Character Recognition for Hindi in Devanagari Script

***Abstract*—Optical Character Recognition (OCR) is the electronic conversion of scanned images of hand written text into machine encoded text. Existing OCR engines are modelled through deep neural networks. In this project we try to explore a method which reduces the OCR task to a classification problem. It is assumed that the main image is broken into constituent characters after segmentation by plotting horizontal and vertical pixel densities which produces a number of isolated character images. These character images are processed and given to a classifier to perform the Character recognition. Various classification algorithms have been explored and compared, to design high performance character recognition software for Indian Language Hindi based on Devanagari script.**

Keywords—Devanagari, classification, ensemble.

I. INTRODUCTION

A.Motivation

OCR finds wide applications as a telecommunication aid for the deaf, postal address reading, direct processing of documents, foreign language recognition etc. This problem has been explored in depth for the Latin script. However, there are not many reliable OCR software available for the Indian language Hindi (Devanagari), the third most spoken language in the world[1]. [2] provides a good starting point for the problem and presents a good overview. The objective in this project is to design high performance character recognition model for Devanagari script that can help in exploring future applications such as navigation, for ex. traffic sign recognition in foreign lands etc.

B. Hindi Language Fundamentals

The Hindi Language consists of 12 vowels and 34 consonants. The presence of pre and post symbols added to demarcate between consonants and vowels introduces another level of complexity as compared to Latin script recognition. As a result, the complexity of deciphering letters out of text in Devanagari script increases dramatically because of presence of various derived letters from the basic vowels and consonants. In this project emphasis has been laid on recognizing the individual base consonants and vowels which can be later extended to recognize complex derived letters & words.

Fig. 1: Hindi Alphabet.

C. Devanagari Handwritten Character Dataset

Devanagari Handwritten Character Dataset is taken from Computer Vision Research Group [3]. It was created by collecting the variety of handwritten Devanagari characters from different individuals from diverse fields. Handwritten documents are than scanned and cropped manually for individual characters. Each character sample is 32x32 pixels and the actual character is centered within 28x28 pixels. Padding of 0 valued 2 pixels is done on all four side to make this increment in image size. The images were applied gray-scale conversion. After this the intensity of the images were inverted making the character white on the dark background. To make uniformity in the background for all the images, they suppressed the background to 0 value pixel. Each image is a gray-scale image having background value as 0. Devanagari Handwritten Character Dataset contains total of 92, 000 images with 72, 000 images in consonant dataset and 20, 000 images in numeral dataset. Handwritten Devanagari consonant character dataset statistics is shown in Table I and handwritten

Devanagari numeral character dataset statistics is shown in Table II.

TABLE I: Consonant Character Dataset

Class	क	ख	ग	घ	ङ	च	छ	ज	झ	ञ	ट	ठ
#	2,000	2,000	2000	2,000	2,000	2,000	2,000	2000	2,000	2,000	2,000	2,000
Class	ड	ढ	ण	त	थ	द	ध	न	प	फ	ब	भ
#	2,000	2,000	2000	2,000	2,000	2,000	2,000	2000	2,000	2,000	2,000	2,000
Class	म	य	र	ल	व	श	ष	स	ह	क्ष	त्र	ज्ञ
#	2,000	2,000	2000	2,000	2,000	2,000	2,000	2000	2,000	2,000	2,000	2,000
Total	**72,000**											

TABLE II: Numeral Dataset

Class	०	१	२	३	४	५	६	७	८	९
#	2,000	2,000	2000	2,000	2,000	2,000	2,000	2000	2,000	2,000
Total	**20,000**									

D. Proposed Methodology

The image containing text is broken into constituent characters after segmentation by plotting horizontal and vertical pixel densities which produces a number of isolated character images. These character images are processed and given to a classifier to perform the character recognition. [4] and [5] gives a good idea about the implementation of character segmentation.

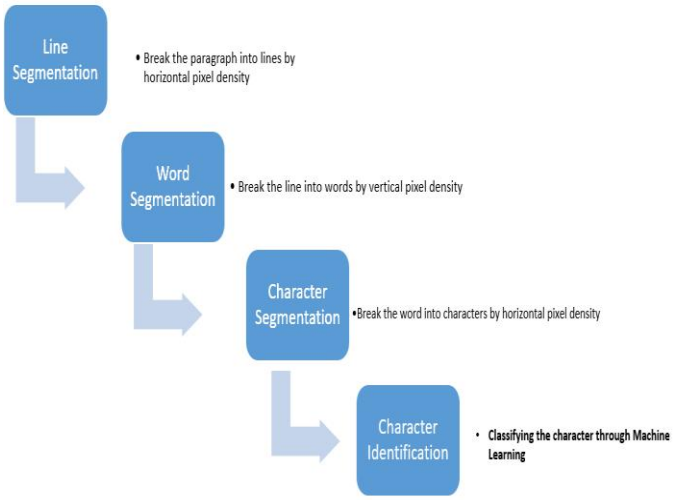

Fig. 4: Approach for proposed OCR.

E. Challenges in Devanagari Character Recognition

There are many pairs in Devanagari script, that has similar structure differentiating each with structure like dots, horizontal line etc. Some of the examples are illustrated in Fig. 2. The problem becomes more intense due to unconstrained cursive nature of writings of individuals. Two such examples are shown in Fig. 3.

छ	६	Difference being horizontal line at top
ड	ङ	Difference being presence of single dot on right side
द	ढ	Difference being presence of small circle and small down stroke line

Fig. 2: Structural formation of characters.

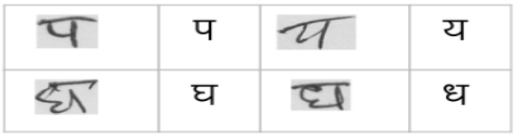

Fig. 3: Different characters written similarly.

II. Classification Methods

The task of classification is to assign an input pattern represented by feature vectors to one of many pre-specified classes. My main focus was on the following classifiers due to their unique characteristics.

A. Support Vector Machines (SVM)

SVM's (Support Vector Machines) are a useful technique for data classification. SVM is a supervised learning classifier. A classification task usually involves separating data into training and testing sets. Each instance in the training set contains one target value (class label) and several attributes (features). The goal of SVM is to produce a model which predicts the target value. Given a training set of attributes label pairs, (x_i,y_i), i=1…,l where $x_i \in R^n$ and $y \in \{-1,1\}$, the support vector machines require the solution of the following optimization problem given by (1) :

$$\min_{w,b,\xi} \quad \frac{1}{2}w^T w + C\sum_{i=1}^{l}\xi_i \qquad (1)$$

$$\text{subjected to} \quad y_i\,(w^T\phi(x_i) + b\,) \geq 1 - \xi_i\,, \qquad \xi_i \geq 0$$

B. k-Nearest Neighbour (kNN)

The nearest-neighbor classifier is one of the simplest of all classifiers for predicting the class of the test sample. Training phase simply store every training sample, with its label. To make a prediction for a test sample, its distance to every training sample is computed. Then, keep the k closest training samples, where k≥1 is a fixed integer. Then a label is searched that is most common among these samples. This label is the prediction for this test sample. This basic method is called the kNN algorithm. There are two major design choices to make: the value of k, and the distance function to use. We have

chosen k = 1, 3, 5 and 7 and for the minimum distance, the metric employed is the Euclidean distance given by

$$d(x,y) = \| x - y \| = \sqrt{(x-y)*(x-y)} = \left(\sum_{i=1}^{m} (x_i - y_i)^2 \right)^{\frac{1}{2}}$$

Where $x, y \in \mathbb{R}^m$

which evaluates the distance d(x, y) between test and training sample.

C. Random Forest

A random forest is a classifier consisting of a collection of tree-structured classifiers {h(x, θ_k), k = 1, ... } where { θ_k} are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input x. A summary of the random forest algorithm for classification is given below:

• Draw n_{tree} bootstrap samples from the original data.

• For each of the bootstrap samples, grow an unpruned classification tree, with the following modification: at each node, rather than choosing the best split among all predictors, randomly sample m_{try} of the predictors and choose the best split from among those variables. Bagging can be thought of as the special case of the random forest obtained when m_{try} = p, the number of predictors.

• Predict new data by aggregating the predictions of the n_{tree} trees, i.e., majority votes for classification, average for regression.

Random Forest generally is expected to be most stable given it's an ensemble of many decision trees and captures variance from a large number of variables.

III. Experimental Results

After generating the features, for choosing the classifier algorithm for OCR I experimented with many models.

Classifier	Score
SVM	0.421
kNN	0.704
Random Forest	0.543

Most of them performed poorly due to overfitting. When evaluating different settings ("hyper parameters") for classifiers, such as the C setting that must be manually set for an SVM, there is still a risk of overfitting on the test set because the parameters can be tweaked until the estimator performs optimally. This way, knowledge about the test set can "leak" into the model.

A solution to this problem is a procedure called cross-validation. The biggest concern with cross validation is to manage the tradeoff between minimize over-fit and minimize selection bias. The solution is to do a k-fold validation. K=7 was chosen and cross validation was performed on several classifiers. Below table gives the statistics

Trial	Bernoulli NB	Gaussian NB	Kneighbors Classifier	Nearest Centroid
1	0.49	0.43	0.72	0.52
2	0.47	0.38	0.70	0.49
3	0.48	0.41	0.71	0.52
4	0.46	0.39	0.72	0.50
5	0.48	0.42	0.70	0.49
6	0.51	0.42	0.70	0.53
7	0.51	0.42	0.71	0.54
Mean Accuracy	0.49	0.44	0.71	0.55

The highest mean accuracy of 0.71 is given by Knn classifier which is a marginal increase from previous result. To further improve the performance, I experimented with bagging algorithm. The Extra trees model is chosen for classification. It implements a meta estimator that fit a number of random trees on the training data and averages the predictive accuracy. The choice of number of trees is generally equal to sqrt(n) where n is the number of features.

I performed the analysis with number of trees ranging from 8 to 256. As number of trees increase the computation complexity increases.

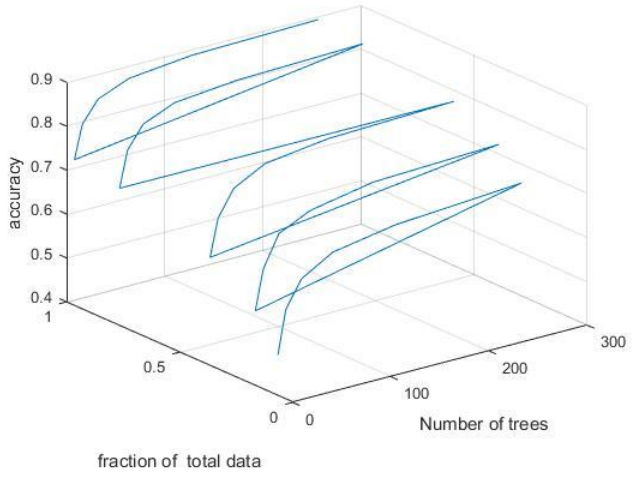

Fig. 4: 3D plot of Number of trees vs accuracy vs data size

From figure 4 it is evident that as the training data size becomes large better results are obtained. The best accuracy that is obtained is 89.36%. The accuracy almost converges at 90% with number of trees increased to 1024 although the computation time exponentially increases.

IV. Conclusions and Discussions

Existing character recognition modules utilize neural networks to achieve high accuracy. An attempt to perform character recognition through classification was made. After parameter tuning kNN classifier gave the best result of 71%. When bagging algorithm was implemented through Extra trees classifier a remarkable accuracy of 89.36% is achieved with little increase in computation cost.

It would be interesting to find how regression algorithms give results for multi class datasets. Dimensionality reduction through PCA and Adaboost can be experimented to find optimal model in terms of computation cost and accuracy.

Acknowledgment

I sincerely thank XXXXXXXXXXXXX for his continued efforts throughout the semester in making me understand theconcepts of machine learning and providing valuable inputsafter presentation that motivated me for further analysis.

References

[1] Wikipedia (https://en.wikipedia.org/wiki/Listoflanguagesbytotalnumberofspeakers)

[2] A. K. Pant, S. P. Panday, and S. R. Joshi, "Off-line nepali handwritten character recognition using multilayer perceptron and radial basis function neural networks," in Internet (AH-ICI), 2012 Third Asian Himalayas International Conference on. IEEE, 2012, pp. 1–5.

[3] Computer Vision Research Group (https://web.archive.org/web/20160105230017/http://cvresearchnepal.com/wordpress/dhcd/)

[4] Veena Bansal and R. M. Sinha, "A Complete OCR for Printed Hindi Text in Devanagari Script" 0-7 695-1 263- 1/010 2001 IEEE

[5] Brijmohan Singh et al., "Parallel Implementation of Devanagari Text Line and Word Segmentation Approach on GPU" in International Journal of Computer Applications (0975 – 8887) Volume 24– No.9, June 2011

[6] Pierre Geurts, Damien Ernst, Louis Wehenkel, "Machine Learning"

[7] E. Gose, R. Johnsonbaugh, and S. Jost. Pattern Recognition and Image Analysis. Prentice-Hall, 1996