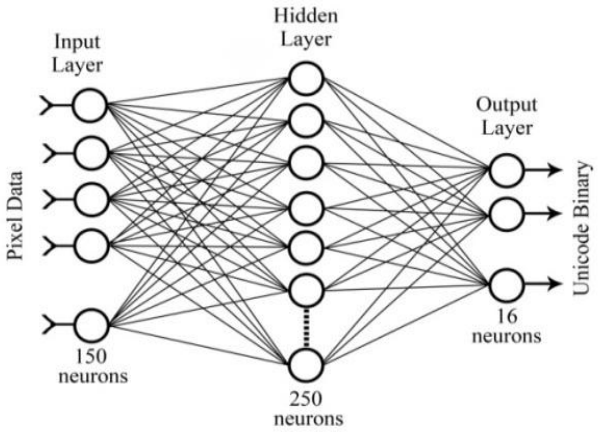

Fig. 3: Multi-Layer Perceptron (MLP) neural network

(1) Training:

Once the network has been initialized and the training input space prepared the network is ready to be trained [11]. Some issues that need to be addressed upon training the network are (a) a chaotic input varies randomly and in extreme range without any predictable flow among its members, (b) Complexity of the patterns which are usually characterized by feature overlap and high data size, (c) the number of iterations (epochs) are needed to train the network for a given number of input sets, (d) error threshold value must be used to compare against in order to prematurely stop iterations if the need arises. All these issues can be addressed by setting appropriate values for Learning rate, sigmoid slope, number of epoch and weight bias.

Algorithm:

1. Form network according to the specified topology parameters
2. Initialize weights with random values within the specified ±weight_bias value
3. load trainer set files (both input image and desired output text)
4. analyze input image and map all detected symbols into linear arrays
5. read desired output text from file and convert each character to a binary Unicode value to store separately
6. for each character :
a. calculate the output of the feed forward network
b. compare with the desired output corresponding to the symbol and compute error
c. back propagate error across each link to adjust the weights
7. move to the next character and repeat step 6 until all characters are visited
8. compute the average error of all characters
9. repeat steps 6 and 8 until the specified number of epochs
a. Is error threshold reached? If so abort iteration
b. If not continue iteration

(2) Testing:

The testing phase of the implementation is simple and straightforward.

Since the program is coded into modular parts the same routines that were used to load, analyze and compute network parameters of input vectors in the training phase can be reused in the testing phase as well.

Algorithm:

1. load image file
2. analyze image for character lines
3. for each character line detect consecutive character symbols

a. analyze and process symbol image to map into an input vector

b. feed input vector to network and compute output

c. convert the Unicode binary output to the corresponding character and render to a text box

F. Post-processing

Post-processing stage is the final stage of the proposed recognition system [3]. It prints the corresponding recognized characters in the structured text form by calculating equivalent Unicode value using recognition index of the test samples.

IV. EXPERIMENT & RESULTS

The OCR is implemented in Microsoft .NET using visual C#. The neural network has been trained and tested for a number of mizo fonts such as Arial, Tohoma, Cambria, and New Times Roman. The necessary steps involved preparing the sequence of input character images in a single image file (*.bmp), typing the corresponding characters in a text file (*.cts) and saving the two in the same folder (both must have the same file name except for their extensions). The application will provide a file opener dialog for the user to locate the *.cts text file and will load the corresponding image file by itself [6]. A screen shot of the software is shown in Fig 4 below.

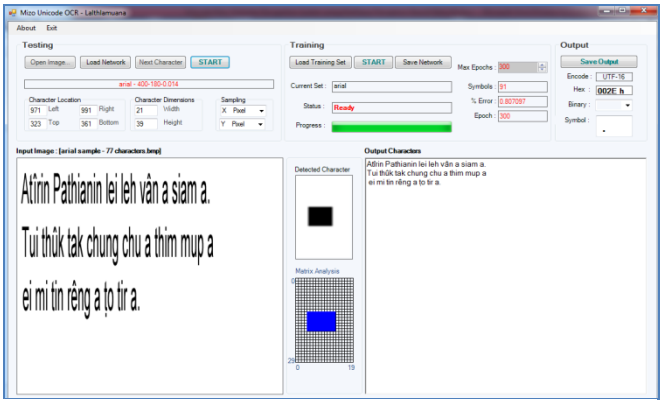

Fig. 4: A screen shot of Mizo OCR software

In this experiment, the MLP network is used with 3 layers having one input layer constitutes of 150 neurons which receive pixel binary data from a 10x15 symbol pixel matrix, one hidden layer constitutes of 250 neurons whose number is decided on the basis of optimal results on a trial & error basis and one output layer composed of 16 neurons corresponding to the 16-bits of Unicode encoding. The MLP network parameters are sets with Learning rate = 108-180, sigmoid slope = 0.014, weight bias = 30 (determining trial and error), no of epochs = 400-700 (depending on the complexity of the fonts types), mean error threshold value = 0.0007 (determined by trial and error).

The network has been trained and tested for four types of mizo fonts which are extensively used in Mizoram. It was observed the number of wrong characters and percentage error by variation of particular parameters keeping all other constant. The experimental results have been found as below:

A. Result for variation in number of epochs (iterations)

No of characters = 77, Learning rate = 180, Sigmoid slope = 0.014

Mizo Font Type	400		700	
	No of wrong characters	% Error	No of wrong characters	% Error
Arial	0	0	1	1.30
Tahoma	1	1.30	1	1.30
Cambria	1	1.30	1	1.30
Times New Roman	0	0	0	0

B. Result for variation in number of Input characters

No of epochs = 400, Learning rate = 180, Sigmoid slope = 0.014

Mizo Font Type	108		130	
	No of wrong characters	% Error	No of wrong characters	% Error
Arial	0	0	0	0
Tahoma	1	0.93	1	0.77
Cambria	1	0.93	1	0.77
Times New Roman	0	0	1	0.77

C. Result for variation in learning rate parameter

No of characters = 130, No of epochs = 500, Sigmoid slope = 0.014

Mizo Font Type	120		140	
	No of wrong characters	% Error	No of wrong characters	% Error
Arial	0	0	2	1.43
Tahoma	1	0.83	1	0.71
Cambria	0	0	0	0
Times New Roman	1	0.83	1	0.71

Effect of changing various parameters is studied and the influence of each variation is as given below:

(1) Increasing the number of iterations has generally a positive proportionality relation to the performance of the network. However in certain cases further increasing the number of iteration has an adverse effect of introducing more number of wrong recognitions. This phenomenon is known as overlearning.

(2) The size of the input character is also another factor influencing the performance. More number of input character set result the network susceptible for error. Usually the complex and large sized input sets require a large topology network with more number of iterations.

(3) Learning rate parameter variation affects the overall network performance for a given iterations. The less value of learning rate parameter results lower the value of network updates its weights which will impact more number of iterations required to reach its optimal state.

V. CONCLUSION

In the present work the ANN has been trained and tested for a number of widely used fonts type in the mizo alphabet. The Software has been developed in .NET framework incorporating Multilayer Perceptron neural network which is scalable to large scale production. The system developed gives about 98.7% accuracy which is quite satisfactory for the given four types of mizo fonts. In future, effort will be made to increase the effectiveness of results by adding more number of mizo fonts from typed language alphabet.

REFERENCES

1. Vivek Shrivastava and Navdeep Sharma, "Artificial Neural Network based Optical Character Recognition", Signal & Image Processing: An International Journal (SIPIJ), vol.3, No.5, October, 2012.
2. Mohanad Alata and Mohammad Al-Shabi, "Text Detection and Character Recognition using Fuzzy Image Processing", Journal of Electrical Engineering, vol.57, No.5, 2006, p258-267.
3. Pritpal Singh and Sumit Budhiraja, "Feature Extraction and Classification Techniques in OCR Systems for Handwritten Gurmukhi Script-A Survey", International Journal of Engineering Research and Applications (IJERA), vo.1, Issue 4, 2011, pp.1736-9622.
4. Seethalakshmi R, Sreeranjani T.R., and Balachandar T., "Optical Character Recognition for Printed Tamil Text using Unicode", Journal of Zhejiang University Science, 2005 6A(11):1297-1305, September, 2005.
5. Pramod J Simha and Suraj K V, "Unicode Optical Character Recognition and Translation Using Artificial Neural Network", International Conference on Software Technology and Computer Engineering (STACE-2012), 22nd July 2012, Vijayawada, Andhra Pradesh, India.
6. Kauleshwar Prasad, Devrat C. Nigam, Ashmika Lakhotiya, Dheeren Umre, "Character Recognition using Matlab's Neural Network Toolbox", International Journal of u- and e- Service, Science and Technology, Vol. 6, No. 1, February, 2013
7. Md. Mahbub Alam, Dr. M. Abul Kashem, "A complete Bangla OCR System for Printed Characters", International Journal of Computer and Information Technology, Vol. 01, Issue 01, July, 2010.
8. Madhup Shrivastava, Monika Sahu, and Dr. M.A. Rizvi, "Artificial Neural Network Based Character Recognition using Backpropagation" International Journal of Computers & Technology, vol. 3, No. 1, Aug, 2012
9. Om Prakash Sharma, M.K.Ghose, Krihna Bikram Shah and Benoy Kumar Thakur, "Recent Trends and Tools for Feature Extraction in OCR Technology", International Journal of Soft Computing and Engineering (IJSCE), volume-2, Issue-6, January, 2013.
10. Mark Hudson Beale, Martin T. Hagan, Howard B. Demuth, The Neural Network Toolbox™ 7 User's Guide. 3 Apple Hill Drive, Natick, MA: The Mathwork Inc., 2010
11. S.N. Sivanandam, S. Sumathi, S.N. Deepa, Introduction to Neural Networks using Matlab 6.0, Tata McGraw-Hill, 2006