Handwritten Devanagari Character Classification

Abhijeet Ghadge (G01274854)

December 12, 2021

Abstract

Researchers are currently interested in handwritten character recognition because of potential applications in supporting technology for blind and visually impaired users, human–robot interaction, automatic data input for commercial documents, and other areas. Handwritten Devanagari characters are difficult to recognize. Handwritten character recognition is one of the most challenging tasks due to individual differences in writing style. I have taken my Devanagari characters dataset from UCI Machine Learning repository. Optical character recognition is performed on this dataset using different classification models. The different classification models I have used are as follows K-Nearest Neighbors, Support Vector Machine, Random Forest, and Convolutional Neural Networks. Among the different experiments performed in this project, the accuracy of 98.06% was obtained using Convolutional Neural Networks.

1. Introduction

Devanagari is employed by more than 600 million people worldwide in nearly 120 languages. It is the most widely used script on the Indian subcontinent, and Hindi, the most widely spoken Indian language, is written in Devanagari. Devanagari is also used to write Nepali, Sanskrit, and Marathi. Furthermore, Hindi is India's national language and the world's third most popular language. The Devanagari script is made up of vowels and consonants, as well as modifier symbols. In today's computerized world, automated Devanagari character identification is an innovative, prominent, and demanding topic that has emerged via the use of artificial intelligence, pattern recognition, machine learning, and data mining principles. Non-Indic languages, such as English, Chinese, Japanese, Korean, German, and others, have more advanced optical character recognition (OCR) systems than the Indian scripts.

2. Problem Statement

Study and compare the recognition of handwritten Devanagari characters using classification algorithms such as K-Nearest Neighbors, Support Vector Machine, Random Forest, and Convolutional Neural Networks. This includes classification of –

- 36 consonants from ‘ka’ to ‘gya’
- 10 digits from ‘0’ to ‘9’

2.1. Notations

OCR: Optical Character Recognition
KNN: K- Nearest Neighbor
SVM: Support Vector Machine
CNN: Convolutional Neural Network
PCA: Principal Component Analysis
T-SNE: T-Distributed Stochastic Neighbor Embedding
UMAP: Uniform Manifold Approximation
PIV: Pixel Intensity Vectors

3. Literature Review

In [1] KNN is used to classify handwritten Devanagari vowels. Recursive subdivision technique is used to perform feature extortion which is nothing but the process of extracting feature points. In [2] Devanagari handwritten character recognition, thinning-based characteristics are employed. Three distinct types of feature points, namely cross, end, and branch points, are extracted initially from thinning images of handwritten Hindi characters. [3] provided an evaluation of feature extraction approaches for Devanagari characters and numerals, demonstrating that Euclidean Distance based K-Nearest Neighbor (ED-KNN) produced a higher recognition rate than SVM. In [4] authors implemented CNN of five different depths. The authors reported the accuracy of 96.09% with seven convolution layers and two fully connected layers. In [5], the authors have employed 6 different architectures of Deep CNN for feature extraction and character recognition. They have used layer wise training model which resulted in highest accuracy of 98%.

4. Methods and Techniques

4.1. Fetching data

- I downloaded the dataset from the UCI Machine Learning Repository. It consisted of 2 folders – 'Train' and 'Test' and each of these folders consisted of 46 (36 consonants and 10 digits) different folders each for a single Devanagari character.
- I iterated through each of these folders and used imread() method of OpenCV library in Python to read these images.

4.2. Exploratory Data Analysis

- The dataset I used is balanced as all the 46 classes have equal amount of data.
- The character images will be converted into pixel intensity vectors using asarray() function of numpy in Python and there will be no missing or Null values in these vectors.

4.3. Data Pre-processing

- The dataset of characters contained grayscale images in the .png format. Hence, to aid pattern recognition it was necessary to convert these images into some other format.
- These images will be first flattened into pixel intensity vectors ranging from values between 0 – 255.
- I performed scaling on the obtained pixel values.
- I also performed some image processing in terms of morphological operations.

4.4. Feature Engineering and Feature Selection

- I used Label Encoding on character names of the training and test data while using Convolutional Neural Networks as the classification algorithm.
- I tried out dimensionality reduction using Principal Component Analysis (PCA) and non-linear dimensionality reduction techniques like t – Distributed Stochastic Neighbor Embedding (t-SNE), and Uniform Manifold Approximation (UMAP).
- I also tried out dimensionality reduction using combination of techniques like PCA and t-SNE together and PCA and UMAP together.

4.5. Methods Used

I used the following classification algorithms to classify different Devanagari characters.

1) **K- Nearest Neighbors:**
   - I experimented with different values of K ranging from 3 to 145.
   - For KNN I got the best accuracy of 93.28% for k = 5 where the character images were morphologically dilated and then the obtained features were scaled.
   - For k greater than 5 the accuracy score dropped as the value of k increased.
   - Dimensionality reduction using PCA, t-SNE, UMAP or a combination of these techniques reduced the computation time but did not help in improving the accuracy.

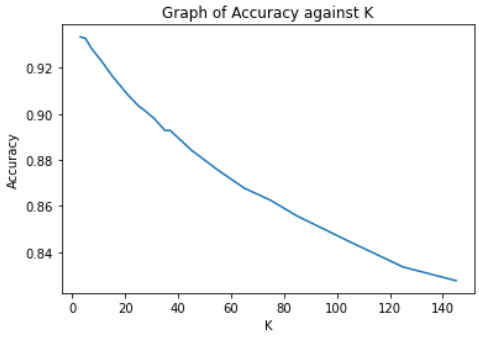

Fig 4.5.1 Accuracy score against different values of k

2) **Random Forest:**
   - For Random Forest, I tried out a range of estimator values from 50 to 1100, keeping the default values for all other parameters. Number of estimators when set to 1100 yielded the best accuracy.
   - I choose to stop at 1100 estimators as the model was already taking long time to fit and increasing the number of estimators would take even longer time to fit.

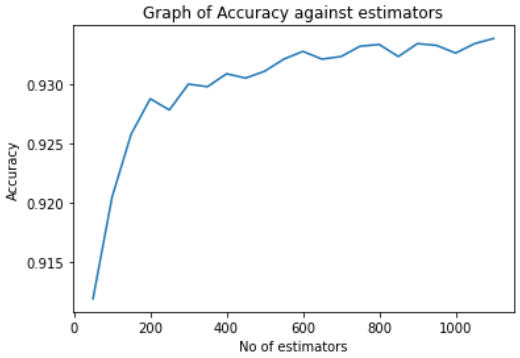

Fig 4.5.2 Accuracy score against number of estimators

3) **Support Vector Machine:**
   - For Support Vector Machine, I tried out different types of kernels and keeping the random state = 42.
   - All other parameters were set to their default values. Using the polynomial kernel gave the best accuracy score of 95.71%.

4) **Convolutional Neural Networks:**
   - In CNN, I label encoded the class labels and experimented with different number of layers to obtain the best accuracy among all the classification algorithms I tried out in this project.
   - The final CNN model included 2 convolutional + max-pooling layers with activation function set to ‘relu’.

- To avoid overfitting, I added a dropout layer after the above convolutional layers.
- Then the layers are flattened and passed on to 3 fully connected layers with ‘softmax’ activation function. Below are the graphs of accuracy and loss for the above-described model.

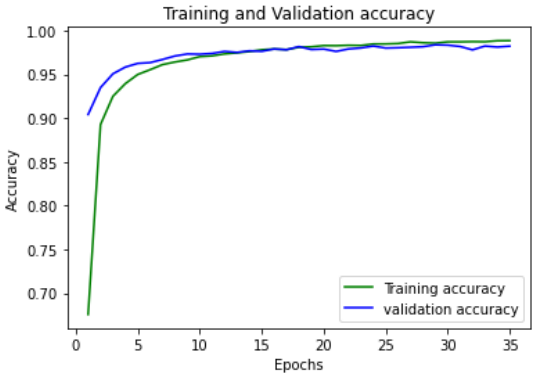

Fig 4.5.3 Graph of training and validation accuracies for 35 epochs

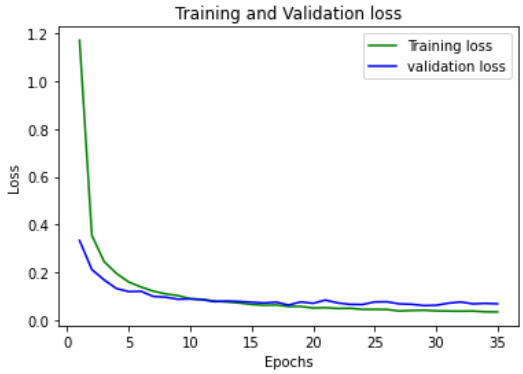

Fig 4.5.4 Graph of training and validation loss for 35 epochs

5. Discussion and Results:

5.1. Datasets

I used the Devanagari character dataset from the UCI Machine Learning Repository (Link - https://archive.ics.uci.edu/ml/datasets/Devanagari+Handwritten+Character+Dataset#).

It consists of 92,000-character images. These images are in grayscale format and have .png extension. Each image has a resolution of 32x32 pixels accounting to 1024 pixels or features per image. There 46 classes/characters in the dataset that are to be classified which includes 36 consonants from ‘ka’ to ‘gya’ and 10 digits from ‘0’ to ‘9’.

Below image consists of all the 36 consonants and 10 digits in the dataset.

Fig 5.1.1 Devanagari characters (36 consonants and 10 digits)

Below image shows some variations in handwriting the Devanagari character ‘ka’.

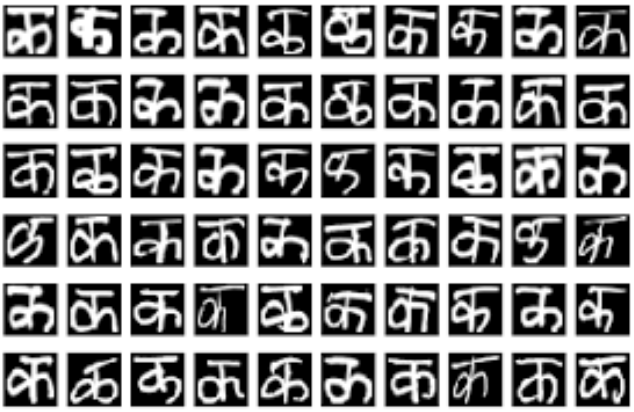

Fig 5.1.2 Handwriting variations

5.2. Evaluation Metrics

I used **accuracy score** from **sklearn.metrics** as of the evaluation metric for this classification problem.

$$\text{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i)$$

- $n_{samples}$ is the number of samples or instances in the test dataset.
- $\hat{y}_i$ is the predicted value of the i^{th} sample.
- y_i is the truth value of the i^{th} example.

5.3. Experimental results

I performed various experiments and tried of different data preprocessing techniques, hyperparameter optimization using Random search in this project. Number of features to retain after dimensionality reduction using PCA was based on the below graph of variance against number of components/features on the training data.

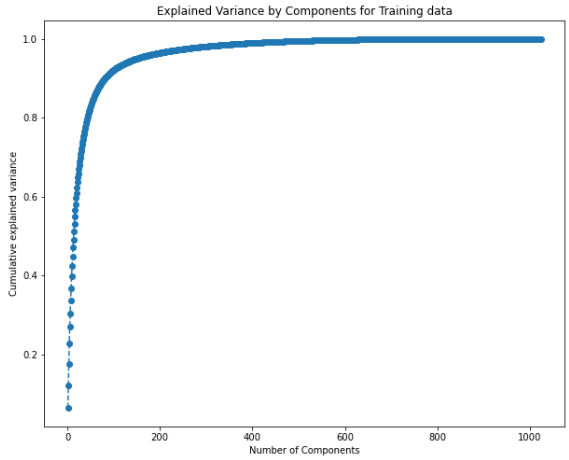

Fig 5.3.1 Variance by Components graph

Also, tried out t-SNE, UMAP and a combination of PCA + t-SNE and PCA + UMAP but the accuracy dropped after using these dimensionality reduction techniques. Due to the presence of characters in Devanagari which may appear similar depending on the person

to person writing variations, reduction in features leads to confusion in pattern recognition and thus reducing the classification accuracy.
Morphological operations such as erosion and dilation on the character images are shown in the image below.

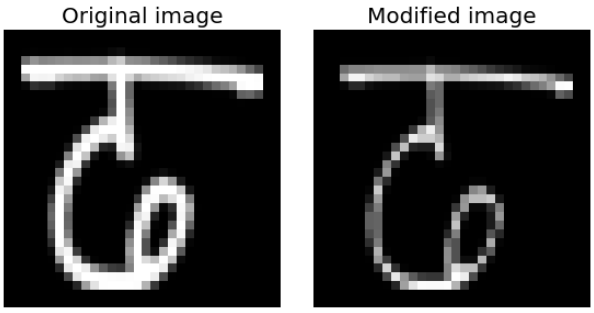

Fig 5.3.2 Thinning of a character

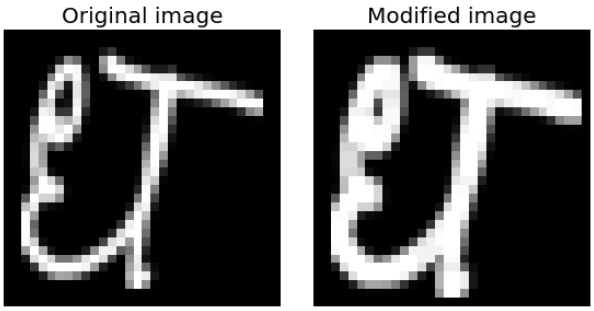

Fig 5.3.3 Dilation of a character

Thinning i.e., reducing the character area resulted in lower accuracy when compared to accuracy obtained on images whose areas were increased using dilation. For all the algorithms used in this project, classification accuracy scores improved when the image processing in the form of dilation was performed and then the flattened pixel intensity vectors were scaled.

Experimental Results for Classification:

Model	Preprocessing	CV accuracy score	Test accuracy score
KNN	Pixel intensity vectors (PIV)	90.19	91.40
	PIV, scaling, morphological dilation	92.86	93.28
SVM	Pixel intensity vectors (PIV), kernel = 'polynomial'	93.63	94.72
	Pixel intensity vectors (PIV), scaling, morphological dilation, kernel = 'polynomial'	94.78	95.71
Random Forest	Pixel intensity vectors (PIV), estimators = 100	89.93	88.81
	PIV, scaling, estimators = 1100	92.22	93.35
CNN	Pixel intensity vectors (PIV), scaling, label encoding	97.16	96.70
	Pixel intensity vectors (PIV), scaling, label encoding, with extra layers	98.59	**98.06**

6. Conclusion

Development of handwritten Devanagari OCR is still a challenging task in Pattern recognition area. In this report, I made a detailed analysis and comparison of 4 classification algorithms on the handwritten Devanagari characters dataset. These include K-Nearest Neighbors, Support Vector Machine, Random Forest, and Convolutional Neural Networks. The accuracy achieved was satisfactory and was about 98.06% using CNN on the test data of Devanagari characters.

This project allowed me to learn about some of the prevalent classification models used in handwritten Optical Character Recognition. I learnt that getting 100% accuracy in OCR of handwritten Devanagari characters is still a challenging task due to a large set of characters along with similar looking characters making the task more difficult. This problem is even more exaggerated when we add person to person handwriting variations to this problem.

6.1. Direction for future work

- Future work for this project could include recognition of complex conjuncts identification and categorization which are formed using modifiers, also called as 'matras'.
- Some Devanagari characters have similar shape and work can be done in that direction to improve their classification accuracy.
- Recognition of complete sentences.
- The project can be improved to work on a larger set of handwritten as well as printed document images.
- Inclusion of other Indic and non-Indic scripts in the proposed OCR.
- Determining context of the entire textual images.

7. References

[1] R. Rathi, R. K. Pandey, V. Chaturvedi and M. Jangid "Offline Handwritten Devanagari Vowels Recognition using KNN Classifier"

[2] A. Elnagar and S. Harous, "Recognition of handwritten Hindi Vowels using structural descriptors,"

[3] Holambe, A. K. N., Thool, R. C., and Jagade, S. M. (2012) "A brief review and survey of feature extraction methods for Devnagari OCR"

[4] B. Chakraborty, B. Shaw, J. Aich, U. Bhattacharya, and S. K. Parui, "Does deeper network lead to better accuracy: A case study on handwritten Devanagari characters,"

[5] M. Jangid and S. Srivastava, "Handwritten Devanagari Character Recognition Using Layer-Wise Training of Deep Convolutional Neural Networks and Adaptive Gradient Methods"

[6] https://towardsdatascience.com/pre-processing-in-ocr-fc231c6035a7