Handwritten Character Recognition

Abhinav Walde[1], Nilesh Thote[2], Shubham Thakare[3], Dr.D.R.Ingle[4]
Department Of Computer Engineering
[1,2,3] Bharati Vidyapeeth College of Engineering, Navi Mumbai
[4] Professor, Bharati Vidyapeeth College of Engineering, Navi Mumbai

Abstract- *Handwritten English text recognition is emerging areas of research in the field of optical character recognition. In this paper, using The K-nearest neighbor decision rule has often been used in these pattern recognition problems it is on based approach is used to recognize the text. The offline handwritten text is segmented into lines, lines into words and words into character for recognition. Shape features are extracted from the characters and fed into SVM classifier for recognition.*

I. INTRODUCTION

Humans recognize characters easily and they repeat the character recognition process thousands of times every day as they read papers or books. However, after many years of intensive investigation and research, the ultimate goal of developing an optical character recognition (OCR) system with the same reading capabilities as humans still remains unachieved. One of the main objectives of an OCR is to reach a 5 characters/second speed with a 99.9% recognition rate, with no errors. OCR is the process of converting an image representation of a document into an editable format. In the middle of the 1940s, the first character recognizers appeared and mainly focused on machine-printed text, and some of them dealt with handwritten text or symbols. In 1950s, commercial character recognizers were available for Latin languages. In 1980s, many structural and statistical methods were used in character recognition; some of those recognizers broke the character image into a set of lines and curves and basically focused on the shape recognition techniques without using any semantic information.

After 1990, complex character recognition algorithms were developed; many recognizers used sophisticated methodologies such as neural networks, hidden Markov models and natural language processing techniques. Many applications such as reading postal address off envelopes, reading customer filled forms, archiving and retrieving text and digitizing libraries benefit from OCR systems.

OCRs are divided into two major categories: typewritten and handwritten. Type written OCR systems recognize a document that has been previously typed and scanned prior to recognition progress. On the other hand, handwritten OCR systems recognize a text that has been written by a human. Comparing to handwritten OCR systems, Typewritten OCR systems are usually easier to design and the recognition rate achieved for typewritten recognition systems is more than the handwritten.

OCRs are further categorized to offline and online recognition systems. In offline OCR systems, the image of the typewritten or the handwritten text is acquired through scanning. The image then is read by the OCR system and is analyzed for recognition. In online OCR systems, input of the OCR system is an image of a handwritten text which is usually acquired using cell phone or a portable personal computer.

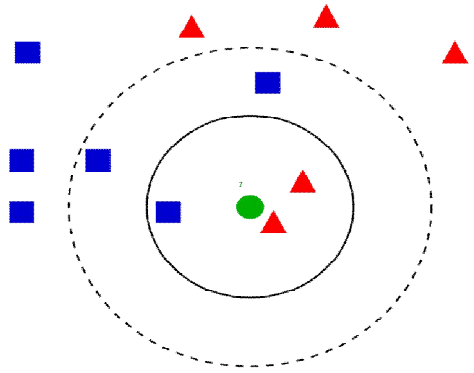
Aim

The aim of this project is to develop such a tool which takes an Image as input and extract characters (alphabets, digits, symbols) from it. The Image can be of handwritten document or Printed document. It can be used as a form of data entry from printed records.

1.2.ARCHITECTURE OF THE PROPOSED SYSTEM

The Architecture of optical characters recognition system on a grid infrastructure consists of three main components. They are:

- Scanner
- Ocr hardwired or software
- K-NN algorithm



1.2. Scope of the project

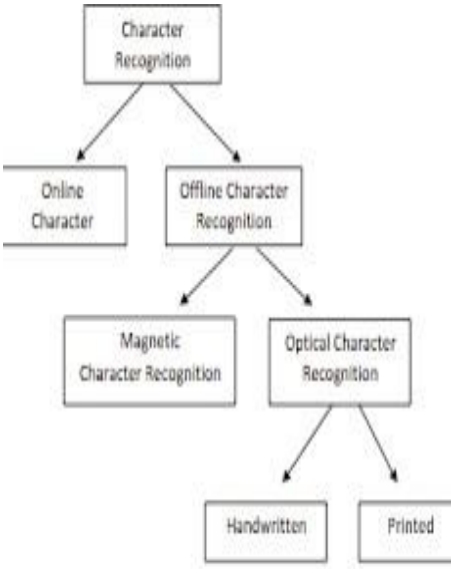
The scope of our product Optical Character Recognition on a grid infrastructure is to provide an efficient and enhanced software tool for the users to perform Document

Image Analysis, document processing by reading and recognizing the characters in research, academic, governmental and business organizations that are having large pool of documented, scanned images. Irrespective of the size of documents and the type of characters in documents, the product is recognizing them, searching them and processing them faster according to the needs of the environment.

1.3. TRAINING

For training purpose we had to know about are:

- Python
- Machine learning
- Data mining



III. NORMALIZATION

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.

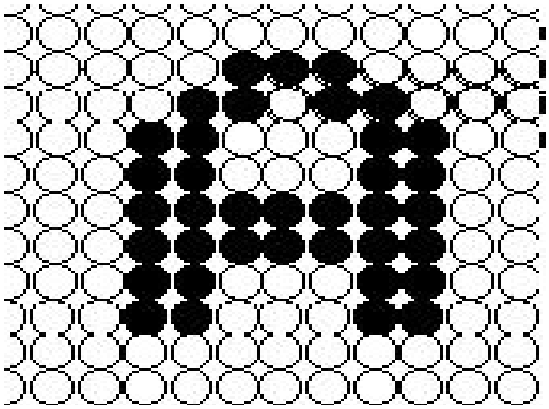


Fig 5(a) shows original representation of the character

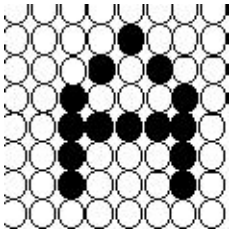
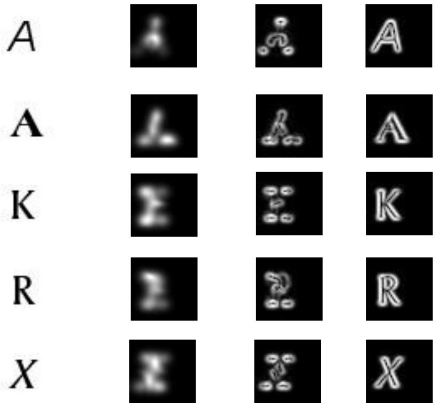


Fig 5(b) shows the Normalized Character representation after normalizing.

The Fig 5(a) is shows a representation of character of 12 X12 size. Using the above algorithm it is converted into character of 8 X 8 as shown in the Fig 5(b).

Skew Detection

The Characters are often found to be skewed. This would impose problems on the efficient character recognition. So to correct the effect of this skewdness we need counter rotate the image by an angle .We use a very simple but effective technique for Skew Correction. We use "Line Fitting" i.e. Linear Regression to find the angle. 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 regress



sion we calculate M= (nxiyi - xiyi) / (nxi2-(xi)2).This angle is equivalent to the skewed angle so by rotating the image by opposite of this angle will remove the skew news. This is a very crude way of removing sleekness there are other highly efficient ways of removing Skew ness . But for Characters that have very low Skew Angles this gets the thing done.

IV.SOFTWARE REQUIREMENTS SPECIFICATION

- System Operating: Windows-XP
- Programming Language : Core Java
- User Interface : Swings

V.HARDWARE REQUIREMENTS SPECIFICATION

Processor : Pentium IV processor or higher
RAM: Minimum of 512 MB RAM

VI. K-NEAREST NEIGHBORS

A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance Function. If K = 1, then the case is simply assigned to the Class of its nearest neighbor

6.1. Use of K-NN algorithm:

KNN can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry. To evaluate any technique we generally look at 3 important aspects:

Ease to interpret output
Calculation time
Predictive Power

Examples to place KNN in the scale:

	Logistic Regression	CART	Random Forest	KNN
1. Ease to interpret output	2	3	1	3
2. Calculation time	3	2	1	3
3. Predictive Power	2	2	3	2

KNN algorithm fairs across all parameters of considerations. It is commonly used for its easy of interpretation and low calculation time.

6.2. K-NN algorithm works:

6.2.1. Characters Extraction and Data Preparation
Singling out each character from an image using OpenCV's **find Contour** operation did not produce reliable results due to noise. For this specific problem, it was more robust to detect the "bounding box" around the character (image cropping) and then "single out" each digit out of the cropped image. The latter step is easy after finding the bounding box since each character will have a fixed coordinates relative to the upper-left corner of the cropped image.

6.2.2. Detecting the Bounding Box

Using third party tools to crop the boundaries of the images did not work well on all images. Instead, I created a simple method to deterministically crop the images and detect the bounding box with 100% accuracy.

The method starts by counting the blue pixels of a rectangle as shown in Figure. If the count of blue pixels exceeds an empirically set value, then the coordinates of the rectangle are the upper boundary of the digits and will be used to crop the image.

6.2.3 Character Extraction

Now that the bounding box is detected, it should be easy to single out each digit since each character will have pre-fixed coordinates relative to the top-left corner of the cropped image. It applied the above code on a set of images and manually sorted the images of each digit into separate folders labeled from 'A' to 'B' as shown below to create my training / testing dataset. Now that we created the training dataset and stored into **features** and **features label** arrays, we then divided our training sets into training and test sets using sklearn's function **train_test_split** and used the result to train a k-NN classifier and finally saved the model as illustrated in the code below.

6.2.4. Predicting

The process of predicting digits on new images follows the same steps of singling out the digits illustrated in the training steps above and then simply applying k-NN's *predict* function k-NN's *predict* function returns a single digit value between 'A' and 'Z' to denote the *prediction class* of the input image.KNN's *predict_proba* function returns the accuracy associated with each predicted class. Finally, predictions = list (map (lambda x: predict(x), hogs)) results in the following vector of tuples where each tuple represents the predicted class of each of the digits on the image with its associated prediction confidence. Any prediction that does not classify an input with 100% confidence will be presented to the user for manual correction as illustrated in the next section.

6.2.4. Presentation

The last step was to present the result of the Machine Learning model in an excel file as shown below. For digits that were not predicted with 100% accuracy, I embedded the image of the expected digit below the actual prediction. This minor presentation tweak decreased the user's time to fix the non accurate prediction by 80%. Furthermore, this activity is not daunting as it does not require significant mental effort. A user can scroll over the file in few minutes and visually matches the actual result to the expected result. Many of the predictions were actually false negative; hence the user did not have to make many corrections.

Fig 9 recognition

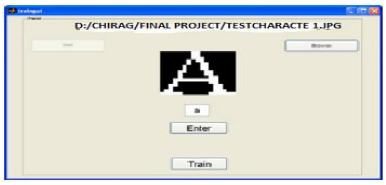


Fig 8.Ttraining automated character extraction

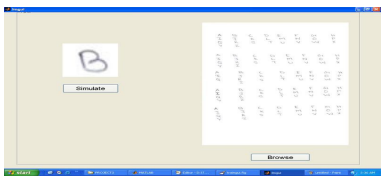


Fig 10 Training – User defined

VII. TESTS 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

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.

VIII. CONCLUSIONS

The back propagation 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 back propagation 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 lgorithms 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".
[6] Hamid, N. A., & Sjarif, N. N. A. (2017). Handwritten Recognition Using SVM, KNN and Neural Network. arXiv preprint arXiv:1702.00723.
[7] Patel, I., Jagtap, V., & Kale, O. (2014). A Survey on Feature Extraction Methods for Handwritten Digits Recognition. International Journal of Computer Applications, 107(12)