Scientific Journal Impact Factor (SJIF) : 1.711
International Journal of Modern Trends in Engineering and Research
ISSN (ONLINE) : 2349 - 9745
ISSN (PRINT) : 2393 - 8161

Multi-font Optical Character Recognition System for Printed Telugu Text

D. Jayaram[1]
[1] Department of Computer Applications CBIT, Hyderabad

Abstract: The Telugu OCR systems available in the market currently recognize only the specific fonts of Telugu. This paper describes the development of a multi-font OCR system for printed Telugu characters using Artificial Neural Networks. In this system classification of the characters is carried out using multi layer neural network Architecture.

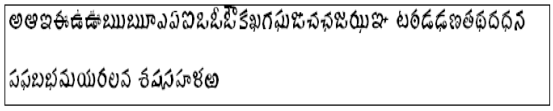
Keywords: OCR, MLP, Neural Networks, Segmentation, Feature extraction, Classification.

I. INTRODUCTION

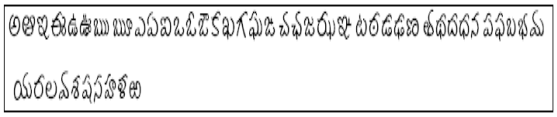
Optical character recognition (OCR) is software that translates handwritten or printed text into machine-editable form. Some of the Applications [1] of OCR are i) Preserving old/historical documents in electronic format. ii) Reading aid for the blind. iii) Automatic text entry into computer for desktop publication, library cataloging, ledgering, etc. iv) Data entry for business documents, e.g. check clearing. vi) Automatic number plate recognition etc..,

In recent years, the physical documents are transformed into electronic documents to facilitate easy communication and storage of the documents for many more years. Hence, there is a great demand for software, which automatically extracts and stores information from physical documents into electronic format for later retrieval. Some of the fonts [2] in Telugu are Pothana, Vemana2000, Krishna, Godavari, Raviprakash, Anuradha, Ponnala, Gautami etc.,

Raviprakash Font

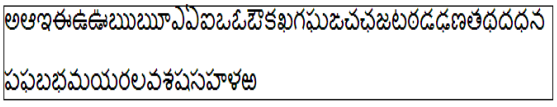


Vemana 2000 Font



Ponnala Font

Gautami Font



C.V Lakshmi et.,[3] al developed an OCR that recognizes basic characters of Harshapriya, Godavari, Hemalatha and Krishna with accuracy 92%. Analysis of their results shows that the recognition fails

only in case of symbols that are very similar to each other. Venkat et., al[4] developed an OCR that recognizes 15 font types with accuracy 96%. Analysis of their results shows that the recognition percentage decreases when no. of font types increases.

II. PHASES OF OCR SYSTEM

The Typical phases [5] of OCR system are:

- Pre-processing
- Segmentation
- Recognition/Classification

In the preprocessing [6] phase, the scanned input document is first converted into a gray scale image and then to a binary image, now the noise and skew is removed from the binary image. This phase consists of binarization, noise removing, thinning and skew detection and correction.

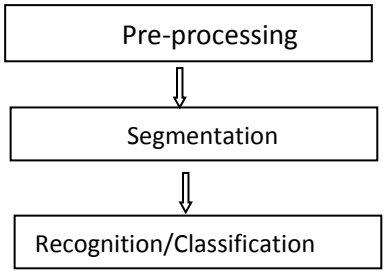


Fig. 1: Phases of typical OCR system

The segmentation [7] phase extracts lines, words and characters from the preprocessed image. The recognition phase consists of features extraction, features selection and classification. The segmented characters are given as the input to the extraction phase, then the extraction phase extract the features and then select the essential features from the selection phase. Finally classification phase recognize the characters based on the selected features.

III. FEATURE EXTRACTION

Feature extraction is a special form of dimensionality reduction. When the input data to an algorithm is too large to be processed and it is suspected to be notoriously redundant (much data, but not much information) then the input data will be transformed into a reduced representation set of features (also named features vector). Transforming the input data into the set of features is called feature extraction. If the features extracted are carefully chosen it is expected that the features set will extract the relevant information from the input data in order to perform the desired task using this reduced representation instead of the full size input.

The first step in the feature extraction is discourser. After this, the character image is resized to an appropriate resolution.

The feature extraction method to be used here is a pixel map where the symbol image is mapped [8] to 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 employs 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 is implemented.

Feature Extraction involves mapping the given character image into a 10x15 pixel matrix. It is implemented as follows.

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

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.
3) Map each element to an element of a linear array (increment array index).
4) If matrix width is reached reset x, increment y.
5) Repeat up to the matrix height (x, y) = (width, height).

Hence, the resultant linear array is our input vector for the MLP Network.

IV. CLASSIFICATION

Classification is carried out using Artificial Neural Networks. A Multi layer Neural Network Model is used for this purpose.

The operations on the network are classified into two phases as follows.

1) Training phase
2) Recognition phase

The training phase implements the following basic 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 Unicode value to store separately.
6) For each character
a. Calculate the output of the feed forward network [9].
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.

The recognition phase of the implementation is simple and straightforward. Since the program is coded into modular parts, the same routines that were used to segmentation, feature extraction and compute network parameters of input vectors in the training phase can be reused in the recognition phase as well. The basic steps can be summarized as follows.

1) Load image file.
2) Analyze image for character lines.
3) For each character line detect consecutive words.
4) For each word detect constituent character symbols.
a. Analyze and process symbol image to map into an input vector.
b. Feed input vector to network and compute output.
c. Render the Unicode binary output to a text box.

The MLP Network used for this one consists of 3 layers [10] that are input, hidden and output layers.
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 the character image that can be mapped without introducing any significant pixel noise.
The hidden layer constitutes of 150 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.

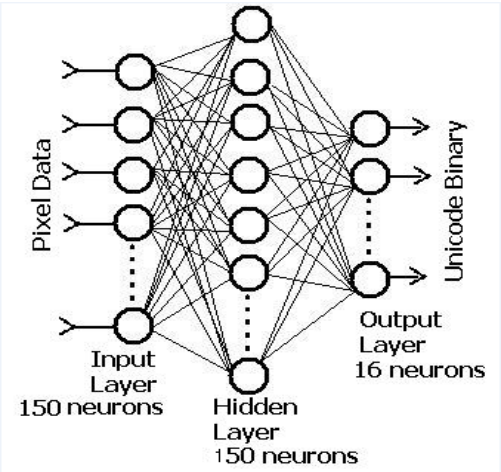


Fig 2: Network Formation

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. Here we have used the hyperbolic tangent function

is $$f(x) = \frac{2}{(1 + e^{-\lambda x})} - 1$$

and derivative is $f`(x) = f(x)(1 - f(x))$.
The parameters used are:

- Learning rate = 150
- Sigmoid Slope = 0.014
- Weight bias = 30 (determined by trial and error)
- Number of Epochs = 1150 (determined by trial and error)
- Mean error threshold value = 0.002 (determined by trial and error)
-

V . RESULTS AND DISCUSSION

The Software User Interface of the Multi-Font Optical Character Recognition System is shown in the figure 3. It is used for training the network as well as for recognizing images using the trained network.

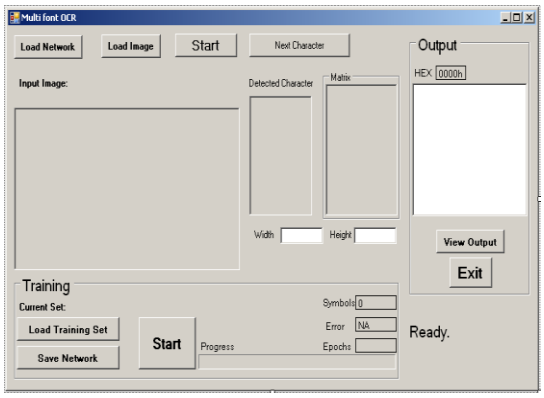
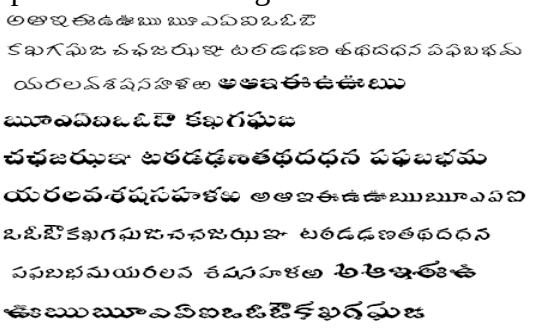


Fig 3: Software User Interface

The network has been trained and tested using widely used font types in Telugu. The following figure represents the training data set of the network.



The following figure represents the loading of the training data set.

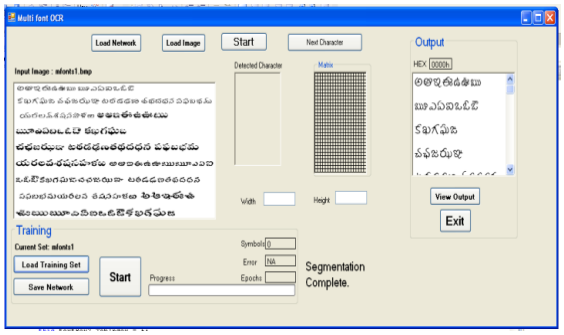


Fig 5: Snapshot of the Network Training in the Multi-Font OCR

The following figure represents the loading an image into the multi-font OCR system.

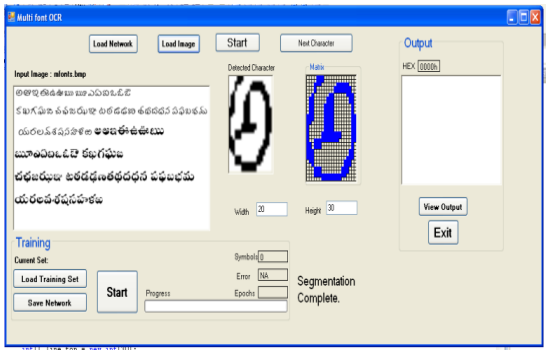


Fig 6: snapshot of the loading an input image into multi-font OCR.

The following figure represents the output of the multi-font OCR system.

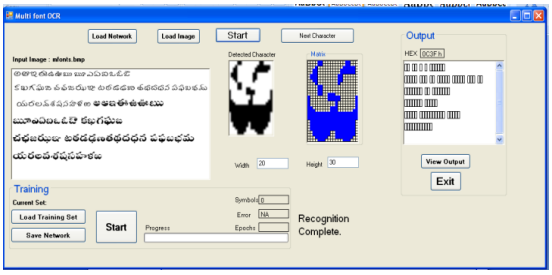


Fig 7: Snapshot of the recognition of multi-font OCR system.

The following table represents the accuracy of various fonts using multi-font OCR system.

Font type	No of characters in input	No of characters Recognized	% of accuracy
Ponnala	49	47	95.91
Vemana	49	46	93.87
Ravi prakash	49	47	95.91
Pothana	49	48	97.9

Table 1: Accuracy of various fonts

VI. CONCLUSION

This research is useful in solving classification problems. This system can recognize base characters of the text documents. This system can't recognize complex characters of Telugu.

REFERENCES

[1] U.Pal, B.B.Chaudhuri on "Indian Script Character Recognition: a survey", Pattern Recognition, vol 37, issue 9 pgs1887-1899.

[2] D.jayaram, CRK Reddy et.,al "An Overview of Optical Character Recognition Systems Research on Telugu Language" IJSAT volume2 issue 9 sep,2012 pgs 23-29.

[3] C.Vasantha Lakshmi, C.Patvardhan "A Multi-font OCR System for Printed Telugu Text" proceedings of the Language Engineering conference(LEC'02), Dec,2002 pgs 7-17.
[4] Venkatesh, jinesh, jawahar "On Multifont Character Classification in Telugu" proceedings of ICISIL, vol 139,March,2011, pgs 86-91.
[5] B.Anuradhaand, Arun Agarwal and C. Raghavendra Rao,"An Overview of OCR Research in Indian Scripts",IJCSES, Vol.2, No.2, 2008.
[6] B Anuradha, srinivas et., al "An Overview of OCR Research in Indian Scripts" IJCSES, vol 2 issue 2,Aprial 2008,pgs141-153.
[7] M.Swamy Das, Dr. C. R. K. Reddy, Dr. A. Govardhan, G. Saikrishna, "Segmentation of overlapping text lines, characters in printed telugu text document images", IJEST Vol. 2(11), 6606-6610, 2010.
[8] Rafael C.Gonzalez and Richard E.Woods, Digital Image Processing, Pearson Education,3rd edition, 2008.
[9] Andrew T. Wilson, *"Off-line Handwriting Recognition Using Artificial Neural Networks"*, University of Minnesota, Morris, 2000.
[10] Neural Networks a Comprehensive Foundations" by simon haykin.

