# HANDWRITTEN CHARACTER RECOGNITION OF DEVANAGARI SCRIPT

A DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE

OF

MASTER OF TECHONOLOGY
IN

**SIGNAL PROCESSING AND DIGITAL DESIGN**

*Submitted by:*

**SHILPA KAUR MANOCHA**
**2K19/SPD/16**

Under the supervision of

**Assistant Prof. Piyush Tewari**

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of Engineering)
Bawana Road, Delhi-110042

OCTOBER- 2021

# CANDIDATE’S DECLARATION

I, Shilpa Kaur Manocha, student of MTech (Signal Processing and Digital Design), hereby declare that the project dissertation titled “Handwritten Character Recognition of Devanagari Script” which is submitted by me to the Department of Electronics and Communication Engineering, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Master of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any degree, diploma associate ship, fellowship or other similar title or recognition.

Place: Delhi

Date: 13 / 10 /2021

Shilpa

**Shilpa Kaur Manocha**

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

# CERTIFICATE

I hereby certify that the project dissertation titled "Handwritten Character Recognition of Devanagari Script" which is submitted by SHILPA KAUR MANOCHA, 2K19/SPD/16 of Electronics and Communication Department, Delhi Technological University, Delhi in partial fulfilment of the requirement for the award of the degree of Master of Technology, is a record of the project work carried out by the student under my supervision. To the best of my knowledge this work has not been submitted in part or full for any degree or diploma to this University or elsewhere.

[illegible]

Place: Delhi **Asst Prof. Piyush Tewari**

Date: 13 / 10 /2021 **SUPERVISOR**

# ACKNOWLEDGEMENT

A successful project can never be prepared by the efforts of the person to whom the project is assigned, but it also demands the help and guardianship of people who helped in completion of the project.

With profound sense of gratitude, I thank Asst. Prof. PIYUSH TEWARI my Research Supervisor, for his encouragement, support, patience, and guidance in this research work. I could not have imagined having a better guide and advisor for my MTech study.

I am also grateful to Prof. N.S. RAGHAVA, HOD, Department of Electronics and Communication Engineering, DTU for providing the right academic resources & environment for this work to be carried out.

Last but not the least, my whole-heartedly thanks to all my family and friends who have patiently extended all sorts of help for accomplishing this undertaking.

Shilpa

**Shilpa Kaur Manocha**

# ABSTRACT

Optical Character Recognition [1] development has been gaining popularity in recent years, for Devanagari script along with other Indic Scripts. The script serves as a base for over 100 languages around the world including few popularly used scripts: - Hindi, Marathi, Sanskrit etc. Development of robust OCR systems for Devanagari script will allow us to preserve old manuscripts by converting the physical files to digital formats. This will make the process of storage, retrieval and transfer very convenient.

This project proposes the use of Convolutional Neural Networks as feature extractor for extraction of features from handwritten Devanagari characters. For classification, classifiers employed are SVM (Linear, Polynomial and RBF), KNN, RF, DT and MLP. Use of CNN model for feature extraction eliminates the need of handcrafted features by traditional pattern recognition methods. The experiments with these seven techniques have been done on the DHCD dataset proposed in year 2015. Use of CNN proved to be very effective for Devanagari characters recognition as all the models achieved recognition accuracy of over 93% and total training time including feature extraction and classification did not exceed a total of 12.16 minutes.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# CHAPTER 1

# INTRODUCTION

OCR (Optical Character Recognition) [1] is the technique of optically scanning the documents and images and then converting the text into machine editable format. The documents and images scanned may either contain printed text or handwritten text. After the conversion to digital form, the text can be stored for longer durations without degradation. The storage, retrieval and transfer of the data from the documents/images/manuscripts becomes very convenient as opposed to physically maintaining the files and records.

## 1.1 Handwriting Recognition

Handwriting recognition is one of the important applications of OCR systems. Handwriting recognition [2] is a major cross domain research area in the field of Image Processing, Pattern Recognition, Computer Vision, Machine Learning and now Deep Learning [3].

## 1.2 Classification of Handwriting Recognition Methods

Handwriting recognition problem can be broadly classified as offline and online [4] [5]. Offline handwriting recognition [4] is performed after the writing is completed by converting the handwritten document into digitally editable form. Example of offline character recognition is OCR system. Whereas Online character Recognition [5] is done on special devices where input is taken in real time using touchscreen or stylus like in a tablet. Our focus will be on Offline handwriting recognition techniques in this project.

The offline recognition systems can be used to extract and read any old documents, manuscripts, writings and documents, reading postal zip codes, bank cheque amounts, signature verification.

Further the techniques can be based on whether they use segmentation [6] to separate characters before classification or are segmentation free. We have studied in this project OCR systems based on segmentation approach. i.e., single characters are identified after segmenting lines into words and then words into characters.  The results are then combined to identify complete word. Such schemes are called as Handwritten Character Recognition (HCR). In case of segmentation free approach, the words are recognised as whole also known as Handwritten Word Recognition (HWR). Figure 1 shows the different types of Handwriting Recognition Methods.

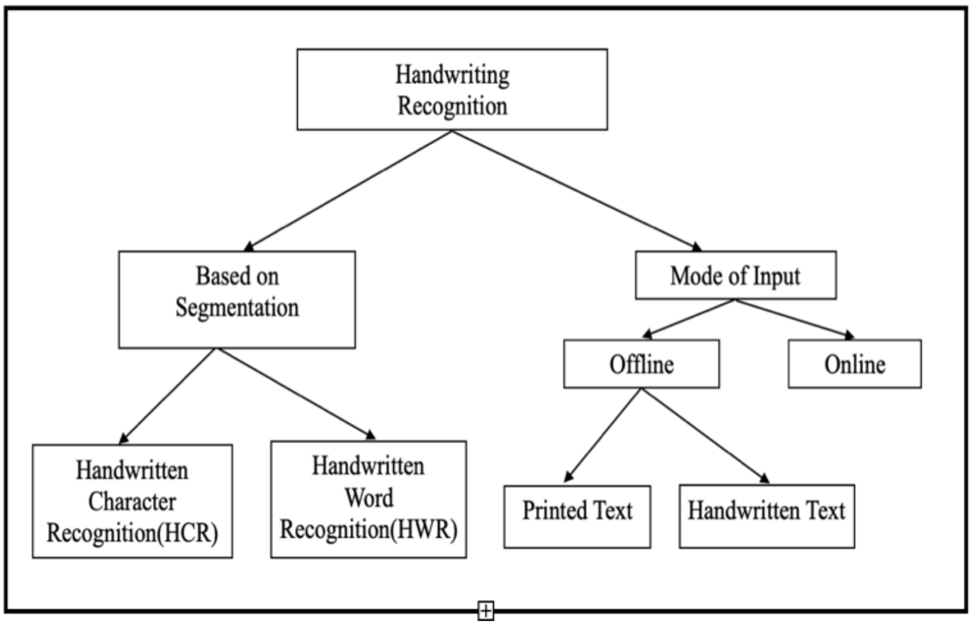


*Fig. 1.1. Classification of Handwriting Recognition Methods*

## 1.3 Applications of Handwriting Recognition

Some of the most widely popular domains where robust handwriting systems find applications include:-

i. Healthcare and medical aid*:* recognition of handwriting on patient forms & on prescription forms. Automatic processing of prescription forms and conversion to digital format will enable us to store the patient's health record and history and can be used in future diagnosis as well.

ii. Insurance firms*:* to process the documents and provide timely claims & enhance the user experience. Automating the complete documentation process of insurance purchase and claims will make the whole process a lot smoother and more convenient.

iii. In banks and finance sector*:* to verify signature [7] which speeds up the process and is an effective way of identifying forgery as opposed to manual verification/check. Systems can be developed to identify signature, cheque number, amount written in words and digits and complete process of reading and processing cheques can be automated.

*iv.* Manuscript Preservation: Easier and efficient storage and retrieval of old manuscripts [8] written Devanagari or related languages. This will allow preservation of cultural heritage in the form of ancient literature.

## 1.4 Challenges in Handwriting Recognition

There are several major challenges [9] [10] while recognising handwritten characters from documents or images starting from image acquisition up to classification. These include the degradation of old images due to noise or low quality of images to be scanned, huge variation in writing styles, difficulty in recognition of cursive handwritten characters, text not written in perfect straight lines leading to skew/slant/rotation/translation and non-availability of standard labelled datasets, to name a few.

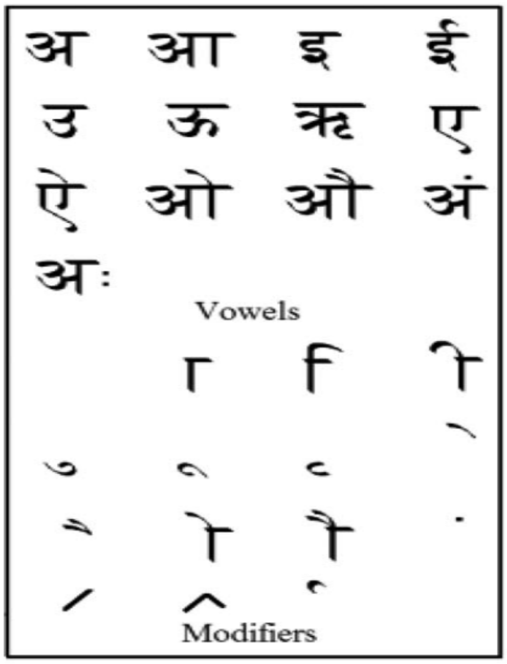
## 1.5Organization of the Report

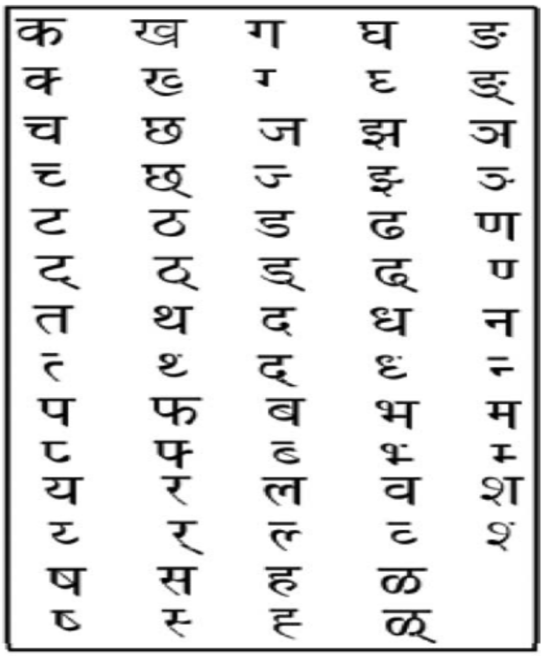
The report consists of the following sections. Chapter 2 describes about Devanagari Handwriting Recognition – the script, challenges, publicly available datasets and general system framework. Chapter 3 contains the literature review of state-of-the-art machine learning and deep learning techniques implemented by researchers for recognition of Devanagari handwritten characters recognition. Then chapter 4 discusses about the Proposed Recognition System and related theories in detail. The complete Experimental Setup is explained in chapter 5 and the results of the experiments are presented in chapter 6. Chapter 7 gives the conclusion drawn from the experiments. Chapter 8, the last section, points out future directions for enhancing research in this domain. Finally, the references are listed.

# CHAPTER 2
# DEVANAGARI HANDWRITING RECOGNITION

## 2.1 Devanagari Script

Devanagari script [11], also called as ‘nagari’, means the language from city of divinity; ‘deva’ meaning heavenly or divine and ‘nagari’ meaning city or abode. It is based on the family of Brahma scripts. It is fourth most widely used script around the world and forms the base for around 120 languages either in base form or derived form. The scripts that make use of Devanagari script either alone or along with other scripts are Sanskrit, Marathi, Hindi, Pali, Nepali, Prakrit, Sindhi to name a few. In India, it is used for languages like Hindi, Marathi and Sanskrit. The script consists of 47 primary characters - 33 consonants, 14 vowels and 10 numeral characters. Like the other Indian languages, it is also phonetic in nature i.e., the shape of each character represents unique sound. The script is written from left to right and is cursive in nature. There is no notion of upper case or lower-case characters. All characters / words have a horizontal line running at the top known as ‘shirorekha’. The script has similar structured characters differing only in dots, horizontal line, loops etc.





(a) (b)

Fig. 2.1. (a) Vowels and Consonants of Devanagari Script. (b) Consonants and their half forms in Devanagari script. [12]

## 2.2 Challenges in Devanagari Script Recognition

The Devanagari script [11] has various characters which have similar structures and differ only in minor details like dots, horizontal line, loops etc. This makes the classification of Devanagari characters very challenging. And on top of that different individuals may write the same characters differently due to cursive nature of the script. Table 1. shows few similar characters in Devanagari. Non-Availability of benchmark datasets of Devanagari script is yet another major issue. At present we have very limited number of Devanagari datasets as compared to English or other non-Indic scripts. There is a requirement of large and inclusive Datasets like we have for English. The number of datasets available are very less and are small in size for both isolated characters and words, when compared with those available for English.

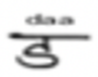
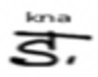
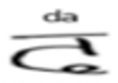
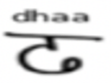
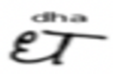
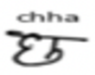
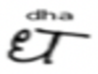
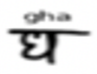
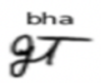
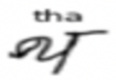
| | | |
|---|---|---|
|  |  | Characters differing only in single dot |
|  |  | Difference only in closed loop |
|  |  | Difference in style of 'Shirorekha'-header line and closed loop |
|  |  | Difference only in loop |
|  |  | Characters differ only in loop and curve |

Table 2.1: Similar Devanagari handwritten characters

## 2.3 Publicly Available Datasets - Devanagari Handwritten Character Database

Some of the popular Devanagari Handwritten characters datasets have been mentioned in table 2.

| Dataset Name | Year Proposed | Size | Types of characters | Number of Classes | Reference |
|---|---|---|---|---|---|
| - | 2020 | 5800 | Consonants, Vowels and Numerals | 58 | [13] |
| DHCD | 2015 | 92000 | Consonants and Numerals | 46 | [14] |
| CPAR -2012 | 2012 | 113400 | Characters and Numerals | 60 | [15] |
| CVPR ISI | 2009 | 22,556 | Numerals | 10 | - |

Table 2.2: Devanagari handwritten characters datasets

## 2.4 Devanagari HCR Framework

Figure 3 shows the general framework of a Devanagari Handwriting Recognition System.

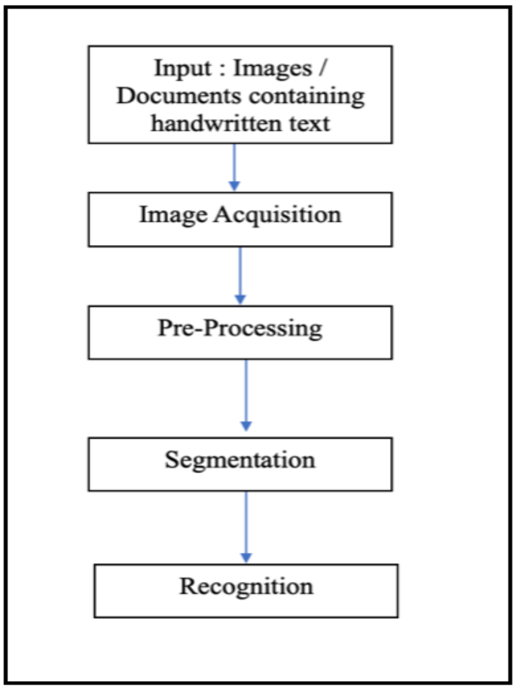


*Fig. 2.2. Offline DHRS Framework.*

Offline DHRS systems consist of four stages:

i. Image Acquisition: the handwritten characters to be recognised have to be acquired through scanners or cameras from images or documents .

ii. Pre-processing: It is done for the scanned image to make it suitable for further processing. The process includes steps like binarization, normalization, slant removal as well as noise removal .

iii. Segmentation: The scanned image/document is segmented into lines, words, and individual characters before applying to the model for training or testing in case of Handwritten Character Recognition.

iv. Character Recognition: this steps involves 2 stages. First stage is the feature extraction stage followed by Classification of characters into their correct classes based on the combination of features extracted for each character.

# CHAPTER 3

# LITERATURE REVIEW

## 3.1. Review of Machine Learning Algorithms based Recognition Systems

Handwriting Recognition Systems were developed using traditional methods like Fuzzy logic-based classification, Support Vector Machines (SVM), K-Nearest Neighbour (KNN), Hidden Markov Models (HMM) and other classifiers. The conventional methods required feature selection and Feature extraction stages to be done manually. The features could be structural, geometrical or global like closed loops, horizontal & vertical lines, inflection points, aspect ratio etc. of each character. In any recognition scheme, feature extraction is a critical step. Selection of appropriate features and their efficient extraction affects the recognition rate vastly. The features are applied as inputs to a classifier like SVM or HMM to recognise the text. The recognition accuracy and performance of these methods varies, and the systems are not robust as it is not possible to select best features always. Feature selection and extraction depends largely on the type of characters and dataset applied.

In [24], the authors made use of discrete cosine transform for extraction of features and Neural network-based classification for recognition of Devanagari characters. The NN used for experiments was Artificial NN with different number of hidden layers, starting from 10 up to 80 layers. The highest recognition accuracy obtained was 94.89% for 55 hidden layer architecture.

In [25], the authors classified 1000 Hindi characters by calculating the histogram of projections based on mean values, pixel values & vertical zero crossing of characters. Then the characters were classified with artificial neural network having 2 hidden layers. The classification accuracy obtained was 98.25%.

In [26], the authors have recognised Devanagari script characters by finding the gradient and curvature of characters and then classifying them using combination of two classifiers namely Modifier Quadratic Discriminant Function (MQDF) and Support Vector Machine (SVM). The recognition accuracy obtained was 95.13%.

In [27], the authors employed GLAC (Gradient Local Autocorrelation) feature extraction method and SVM classifier for recognition of Devanagari handwritten characters. Accuracies reported for ISIDCHAR and V2DMDCHAR datasets are 93.2% and 95.2% respectively.

In [28], the authors classified Devanagari handwritten characters by combining wavelet-based feature extraction method and Back Propagation NN classifier. The highest accuracy achieved was 70%.

In [29], the authors developed a recognition system for Devanagari script by evaluating various structural and geometrical features and performing classification using MLP classifier. Maximum classification accuracy achieved was 82.7%.

In [30], the authors developed a novel system for recognition of Devanagari script by evaluating structural features and performing classification using FFBPN, CFBPN and EBPN classifiers. Accuracy rates obtained were 97.20%, 97.46% and 98.10% respectively.

In [31], the authors classified handwritten Devanagari words' database having 10000 samples. The dataset was divided into 7000 characters for training and 3000 characters for testing. Features were extracted using strokes method and classifier used was Hidden Markov Model (HMM). Accuracy observed was 87.71% on the training dataset and 82.89% on the testing dataset.

In [32], authors proposed the use of curvelet transform for the extraction of textual features and classifiers used were SVM and KNN. KNN model produced best results with a maximum accuracy of 93.21%.

In [33], a hybrid feature extraction method has been employed by the authors, by combining skeleton and contour-based features. The feature maps were classified by SVM classifier. In [34], the authors used combination of directional and gradient structural curvature features for classification using SVM classifier.

| **Paper** | **Dataset** | | **Feature Extraction Method** | **Classifier** | **Recognition Accuracy** |
|---|---|---|---|---|---|
| | **Size of Testing dataset** | **Type of characters** | | | |

| | | | | | |
|---|---|---|---|---|---|
| [24] | 2760 | Devanagari online characters | DCT | NN | 94.89% |
| [25] | 1000 | Hindi words | Histogram of projections | ANN | 90% |
| [26] | 36,172 (28,937 training set size and 7,235 testing set size) | Devanagari consonants and vowels | Gradient and curvature of characters | Combination of MQDF and SVM | 95.13% |
| [27] | 36,172 | Devanagari handwritten characters (49 classes) | GLAC | SVM | 93.21% |
| | 20,305 | Devanagari handwritten characters (50 classes) | GLAC | SVM | 95.21% |
| [28] | 2000 | Devanagari | Wavelet based features | BPNN | 70% |
| [29] | 5375 (alphabets) and 3000 (numerals) | Handwritten Devanagari alphabets and digits | Structural and geometrical features | MLP | 82.7% |
| [30] | 40,000 (60% training, 20% testing and 20% validation) | Devanagari (offline) | Structural features | FFBPN, CFBPN & EBPN networks. | 97.20%, 97.46% and 98.10% respectively |
| [31] | 10000 (7000 training set and 3000 testing set) | Devanagari (offline) | Stroke method | HMM | 82.89% |
| [32] | 28,500 | Devanagari (offline) words | Curvelet Transform | SVM and KNN | 85.61% (SVM) and 93.21% (KNN) |
| [33] | 39,700 (training set size -22,500 and testing set size- 17,200) | Devanagari (offline) | Combination of skeleton and contour-based features | SVM | 79.01% |
| [34] | 39,700 (training set size -22,500 | Devanagari (offline) | Combination of DDD and GSC features | SVM | 88.75% |

| | | | | | |
|---|---|---|---|---|---|
| | and testing set size- 17,200) | | | | |

Table 3.1: Recognition Results of State of the art Machine Learning based Devanaagri Recognition Systems

## 3.2. Review of State-of-the-Art Deep Learning based Recognition Systems

With the emergence of deep learning [3] methods, major increments in accuracy rate have been observed in handwriting recognition. The models can extract the features automatically from the dataset so the need of feature selection by hand engineering has been eliminated completely. Thus, the feature extraction and character classification stage are combined into a single stage- character as recognition stage.

Deep learning models are being used since early 2000's for applications ranging from object detection, pattern recognition, computer vision and image processing. Various researchers have also shown that using deep learning models we can get excellent results in handwriting recognition. The Neural Network models mimic the learning pattern of human brain. Convolutional Neural Network [35] is one of the popular NNs. It consists of 3 types of layers- first is convolution layer that has filter banks, and it convolves with input images to generate features maps. Next is Pooling layer which performs average or max pooling to reduce size of feature maps. These 2 stages combined are for feature Extraction. Fully Connected layer activates every neuron from last layer and performs the task of classification. Due to its properties like weight sharing, multiple layers and pooling, CNN can detect features automatically. But the model requires large datasets for training and considerable amount of time. Many researchers have explored the options and state of the art methods like pre-trained architectures, transfer learning, use of additional classifier, regularization methods, other techniques to avoid problem of overfitting/under-fitting, use of different optimizers and so on. We have discussed those experiments and results in the following paragraphs.

In [36], the author has made use of transfer learning scheme using AlexNet [37] which is trained over a database of 16870 images consisting of 22 more frequently used consonants. Dataset has been divided as 56%- training, 24%- validation & 20%-testing for evaluation of performance. A 3D filter has been added to the first stage of Network to convert grayscale images of size 220*220 to coloured images of size 227*227. The results demonstrated that Transfer Learning is a faster way of achieving training with fewer training samples. In 3 epochs only, the training accuracy went above 90%. The highest validation accuracy and test accuracy attained are 94.49 % and 95.46% respectively.

In [38], the authors have proposed a new dataset named -DHCD (Devanagari Handwritten Character Dataset) consisting of 46 different characters of Devanagari script. The author has focused on the use of dropout and dataset increment methods to improve the test accuracy. The evaluation has been done on both shallow and deep models - Model1 and Model2 respectively. After training for 50 epochs, the accuracy remained nearly constant. The results show that Deep architecture's training accuracy improves with extended dataset (98.47%) and almost no improvement with dropout. On the contrary, the shallow architecture depicts increase in training accuracy with dropout and only a slight change with extension of dataset (98.26%).

In [39], the authors have proposed a new dataset named as UCI Devanagari dataset and has made it publicly available. DHCRS (Devanagari Handwritten Character Recognition System) has been proposed specifically for Devanagari characters. The scheme comprises of 2 stages- In first stage, pre-trained VGG16 [39][40] architecture has been implemented followed by next stage of newly created Fully Connected layer. The first stage network is wide with a greater number of parameters as opposed to the compact network of 2nd stage. The fine tuning of model in 2nd stage increases the overall efficiency of the model. Techniques like Runtime data augmentation and Regularization techniques like – 'dropout' and 'batch normalization' are also incorporated to avoid over-fitting. The results were also impressive with 96.55% recognition accuracy with training loss 0.12 on the UCI dataset. Thus, the model proposed provides considerably good results with fewer number of training parameters and a small dataset.

In [41], Layer wise trained Deep convolutional neural network has been used to train 6 network architectures (NA). The 6 Network architectures used to evaluate performance are namely NA1 to NA6. The optimizers used are SGD, Adagrad, Adam, AdaDelta, AdaMax, and RMSProp. The Layer-wise trained Deep Convolutional NN have showed highest recognition accuracy & faster convergence rate on all the 6 architectures when compared with other state of the art DCNN methods. Highest recognition accuracy has been observed for NA-6 architecture after layer wise training and using RMSProp optimizer. Thus, the paper

[14] summarized the effects of choosing various optimizers and architectures in an extensive manner.

In [42], DevNet-Handwritten Devanagari Character Recognition has been developed specifically for enhancing performance and the recognition accuracy of Devanagari handwritten characters and numerals. The model has been based on CNN architecture since it is very powerful and capable of extracting relevant features. To prove the efficacy of the proposed model it has been implemented on four different databases varying in size and collected from different sources. The proposed method proved to be highly accurate on all databases used and the accuracy came out to be greater than 97% in all cases (Refer Table 1).

In [43], use of Residual Network (ResNet) has been made to classify the Devanagari characters of DHCD dataset. The ResNet is able to overcome the problem of degradation observed in deep CNN owing to presence of large number of layers. The ResNet uses short connections which reduces number of layers while still providing good results. The two that were employed are ResNet 34 and ResNet 50, where 34 and 50 denote the number of layers in the network. The highest recognition accuracy achieved was 99.35% for ResNet50 and 98.73% for ResNet30. ResNet therefore can be used in case of deep learning approaches to avoid the problem of vanishing gradients and thus degradation of performance due to deep architecture.

In [44], the authors have proposed the model DeepNetDevanagari for recognition of Devanagari characters. The model is based on CNN architecture. The model has been trained on a small database created for testing the efficacy of proposed method. The database has handwritten characters of Devanagari in base form. Even with a small dataset, the model was able to recognize the characters with 93.73% accuracy.

In [45], Devanagari numeral characters of database ISI Devanagari characters are recognized. The authors have used CNN architecture for feature extraction and SVM as classifier. The model produced excellent results with up to 99.41 % accuracy.

In [46], 7 pre-trained architectures have been used - AlexNet, DenseNet(DenseNet-121, DenseNet-201), VGG(VGG-11, VGG 16, VGG 19) and Inception-V3. Usage of pre-trained models eliminates the requirement for training the model from scratch. The work focused on examination of different pre-trained models for the DHCD dataset. The effects of choosing pre-trained models on training time and accuracy have been analysed after performing experiments. The number of epochs were kept low, and the focus was on reducing the number of epochs as well as average time per epoch for training and recognition. The Inception [46][47] model gave the accuracy of 99% in single epoch only while the AlexNet model required in 3 epochs to attain best accuracy. The DenseNet [46][48] took the greatest number of epochs to reach accuracy of 89% and 90%.

| **Paper** | **Dataset** | | | | | | **Recognition Technique** | **Recognition Accuracy** |
|---|---|---|---|---|---|---|---|---|
| | **Name** | **Size of dataset** | **Training Data size** | **Testing Data size** | **Number of Classes** | **Types of Characters** | | |
| [36] | - | 16870 | 9447 | 3374 | 22 | Consonants | Transfer Learning using AlexNet | 95.46% |
| [38] | DHCD | 920000 | 78200 | 13800 | 46 | Consonants and numerals | Deep CNN - with focus on dropout and dataset increment | 98.47% and 98.26% |
| [39] | UCI Devanagari Dataset | 5800 | 4640 | 1160 | 58 | Consonants, vowels and numerals | DCHRS- 2 Stage VGG Architecture -Fine Tuning with Deep CNN | 96.55% with 0.12 training loss |
| [41] | ISIDCHAR | 36172 | - | - | 49 | Consonants and vowels | Deep CNN - trained Layer Wise | 97.30% |
| | V2DMDCHAR | 20305 | - | - | 50 | Consonants and vowels | Deep CNN - trained Layer Wise | 98% |
| [42] | UCI DCD | 920000 | 78200 | 13800 | 46 | Consonants and numerals | DevNet: Modified CNN for Devanagari | 99.54% |
| | CVPR ISI | 22000 | 18784 | 3772 | 10 | Numerals | DevNet: Modified CNN for Devanagari | 99.63% |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | CMAT Edb 3.2.1 | 3000 | 2000 | 10000 | 10 | Numerals | DevNet: Modified CNN for Devanagari | 98.70% |
| | Kaggle Devanagari character dataset | 12992 | 10350 | 2642 | 58 | Consonants, vowels and numerals | DevNet: Modified CNN for Devanagari | 97.29% |
| [43] | DHCD | 920000 | 78200 | 13800 | 46 | Consonants and numerals | Residual Network(ResNet) | 99.35% |
| [44] | - | 5484 | 4113 | 1371 | 33 | consonants, vowels in base form | CNN | 93.73% |
| [45] | ISI Kolkata Devanagari Dataset | 22556 | - | - | 10 | Numerals | CNN (feature extractor) +SVM (classifier) | 99.41% |
| [46] | DHCD | 920000 | 78200 | 13800 | 46 | Consonants and numerals | AlexNet with Deep CNN | 95% |
| | DHCD | 920000 | 78200 | 13800 | 46 | Consonants and numerals | DenseNet with Deep CNN | 90% |
| | DHCD | 920000 | 78200 | 13800 | 46 | Consonants and numerals | VGG with Deep CNN | 97% |
| | DHCD | 920000 | 78200 | 13800 | 46 | Consonants and numerals | Inception with Deep CNN | 99% |

Table 3.2: Recognition Results of State of the art Deep Learning based Devanaagri Recognition Systems

# CHAPTER 4

# PROPOSED DEVANAGARI HANDWRITTEN CHARACTER RECOGNITION (DHCR) SYSTEM

## 4.1. Objective

Development of a robust handwritten character recognition system requires the selection of best feature extraction algorithm. Since there are numerous feature extraction methods available, it becomes difficult to choose the suitable algorithm. The aim of proposed method is to save the hassle and struggle of selection & extraction of relevant features from handwritten characters and make the process completely automatic. While doing so, the classification accuracy must also increase. The training time and prediction time must also be a reduced value.

Thus, the motivation behind proposed method is to

(i) Minimise the requirement for image pre-processing.

(ii) Eliminate the need of hand engineered feature selection and extraction.

(iii) Improve accuracy over other recognition systems available.

(iv) Reduction in training and prediction time.

## 4.2. Proposed Recognition System Framework

The proposed system architecture is described below. Various steps and related theories are explained in upcoming sections. Figure 4 shows the framework of the proposed recognition system.

Loading the dataset:

The dataset consists of isolated characters of each character class in separate folders. The raw pixel information is available in a csv file. Each character is of size 32*32 i.e., having 1024-pixel values. The character labels have also been mentioned for all characters. Input features (1024) are read into one array and character labels in another array.

Image Pre-processing:

The pixels contain information of the gray scaled images. The images are converted to binary. Normalisation is performed for all features by dividing them by 255(highest pixel value in black and white image). Thus, we get features normalised to a scale of 1. The features and character label arrays are split into training and testing set.

## Feature Extraction using CNN:

CNN automatically extracts relevant features from the data fed. Convolution layers are used to create feature maps from training samples and pooling layers are used for reducing the size of feature maps generated. Instead of connecting the last fully connected dense layer for classification, the features are obtained in an array and fed to different classifiers as input. The model is trained for 3 epochs to learn the features from the training samples. Figure 2 shows the CNN model architecture used for the purpose of feature extraction.

## Classification:

Features extracted from last layer of CNN model are fed to various classifiers and test samples are applied for prediction. Classifiers are fed with two arrays, X - input features obtained from last/Dense layer of CNN and y-class labels for training and then tested on test samples. Total number of character classes are 46. Default parameter values have been used for each classifier for simplicity. Classifiers employed are Support Vector Machine(SVM), K- Nearest Neighbours(KNN), Decision Tree(DT), Random Forest(RF) and Multi-layer perceptron(MLP).

## Performance Evaluation:

For performance analysis, metrics used are classification accuracy and prediction time, for all classifier models. Since the dataset is balanced, classification accuracy is a reliable parameter and there is no need for evaluating precision, recall or F1-score values.

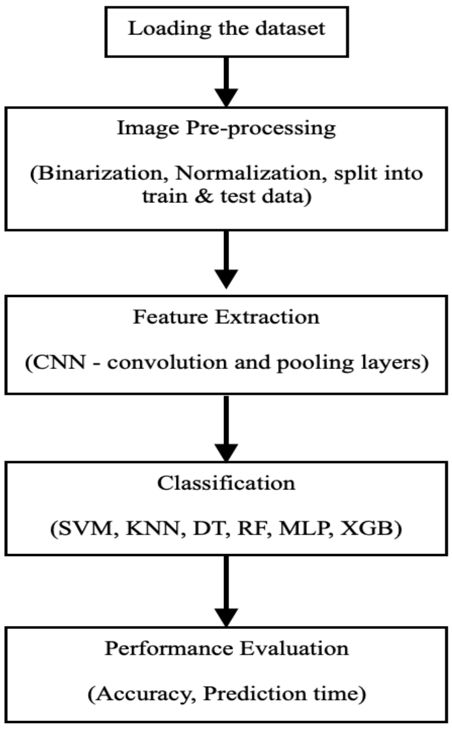

Loading the dataset
Image Pre-processing
(Binarization, Normalization, split into train & test data)
Feature Extraction
(CNN - convolution and pooling layers)
Classification
(SVM, KNN, DT, RF, MLP, XGB)
Performance Evaluation
(Accuracy, Prediction time)

*Fig. 4.1. Framework of the Proposed Recognition System*

## 4.3. CNN Model

Convolutional Neural Networks [49] are popularly used for image classification, computer vision and object detection tasks. The Network is inspired by the learning model of neurons of human brain and requires very less pre-processing of input images. They consist of five different types of layers:

i. Convolution layers: - which perform the operation of convolution of input images with filter or kernel and generates feature maps. The filters slide over the input image for performing the mathematical operation of convolution, by finding dot product of filter value with image pixel value. The feature maps give information about text in images like it may detect lines, corners, edges etc. The feature maps are large due to multiple convolutions with the entire image.

ii. Pooling Layer: - down samples the size of feature maps. It reduces the dimensions of features by performing pooling or aggregation operation. The most commonly and widely used pooling methods are average pooling and max pooling. Average pooling takes average pixel value from a section of input image and rest pixel values are discarded. Similarly, in max pooling operation, maximum value from nearby pixel values is retained. This is done to reduce computational costs. The convolution layers and pooling layers are together called as Feature Extractors.

iii. Dropout layer: - If all learned features are connected to the FC layer, the model may learn all necessary and unnecessary features resulting in overfitting on the training dataset. To avoid any such issue, we can add one or multiple dropout layers which randomly drops/ removes specified percentage of neurons. This also results in reduced size of network model. A dropout layer can be inserted after convolution layer or pooling layer.

iv. Flatten layer: - this layer flattens the feature maps obtained from convolution and pooling layers and converts them into a 1-dimensional feature map. This flattened 1D vector is fed to fully connected layer/s for classification.

v. Fully Connected layer: - consist of weights and biases and units/neurons which are used to establish connections between two layers. These are the layers responsible for performing the classification of dataset images into respective classes. They take the feature maps from previous layers as inputs and based on the combination of features learned, this layer can classify the samples correctly. There may be multiple fully connected layers in the network.

| Layer type | Functions | Parameters | Input | Output |
|---|---|---|---|---|
| Convolution Layers | ▪ Convolution of filters with images to generate feature maps.<br>▪ Filters are made of kernels having 1 bias per filter.<br>▪ Activate all values in feature map | ▪ No. of Kernels.<br>▪ Size of kernels (width and height)<br>▪ Activation function<br>▪ Stride size<br>▪ Padding value<br>▪ Value & type of Regularization | 3D array, prior feature maps. | 3D array, 1- 2D feature map for every filter. |
| Pooling Layers | ▪ Dimensionality Reduction.<br>▪ Extract average or max. pixel value from a region.<br>▪ Sliding window approach. | ▪ Stride size<br>▪ Window size | 3D array, prior feature maps. | 3D array, 1 2D map per filter, reduced spatial dimensions |

| | | | | |
|---|---|---|---|---|
| Fully connected layers | ▪ Summation of information from all feature maps.<br>▪ Classification into classes. | ▪ No. of nodes<br>▪ Activation function : if aggregating info, use ReLU, for final classification use SoftMax | Flattened 3D cube, previous set of feature maps. | 3D cube, one 2D map per filter |

Table 4.1: Types of layers in a CNN model.

Besides these layers, activation function plays an important role. An activation function adds non-linearity to the model. Commonly used activation functions are ReLU, sigmoid, tanh and sigmoid. We have used ReLU activation function in all convolution layers and in the output layer the 'SoftMax' activation function is used for the multi-class classification task. The 'SoftMax' activation used gives a probability for each class and the model will makes prediction based on the class with highest probability. The model learning happens because of loss function and optimizer. The loss function tells the model its task. e.g., to reduce the mean absolute error between predicted class label and the actual class label. The loss function employed is Cross-Entropy. The optimizer tells the network on how to achieve that aim. Most popular optimizers are Adam, AdaGrad and RMSProp that do an excellent job of updating weights in an adaptive manner and thus improving accuracy of model. The optimizer employed is 'Adam' optimizer.

<u>ReLU Activation function:-</u>

The Rectified Linear Unit is one of the most commonly used activation functions. It can be represented mathematically as: f(x) = max (0,x) .i.e., it reduces the negative values to zero and returns the original value for positive numbers. Though in formulation the function is very simple, yet it generates excellent results when used in deep models for introducing non-linearities. It helps the models to account for interactive effects and non-linear effects well.

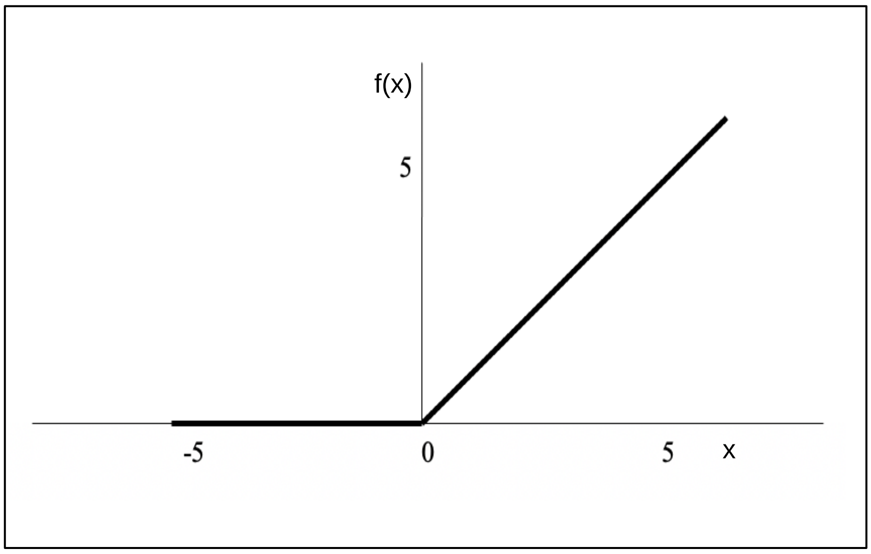


*Fig. 4.2. Graphical Representation of the ReLU function*

SoftMax Activation function:-

SoftMax activation is used in the output layer of a multi-class classification model. It normalizes the outputs of each class by converting their weighted sum values to probability values. The probabilities of all the classes add up to value one. The classification is then done by the classifier based on class probabilities. The input is classified into the class having maximum probability.

Adam Optimizer: -

Adam optimizer is an improvement over stochastic gradient descent (SGD) algorithm, used or updating the network weights in an iterative manner. It is widely used in training of deep learning models. It combines the best features of AdaGrad and RMSProp optimization algorithms. Besides it is simple to implement, has very less memory requirements and works well with (non-stationery) noisy or large data problems.

Loss function: Categorical Cross-Entropy: -

Categorical cross entropy loss function is used in a multi-class classification task, where the output can be classified into one of the multiple class labels available. The model uses cross entropy function and assigns a high probability to the correctly classified class label and a low probability to other classes.

We have built a CNN model and trained it to learn best features from the images and for the purpose of classification, various classifiers have been used instead of the fully connected layer in CNN.

## 4.4. Brief Explanation of the Classifiers Used

1. *Support Vector Machine (SVM): -*

Support Vector Machine [50] is a supervised machine learning method used for classification and regression tasks. SVM is a popular and widely used classifier for image classification tasks. It distinguishes the sample points by identifying a hyperplane that separates the classes. The sample points that are nearest to the hyperplane are called "support vectors". The classifier aims to maximise the margin (difference between support vectors) on either side of hyperplane. The classifier takes 2 inputs: an array of shape (number_of_samples, number_of_features) and another array containing class labels, either as numerical or string format, of shape (number_of_samples). A kernel function has to be specified which can be linear, polynomial, radial basis function or any other custom function.
Different Kernel types used in the project are: -

I. Linear Kernel: - {x, x'}

II. Polynomial (Poly): - {γ (x, x') + r) ^ d, where γ- gamma, d is degree, r- coefficient.

III. Radial Basis Function (RBF): (−γ‖x−x′‖2), where γ is gamma (> 0).

The classifier has following advantages: - powerful in high dimensional spaces, effective in cases where number of dimensions are more than the number of examples, is memory effective as it only focuses on a subset of training points i.e., the support vectors and is versatile i.e., different kernel functions can be used.
The disadvantages of the classifier are overfitting may occur in cases where number of features are greater than the number of sample points, selection of kernels is difficult, and calculation of scores & probabilities is done using expensive k-fold cross-validation.

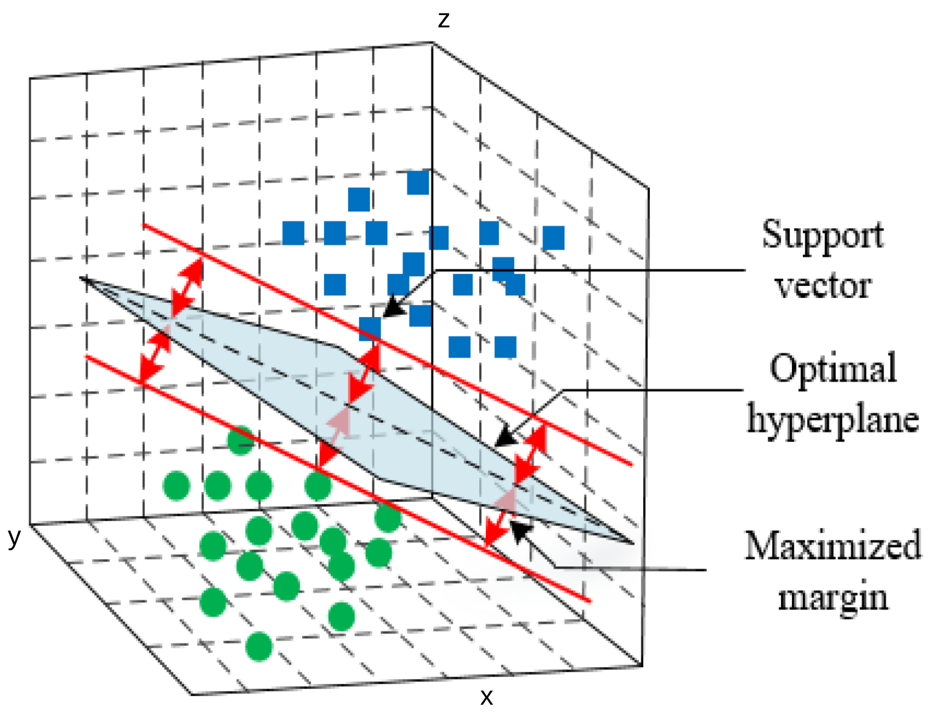


*Fig. 4.3. Graphical Representation of the SVM Classifier [55]*

2. *K- Nearest Neighbours (KNN): -*

KNN [51] is a supervised learning method. The idea behind this algorithm is that similar things occur in proximity. Closeness between points is measured using distance between the points ex. by using Euclidean distance. It finds distance between the query and all data points and selects a specified number of samples (k) closest data points from query. The query is classified to the class label that is the most repeating in the chosen k points group. The advantages of using KNN are: - easy and simple to implement, no need of tuning parameters, can be employed for classification as well as regression tasks. The disadvantage of algorithm lies in the slow classification time for large number of samples/classes. KNN algorithm is a popular choice in recommender systems, as they work on principle of close data points.

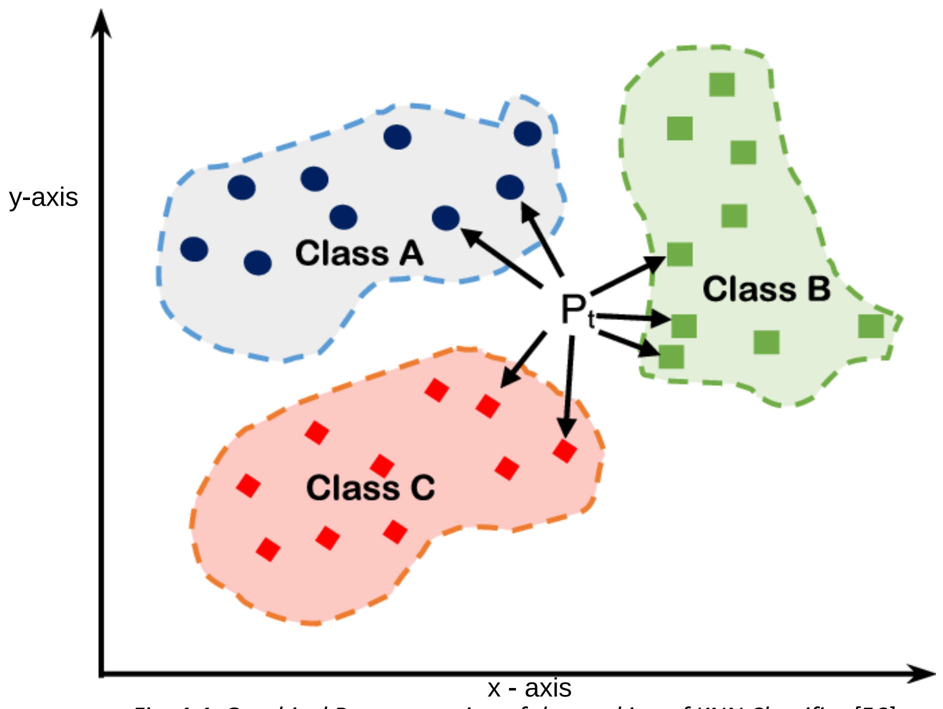


*Fig. 4.4. Graphical Representation of the working of KNN Classifier [56]*

3. *Decision Tree (DT): -*

Decision Tree [52] is a powerful classification and prediction tool that learns using an inductive approach. It has a tree structure like that of a flowchart. In a decision tree, there are several nodes, that present a decision condition, starting from parent node at the top to the terminal node as the final class label. At each node, the data is classified based on a test on the attribute and each branch represents the result of the test/decision. A DT is built by recursively splitting the data samples into subsets based on the attribute tests at each node. The splitting is stopped when we have the final predicted classes at the final nodes. An optimal DT is one that is able to represent all data points with least number of node levels. Pruning methods are used to drop nodes that are not critical for classification, hence reducing size of DT. Use of DT has following advantages: - they can handle both categorical and numerical values, no prior domain knowledge requirement, can be visualised, results can be interpreted and analysed to understand critical decision rules. DT have following limitations: - may lead to overfitting on data by creating complex tree, problem of variance i.e change in tree with even slight changes in data points, no guarantee of optimal tree by use of greedy algorithms and it results in a biased tree if some classes in the dataset dominate.

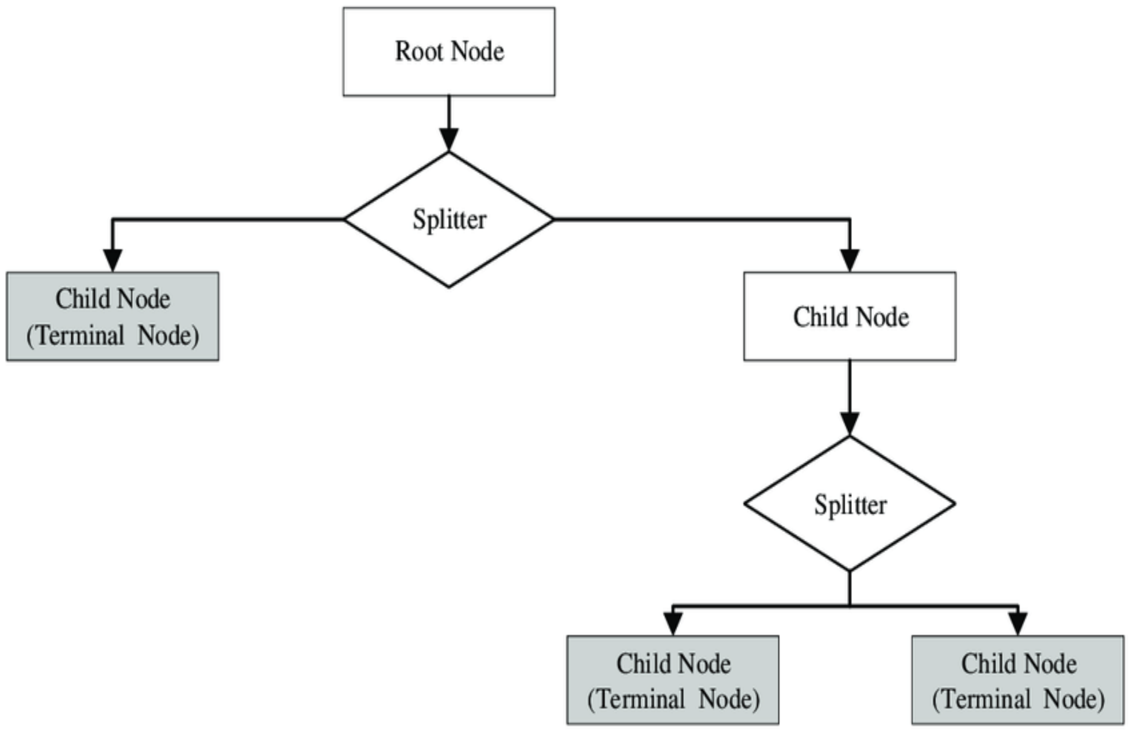

Root Node
Splitter
Child Node (Terminal Node)
Child Node
Splitter
Child Node (Terminal Node)
Child Node (Terminal Node)

*Fig. 4.5. Structure of a Decision Tree [57]*

4. *Random Forest (RF): -*

Random Forest [53] is a supervised learning algorithm that combines decision tree classifier with the bagging method. A forest is a set of decision trees, that collectively vote for the correct label. The model builds multiple trees by randomly a subset of features for each tree. Thus, multiple trees of reduced depth are built, and overfitting is reduced. The results from subtrees are averaged to reach to final class label. The classifier also makes it possible to measure the importance of features on a scale of 0-1. Important features are those having higher importance value. The classifier doesn't require much hyper-parameter tuning, is simple and easy to build, reduces chances of overfitting and mostly results in good accuracy for both classification as well as regression tasks. The training of model doesn't need much time, but prediction takes some time. In case of large number of trees, model gives higher accuracy, but the computations become slow.

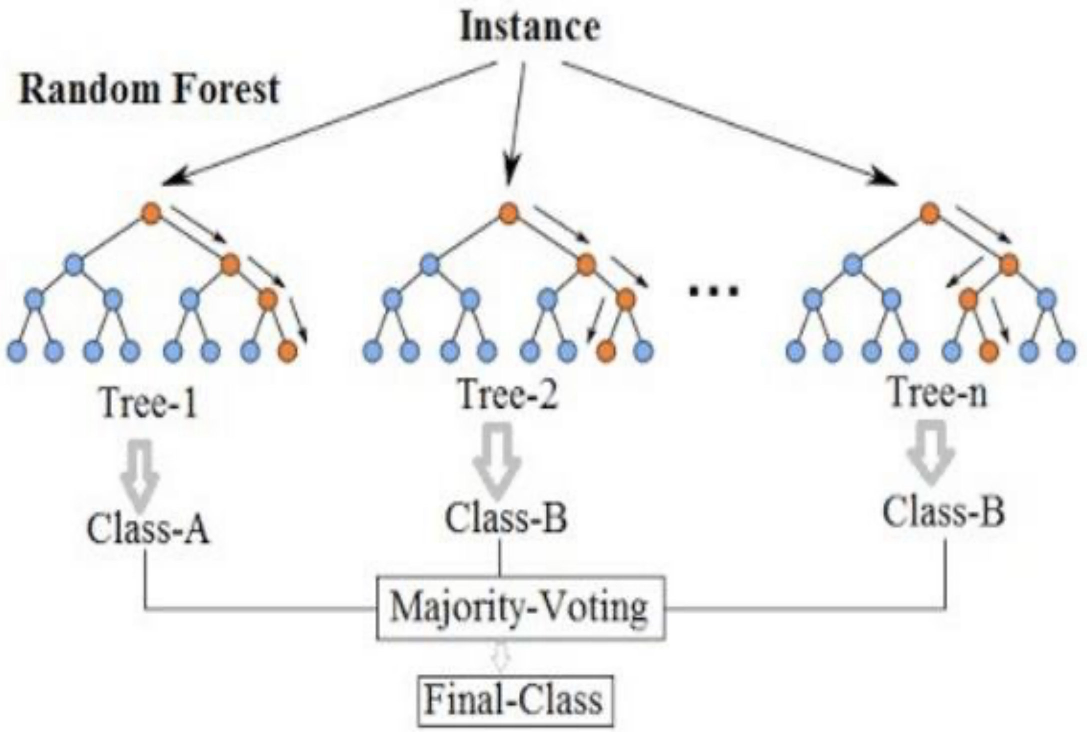


*Fig. 4.6. Structure of Random Forest Classifier [58]*

5. *Multi-Layer Perceptron (MLP): -*

Multi-layer perceptron [54] is a deep, artificial neural network (ANN). It has 3 types of layers- 1 input layer, multiple hidden layers and one output layer. The input layers relate to output layers as directed graph. The classifier trains by method of back-propagation. It can be used as a classifier or a regressor. The model minimises cross entropy loss function for classification. For multi-class classification, SoftMax activation function is used. The advantages of using MLP are: - it can understand nonlinear models & can learn models in real time. The limitations include requirement of tuning of hyper parameters such as the no. of hidden units, no. of layers and no. of iterations, difference in validation accuracy with different random weight initialisations due to non-convex loss function of hidden layers. The model is also sensitive to feature scaling.

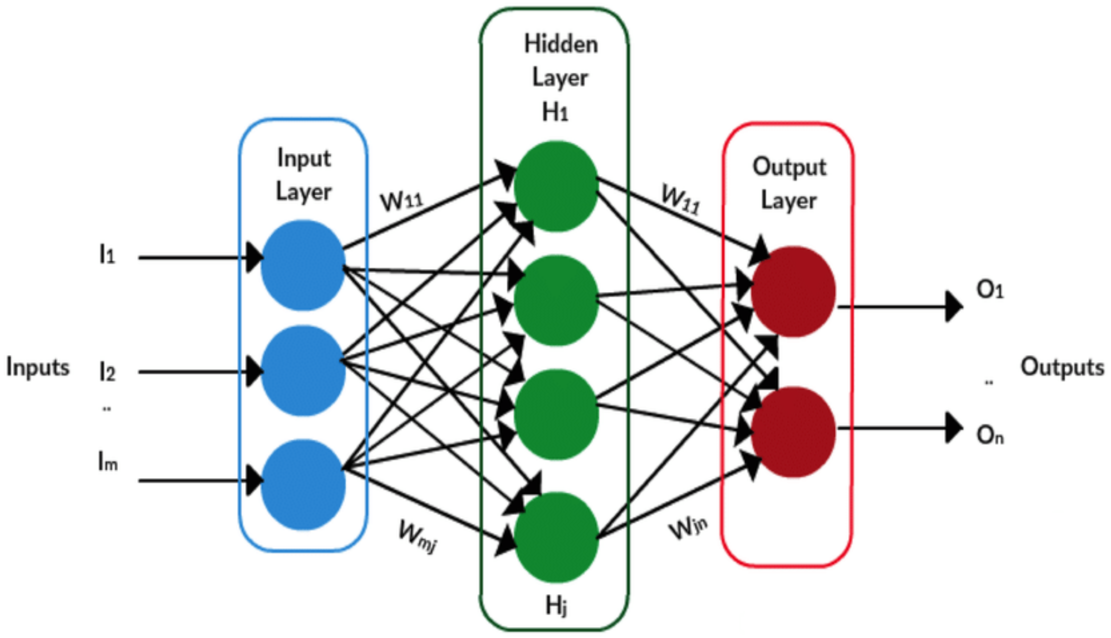


*Fig. 4.7. Multi-Layer Perceptron Classifier [59]*

## 4.5.Performance Evaluation Criteria

For recognition of handwritten characters, the most widely used evaluation parameters are accuracy and time required for training the model. Since the dataset is balanced, accuracy proves to be an effective evaluation metric.

1. Classification Accuracy: It is the ratio of correct predictions made to the total number of test samples. High accuracy percentage is the desired outcome.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions made}}$$

2. Prediction Time: The time required by the classifier to segregate the test samples into different classes, after they have been fed with input features to be classified. Lowest prediction time for correctly identifying the classes is the desired outcome.

# CHAPTER 5

# EXPERIMENTAL SETUP

The proposed work was implemented in Anaconda Jupyter Notebooks, and Python 3 was used as the programming language. The complete experimental setup is described below stepwise:

## 5.1. Installation of libraries:

Following libraries have been used in the project: -

Data Pre-processing libraries: - Pandas, NumPy

Pandas: Pandas [60] stands for "Python Data Analysis Library". It creates a python object called dataframe that provides an easier way to read and represent data, from a csv or tsv file, in the form of rows and columns. The library is used for converting a list, dictionary or NumPy array to a Pandas data frame. It can be used to open a file, usually a CSV file. Storing the data in a tabular form allows easier manipulation. We can also get statistics on the dataframe like mean, correlation, median, count, max, min, std. which helps in understanding the data in a better way. The library can be used for selection of data, filtering, sorting & grouping of data, cleaning of data, combining or joining of data frames. Pandas functions used in the project include pandas.read_csv ("File_path") - for reading the csv file of dataset, pandas.groubpy() – for grouping the values by "character" count.

NumPy: NumPy [61], stands for "Numerical Python", and is a array processing package. The library is used for performing mathematical & logical operations on multidimensional arrays. The arrays can be arranged, reshaped or flattened. We can also create arrays having all zeroes, or a specific number. Elements of array can be operated on using unary, binary or universal functions pre-built in the NumPy library. Various sorting methods are also available.
NumPy functions used in the project include numpy.array(), shape, reshape().

Model building libraries: - Sklearn, Keras, TensorFlow

Sklearn: Scikit-learn or sklearn library [62] is used of building machine learning models in Python. The library provides tools for performing machine learning and statistical modelling tasks like classification, clustering, regression & dimensionality reduction. Classifiers like K Neighbors Classifier, SVC, Random Forest Classifier, MLP Classifier, Decision Tree Classifier, XGB classifier have been imported from the sklearn library in the project.

TensorFlow: TensorFlow [63] is a symbolic math library used for creating neural networks and deep learning models. The library is flexible and allows distributed computing. Though it is a very powerful library but difficult to use for creation of deep neural models.

Keras: is one of the most popular and easy to use python library for building deep learning models. It has been built on top of deep learning libraries like TensorFlow, Theano, Caffe etc. The library has a very minimal structure which makes it a favourable choice for deep learning applications. It supports multiple platforms and runs on both CPU & GPU. From Keras [64] , following imports have been made in the project: Sequential from keras.models, layers like Dense, Conv2D, MaxPool2D, Flatten, Dropout from keras.layers and the function to_categorical from keras.utils.

Visualization Libraries: - Matplotlib

Matplotlib: is a visualization library used in Python for plot of arrays. It consists of several types of plots:- line_plot, bar_plot, scatter_plot, histogram etc. The plots allow one to recognise and analyse patterns and trends among the data. Matplotlib.pyplot [65] is used in Python that works like MATLAB and helps in creating figures, plots and graphs. In the project matplotlib.pyplot is used to display results in the form of bar plots.

## 5.2. Dataset Preparation:

For Experimental Analysis, we have made use of the DHCD [38] dataset, proposed by Acharya et. al, in 2015. The dataset includes 46 handwritten Devanagari characters - 36 alphabets and 10 numerals. Each character label has 2k image samples. i.e., a total of 92k samples. The dataset is divided into two parts: training set -to train and extract the features from CNN and testing set - fed to classifiers for testing the performance of classifier.

We have considered 3 cases of dataset splitting for our experiments. Case 1) 70:30, Case 2) 75:25, Case 3) 80:20.

**Algorithm:**

1. The dataset is read into a Pandas data frame.
2. The dataset is segregated such that the pixel values of each character are stored in variable X and the character class labels are stored in variable Y.
3. Variable ‘n_classes’ stores the number of different character classes available in the dataset ( here 46).
4. Normalization of Pixel values in X by dividing each pixel value by the highest pixel value (255).
5. Split of dataset into training and testing data using ‘train_test_split’ function from sklearn library. Train: Test ratios used: - 1) 80:20 2) 75:25 3) 70:30. (For each split ratio separate experiments are conducted).
6. Label Encoding of character classes in Y. The categorical values are transformed to numerical values for simplified processing.
7. Input Image Size used is 32*32*1.

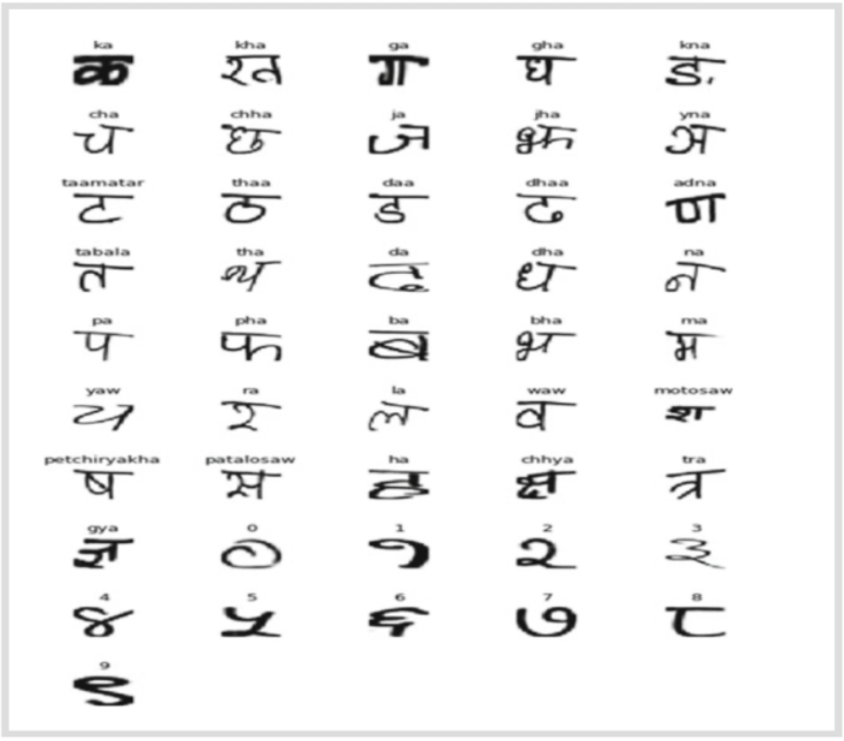


*Fig. 5.1. Devanagari Script characters from DHCD Dataset*

Character class labels are described in the table below.

| **Label Number** | **Character Type** | **Character Class** |
|---|---|---|
| ‘Character_01’ | Characters – consonants and vowels | Ka |
| ‘Character_02’ | | Kha |
| ‘Character_03’ | | Ga |
| ‘Character_04’ | | Gha |
| ‘Character_05’ | | Kna |
| ‘Character_06’ | | Cha |
| ‘Character_07’ | | Chha |
| ‘Character_08’ | | Ja |
| ‘Character_09’ | | Jha |
| ‘Character_10’ | | Yna |
| ‘Character_11’ | | Taamatar |
| ‘Character_12’ | | Thaa |
| ‘Character_13’ | | Daa |
| ‘Character_14’ | | Dhaa |
| ‘Character_15’ | | Adna |
| ‘Character_16’ | | Tabala |
| ‘Character_17’ | | Tha |
| ‘Character_18’ | | Da |
| ‘Character_19’ | | Dha |
| ‘Character_20’ | | Na |
| ‘Character_21’ | | Pa |
| ‘Character_22’ | | Pha |
| ‘Character_23’ | | Ba |
| ‘Character_24’ | | Bha |
| ‘Character_25’ | | Ma |
| ‘Character_26’ | | Yaw |
| ‘Character_27’ | | Ra |
| ‘Character_28’ | | La |
| ‘Character_29’ | | Waw |
| ‘Character_30’ | | Motosaw |
| ‘Character_31’ | | Petchiryakha |
| ‘Character_32’ | | Patalosaw |
| ‘Character_33’ | | Ha |
| ‘Character_34’ | | Chhya |
| ‘Character_35’ | | Tra |
| ‘Character_36’ | | Gya |
| ‘Character_37’ | Digits – Zero to Nine | Digit_0 |

| | | |
|---|---|---|
| ‘Character_38’ | | Digit_1 |
| ‘Character_39’ | | Digit_2 |
| ‘Character_40’ | | Digit_3 |
| ‘Character_41’ | | Digit_4 |
| ‘Character_42’ | | Digit_5 |
| ‘Character_43’ | | Digit_6 |
| ‘Character_44’ | | Digit_7 |
| ‘Character_45’ | | Digit_8 |
| ‘Character_46’ | | Digit_9 |

Table 5.1: Devanagari character class labels

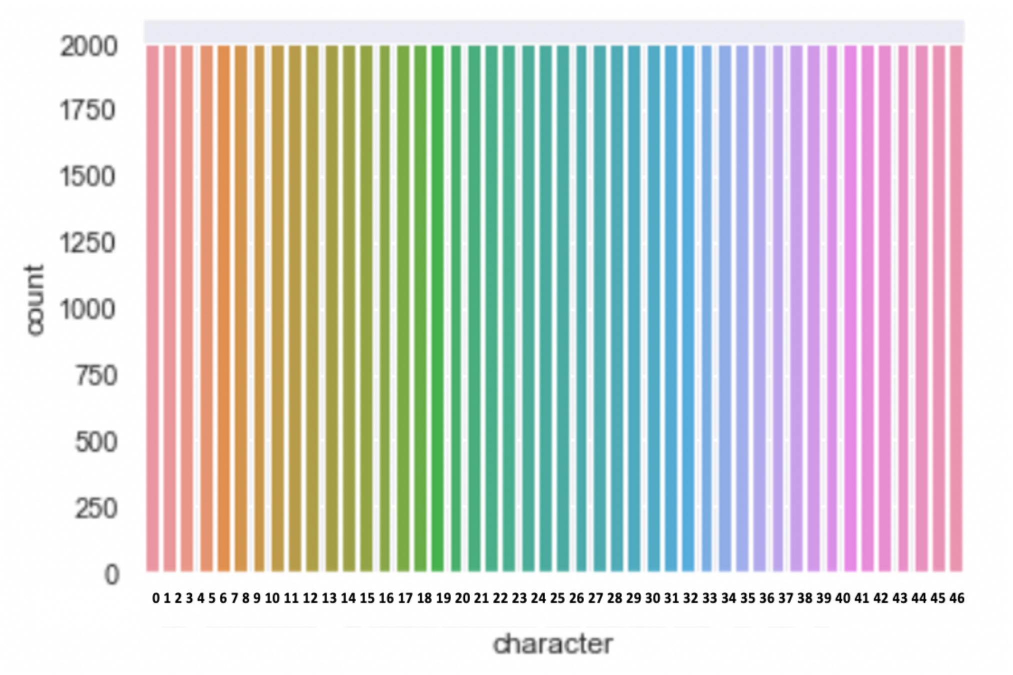


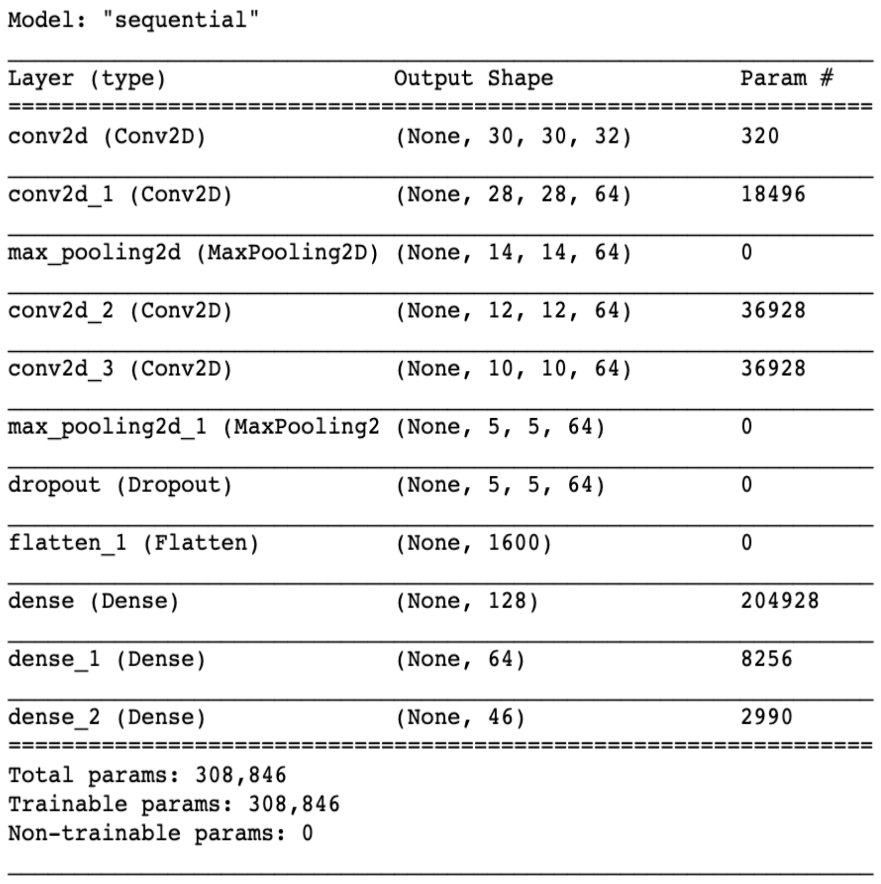
*Fig. 5.2.* Count Plot of character labels

Transformation of categorical variables (characters) into numerical variables is done using label encoder.

```
{'adna': 0, 'ba': 1, 'bha': 2, 'cha': 3, 'chha': 4, 'chhya': 5,
'da': 6, 'daa': 7, 'dha': 8, 'dhaa': 9, 'ga': 10, 'gha': 11,
'gya': 12, 'ha': 13, 'ja': 14, 'jha': 15, 'ka': 16, 'kha': 17,
'kna': 18, 'la': 19, 'ma': 20, 'motosaw': 21, 'na': 22, 'pa':
23, 'patalosaw': 24, 'petchiryakha': 25, 'pha': 26, 'ra': 27,
'taamatar': 28, 'tabala': 29, 'tha': 30, 'thaa': 31, 'tra': 32,
'waw': 33, 'yaw': 34, 'yna': 35}
```

## 5.3. <u>Creation of CNN Model:</u>

CNN is built using Sequential Modelling in Keras i.e., each layer is added one by one using the add() function. This allows one to build a network in a simple manner.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 30, 30, 32)        320
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 28, 28, 64)        18496
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 12, 12, 64)        36928
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 10, 10, 64)        36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
dropout (Dropout)            (None, 5, 5, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1600)              0
_________________________________________________________________
dense (Dense)                (None, 128)               204928
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_2 (Dense)              (None, 46)                2990
=================================================================
Total params: 308,846
Trainable params: 308,846
Non-trainable params: 0
_________________________________________________________________
```

*Fig. 5.3.* CNN Model Architecture

Explanation of the layers used for building the CNN model: -

conv2d: Input Layer : A 2D convolutional layer is added having 32 filters and kernel size of 3*3. Activation function used is 'ReLU'.

conv2d_1: 2D convolutional layer having 64 filters and kernel size of 3*3. Activation function used is 'ReLU'

max_pooling2d: Maximum Pooling layer is added to subsample the size of feature maps from previous layers. Pool size is (2,2) and stride of 2 is used.

conv2d_2: 2D convolutional layer having 64 filters and kernel size of 3*3. Activation function used is 'ReLU'

conv2d_3: 2D convolutional layer having 64 filters and kernel size of 3*3. Activation function used is 'ReLU'

max_pooling2d_1: Maximum Pooling layer is added to subsample the size of feature maps from previous layers. Pool size is (2,2) and stride of 2 is used.

dropout: A dropout layer is added to avoid the chances of overfitting. 20% of the units have been randomly dropped from connections.

flatten_1: Flatten layer is added to convert the feature maps from previous layers of size (5*5*64) into a single dimensional vector of size(1600).

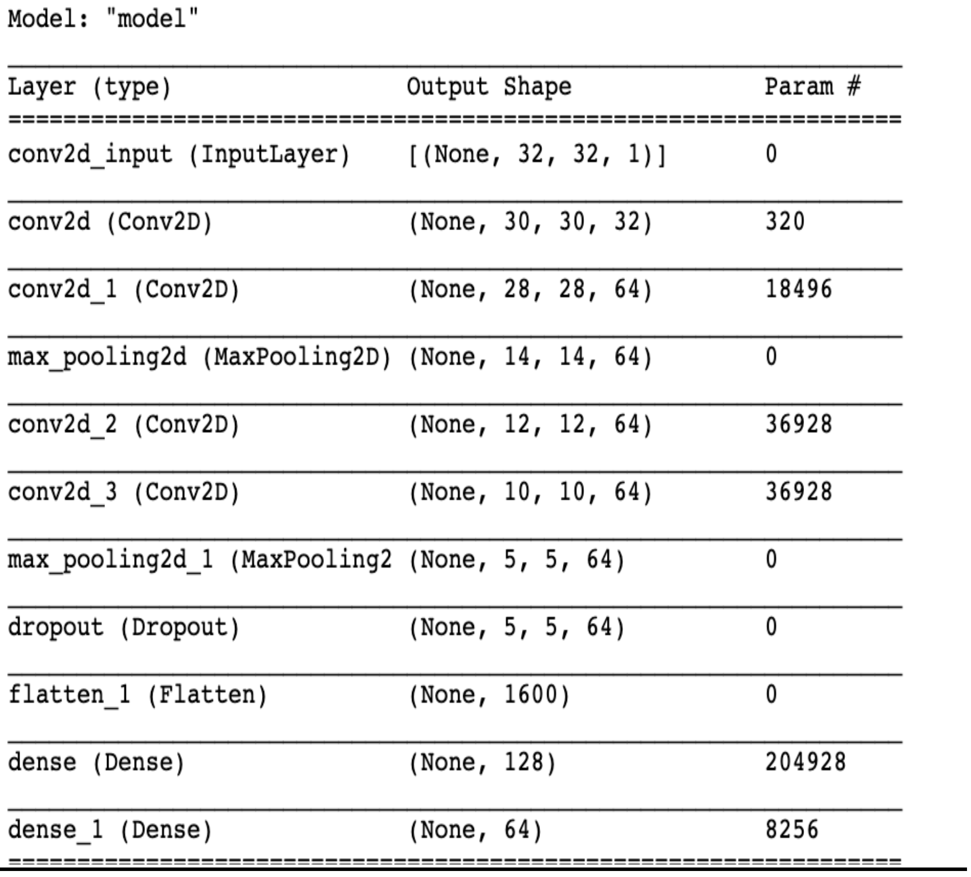
dense: The flattened layer is fed to the fully connected layer having 128 filters.

dense_1: Another fully connected layer is added having 64 filters.

dense_2: Last fully connected layer has number of filters equal to the number of character classes to be recognised (=46 here).

### 5.3.1. Feature Extraction using CNN:

A custom model is created using input model from CNN model. The outputs of this model are 2nd last dense layer (dense_1) of CNN model built above. Therefore, the last fully connected layer having 46 units is removed and the outputs from the custom model are fed to classifiers instead. For feeding the feature maps to a classifier, a generic function is created.



```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_input (InputLayer)    [(None, 32, 32, 1)]       0
_________________________________________________________________
conv2d (Conv2D)              (None, 30, 30, 32)        320
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 28, 28, 64)        18496
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 12, 12, 64)        36928
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 10, 10, 64)        36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
dropout (Dropout)            (None, 5, 5, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1600)              0
_________________________________________________________________
dense (Dense)                (None, 128)               204928
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
=================================================================
```

*Fig. 5.4.* CNN Model for feature extraction

## 5.4. Creation of a generic function for passing various classifiers

A generic function is created that takes following inputs :-

clfr – classifier

x_train_data – pixel values of characters in the training dataset

y_train_data – character_labels in the training dataset

x_test_data – pixel values of characters in the testing dataset

y_test_data - character labels in the testing dataset

acc_str - String to display accuracy of respective classifier

Function methods:

i. Function to train the model using clfr.fit method
ii. Function to predict the character labels from the test dataset fed
iii. Function to calculate the time taken by the classifier for prediction

Values returned from the generic function :

y_pred – predicted character labels

acc - accuracy score i.e., closeness of predicted class labels to the actual class label

Generic function is called by various classifiers- SVM, KNN, MLP, DT, RF

## 5.5. Comparitive Analysis of Models:

The classification accuracy and prediction time for each model is noted. The classifiers are compared on the basis of these two metrics and the results have been compiled in the form of a table and bar plots for easier comprehension. The plots have been analysed in the results section.

# CHAPTER 6

# RESULTS AND DISCUSSION

The models have been analysed on the basis of classification accuracy (performance) and training & prediction time (speed). Recognition results on the DHCD dataset have been presented in the form of tables and graphs for easier interpretation.

Table 6.1. shows the classification accuracy obtained with various models. The top 3 models in terms of accuracy rate are Model 4 (CNN + SVM_RBF), Model1 (CNN + KNN) and Model 7 (CNN+RF), in the decreasing order. Model 6 (CNN + DT) gives accuracy rate of approx. 94% on the given dataset.

| **Model Used** | **Classification Accuracy** | | | **Average Classification Accuracy** |
|---|---|---|---|---|
| | **Case 1: Dataset Split - 70:30** | **Case 2: Dataset Split - 75:25** | **Case 3: Dataset Split - 80:20** | |
| Model 1:<br>CNN + KNN | 98.73% | 98.91% | 98.99% | **98.876 %** |
| Model 2:<br>CNN + SVM_Linear | 98.41% | 98.40% | 98.87% | 98.56 % |
| Model 3:<br>CNN + SVM_Poly | 98.36% | 98.63% | 98.74% | 98.576 % |
| Model 4:<br>CNN + SVM_RBF | 98.88% | 99.01% | 99.13% | **99.006 %** |
| Model 5:<br>CNN + MLP | 98.55% | 98.69% | 98.86% | 98.7 % |

| | | | | |
|---|---|---|---|---|
| Model 6:<br>CNN + DT | 93.54% | 93.85% | 94.41% | 93.93 % |
| Model 7:<br>CNN + RF | 98.57% | 98.84% | 98.7% | **98.703 %** |

Table 6.1: Classification Accuracy Results of the proposed method on the DHCD dataset

Table 6.2. shows the training time taken by CNN model for the extraction of features from handwritten Devanagari characters. Average time taken is around 8 min for 3 cases of dataset split.

| **Feature Extraction time taken by CNN (in seconds)** | | |
|---|---|---|
| **Case 1: Dataset Split - 70:30** | **Case 2: Dataset Split - 75:25** | **Case 3: Dataset Split - 80:20** |
| 497.36 sec<br>= 8.289 min | 467.68 sec<br>= 7.794 min | 532.48 sec<br>= 8.874 min |

Table 6.2: Training time results of the CNN Model for feature extraction

Table 6.3. shows the classification time or prediction time required by various models to classify the Devanagari handwritten characters, by taking input as feature maps obtained from custom CNN model. The top 3 models in terms of prediction time are Model 6 (CNN + DT), Model 2 (CNN + SVM_Linear and Model 3 (CNN + SVM_Poly) in the decreasing order, respectively.

| **Model Used** | **Prediction Time (in seconds)** | | | **Average Prediction Time** |
|---|---|---|---|---|
| | **Case 1: Dataset Split - 70:30** | **Case 2: Dataset Split - 75:25** | **Case 3: Dataset Split - 80:20** | |
| Model 1:<br>CNN + KNN | 21.35 sec | 30.84 sec | 25.60 sec | 25.93 sec |

| | | | | |
|---|---|---|---|---|
| Model 2:<br>CNN + SVM_Linear | 10.78 sec | 10.80 sec | 12.03 sec | **11.203 sec** |
| Model 3:<br>CNN + SVM_Poly | 20.82 sec | 20.79 sec | 22.20 sec | **21.246 sec** |
| Model 4:<br>CNN + SVM_RBF | 23.78 sec | 23.21 sec | 26.60 sec | 24.53 sec |
| Model 5:<br>CNN + MLP | 22.84 sec | 17.73 sec | 30.03 sec | 23.53 sec |
| Model 6:<br>CNN + DT | 2.95 sec | 2.79 sec | 3.43 sec | **3.05 sec** |
| Model 7:<br>CNN + RF | 20.21 sec | 20.20 sec | 23.88 sec | 21.43 sec |

Table 6.3: Prediction time Results of the proposed method on the DHCD dataset

Comparison of all models accuracy rates has been presented in figure 6.1.

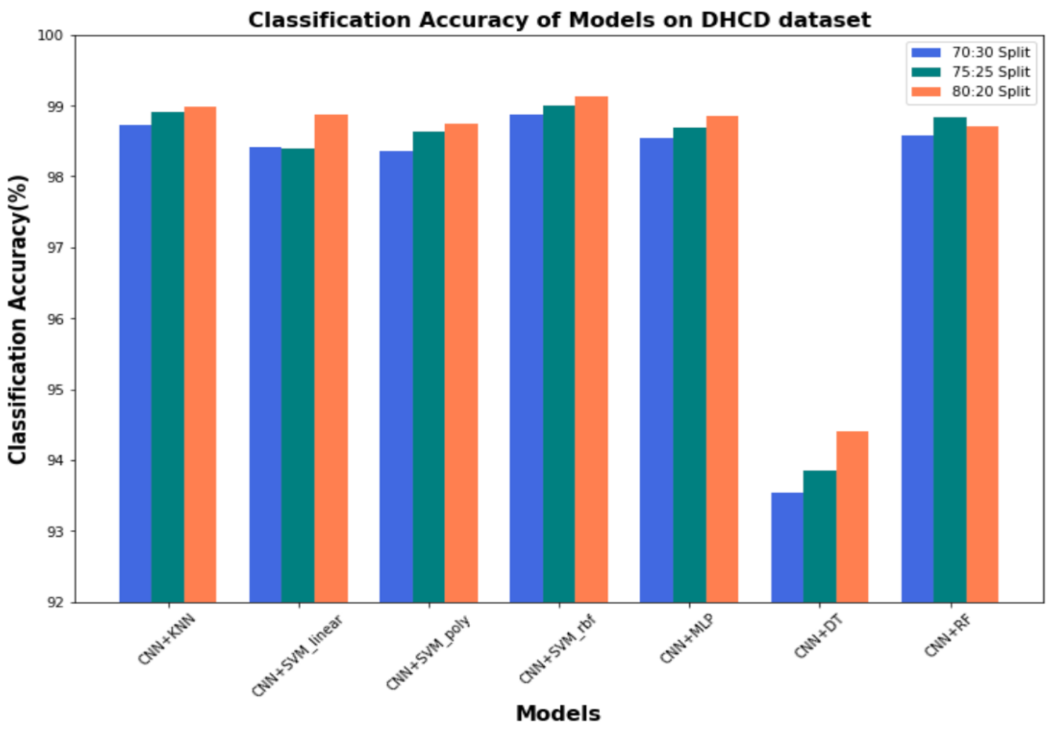


*Fig. 6.1. Comparison of Classification Accuracy of all models* used

Comparison of all prediction time required by all models has been presented in figure 6.2.

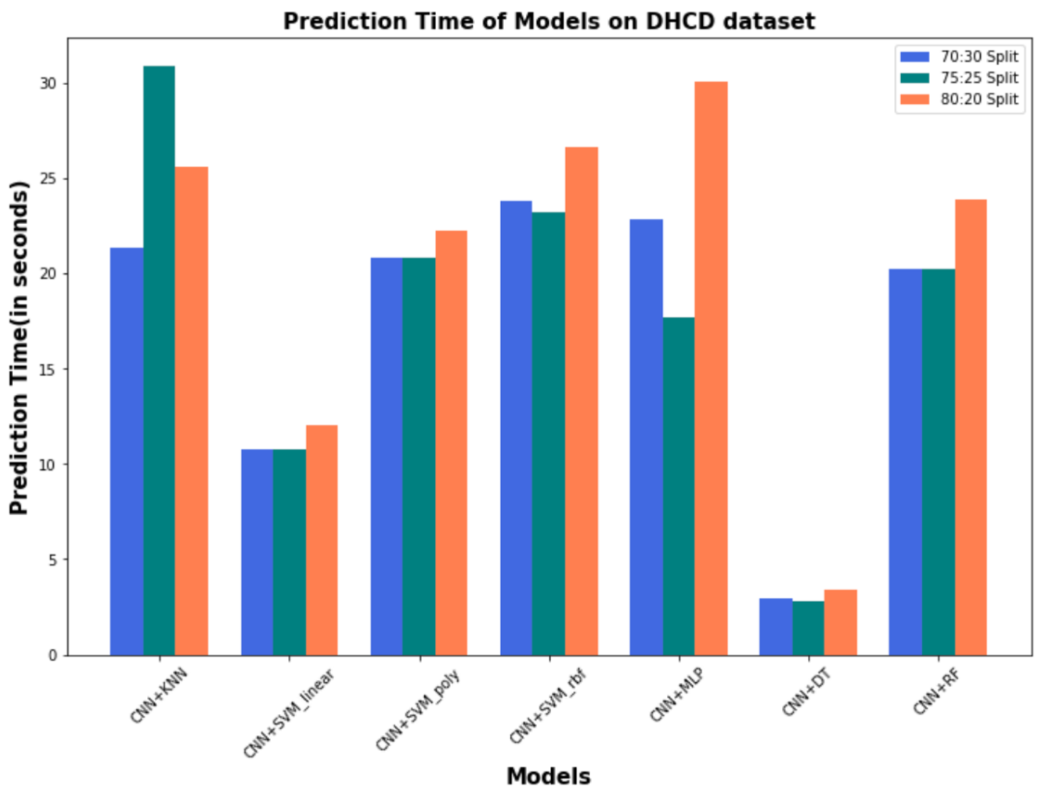


*Fig. 6.2. Comparison of Prediction time of all models* used

# CHAPTER 7

# CONCLUSION

It has been observed that CNN is able to identify relevant features from images and thus results in computational cost saving as compared to traditional pattern recognition techniques for feature extraction. It also saves the hassle and struggle of analysing and identifying best features for the image dataset and then applying various techniques or combination of feature extraction techniques to get appropriate feature maps. Use of CNN as feature extractor reduced the training and prediction time significantly for all classifiers (in the range of few seconds), along with achieving accuracy rates as high as 99% (CNN + SVM_RBF). In terms of computational speed, combination of CNN and DT gives reasonably good accuracy (approx. 94%) within least time, requiring only 3.05 seconds on an average. Optimal choice of method for the recognition of Devanagari handwritten characters, however, is feature extraction with CNN model and classification with SVM classifier.

# CHAPTER 8
# FUTURE DIRECTIONS

From the study and analysis of presently available Devanagari Handwritten character Recognition systems, following future directions are pointed out: -

1. Development of a benchmark handwritten characters dataset containing all primary characters, compound characters and numerals in handwritten form. The dataset must incorporate handwriting of different individuals from a large community, so that the dataset can have more variance in terms of translation, rotation, noise and it must consist of a very large sample size.
2. Propose a novel feature extraction method for selection of appropriate features of Devanagari characters that can allow classifier to distinguish similar characters and improve accuracy for the cursive Devanagari characters.
3. Exploration of Xception Net and XGB Classifier for recognition of Devanagari characters. Use of parameter tuning using XGBoost Algorithm for improvement of classification of handwritten Devanagari characters.

# REFERENCES

[1] Mithe, R., Indalkar, S., & Divekar, N., "Optical character recognition", International journal of recent technology and engineering (IJRTE), 2(1), 72-75,2013.

[2] Yadav, P., & Yadav, N. (2015). Handwriting recognition system-a review. International Journal of Computer Applications, 114(19), 36-40.

[3] Le Cun,Y., Bengio , Y.& Hinton, G." Deep learning", Nature 521, 436– 444 , 2015.

[4] Umapada Pal, Ramachandran Jayadevan, and Nabin Sharma, "Handwriting recognition in Indian regional scripts: A Survey of Offline Techniques.", ACM Transactions on Asian Language Information Processing 11, 1, Article 1, 35 pages, March 2012.

[5] Plamondon, R., & Srihari, S. N. (2000). Online and off-line handwriting recognition: a comprehensive survey. IEEE Transactions on pattern analysis and machine intelligence, 22(1), 63-84.

[6] Caesar, T., Gloger, J. M., & Mandler, E. (1993, October). Preprocessing and feature extraction for a handwriting recognition system. In Proceedings of 2nd International Conference on Document Analysis and Recognition (ICDAR'93) (pp. 408-411). IEEE.

[7] Sanmorino and S. Yazid, "A survey for handwritten signature verification," 2012 2nd International Conference on Uncertainty Reasoning and Knowledge Engineering, 2012, pp. 54-57, doi: 10.1109/URKE.2012.6319582.

[8] Fischer, A., Indermühle, E., Bunke, H., Viehhauser, G., & Stolz, M. (2010, June). Ground truth creation for handwriting recognition in historical documents. In Proceedings of the 9th IAPR International Workshop on Document Analysis Systems(pp. 3-10).

[9] Beigi, H. S., Nathan, K., Clary, G. J., & Subrahmonia, J. (1994, July). Challenges of handwriting recognition in Farsi, Arabic and other languages with similar writing styles an on-line digit recognizer. In Second Annual Conference on Technological Advancements in Developing Countries.

[10] Rusu, A., & Govindaraju, V. (2006, October). The influence of image complexity on handwriting recognition. In Tenth International Workshop on Frontiers in Handwriting Recognition.

[11] Bright, W., “The devanagari script.”, The world’s writing systems, pp.384-390,1996.

[12] R.Jayadevan, S. R. Kolhe, P. M. Patil and U. Pal, "Offline Recognition of Devanagari Script: A Survey," in IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 41, no. 6, pp. 782-796, Nov. 2011.

[13] Deore, S.P., Pravin, A. Devanagari Handwritten Character Recognition using fine-tuned Deep Convolutional Neural Network on trivial dataset. Sādhanā 45, 243 (2020). https://doi.org/10.1007/s12046-020-01484-1

[14] S.Acharya, A. K. Pant and P. K. Gyawali, "Deep learning based large scale handwritten Devanagari character recognition," 2015 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA), pp. 1-6,2015.

[15] R.Kumar, A. Kumar, and P. Ahmed, A Benchmark Dataset for Devanagari Document Recognition Research, WSEAS Press, Lemesos, Cyprus, 2013

[16] N.Majid and E. H. Barney Smith, "Performance Comparison of Scanner and Camera-Acquired Data for Bangla Offline Handwriting Recognition," 2019 International Conference on Document Analysis and Recognition Workshops (ICDARW), 2019, pp. 31-36, doi: 10.1109/ICDARW.2019.30061.

[17] Algiahi, Y. (2010). Preprocessing techniques in character recognition. Character recognition, 1, 1-19.

[18] Khan, A. M., & Ravi, S. (2013). Image segmentation methods: A comparative study.

[19] Choudhary, A. (2014). A review of various character segmentation techniques for cursive handwritten words recognition. Int J Inf Comput Technol, 4(6), 559-564.

[20] Soora, N. R., & Deshpande, P. S. (2018). Review of feature extraction techniques for character recognition. IETE Journal of Research, 64(2), 280-295.

[21] Chacko, A. M. M., & Dhanya, P. M. (2015). A comparative study of different feature extraction techniques for offline Malayalam character recognition. In Computational Intelligence in Data Mining-Volume 2 (pp. 9-18). Springer, New Delhi.

[22] W. Liu, J. Wei and Q. Meng, "Comparisions on KNN, SVM, BP and the CNN for Handwritten Digit Recognition," 2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications( AEECA), 2020, pp. 587-590, doi: 10.1109/AEECA49918.2020.9213482.

[23] P. C. Vashist, A. Pandey and A. Tripathi, "A Comparative Study of Handwriting Recognition Techniques," 2020 International Conference on Computation, Automation and Knowledge Management (ICCAKM), 2020, pp. 456-461, doi: 10.1109/ICCAKM46823.2020.9051464.

[24] S. Kubatur, M. Sid-Ahmed and M. Ahmadi, "A neural network approach to online Devanagari handwritten character recognition," 2012 International Conference on High Performance Computing & Simulation (HPCS), 2012, pp. 209-214, doi: 10.1109/HPCSim.2012.6266913.

[25] Yadav, D., Sánchez-Cuadrado, S., & Morato, J. (2013). Optical character recognition for Hindi language using a neural-network approach. Journal of Information Processing Systems, 9(1), 117-140.

[26] Pal, U., Chanda, S., Wakabayashi, T., & Kimura, F. (2008, August). Accuracy improvement of Devnagari character recognition combining SVM and MQDF. In Proc. 11th Int. Conf. Frontiers Handwrit. Recognit (pp. 367-372).

[27] Jangid,M.,&Srivastava,S.(2014).Gradient localauto-correlation for handwritten Devanagari character recognition. In IEEE International Conference on High Performance Computing and Applications (pp. 1–5). Bhubaneswar: IEEE Press.

[28] Dixit, A. Navghane and Y. Dandawate, "Handwritten Devanagari character recognition using wavelet based feature extraction and classification scheme,"

2014 Annual IEEE India Conference (INDICON), 2014, pp. 1-4, doi: 10.1109/INDICON.2014.7030525.

[29] J. Dongre and V. H. Mankar, "Devanagari offline handwritten numeral and character recognition using multiple features and neural network classifier," 2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom), 2015, pp. 425-431.

[30] S. Shelke and S. Apte, "Performance optimization and comparative analysis of neural networks for handwritten Devanagari character recognition," 2016 International Conference on Signal and Information Processing (IConSIP), 2016, pp. 1-5, doi: 10.1109/ICONSIP.2016.7857482.

[31] Parui, S. K., & Shaw, B. (2007). Offline handwritten Devanagri word recognition: An HMM based approach. In A. Ghose, R. K. De, & S. K. Pal (Eds.), PReMI (Vol. 4815, pp. 528–535). Berlin: Springer-verlag, LNCS.

[32] Singh, B., Mittal, A., Ansari, M. A., & Ghosh, D. (2011). Handwritten word recognition: A curvelet transform based approach. International Journal of Computer Science and Engineering Survey (IJCSES), 3(4), 1658–1665.

[33] Shaw,B.,Bhattacharya, U.,& Parui, S.K.(2014).Combination of features for efficient recognition of offline handwritten Devanagri words. In IEEE 14th International Conference on Frontier in Handwritten Recognition (pp. 240–245). Heraklion: IEEE Press.

[34] Shaw,B.,Bhattacharya,U.,&Parui, S.K.(2015).Offline handwritten Devanagari word recognition: Information fusion at feature and classifier level. In IEEE 3rd IAPR Asian Conference on Pattern Recognition (pp. 720–724). Malaysia: IEEE Press.

[35] S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network," 2017 International Conference on Engineering and Technology (ICET), pp. 1-6, 2017.

[36] P. K. Sonawane and S. Shelke, "Handwritten Devanagari Character Classification using Deep Learning.," 2018 International Conferenceon Information, Communication, Engineering and Technology (ICICET), pp. 1-4, 2018.

[37] Krizhevsky, "One weird trick for parallelizing convolutional neural networks", CoRR, [online], 2014.

[38] S. Acharya, A. K. Pant and P. K. Gyawali, "Deep learning based large scale handwritten Devanagari character recognition," 2015 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA), pp. 1-6,2015.

[39] Deore, S.P., Pravin, A. "Devanagari Handwritten Character Recognition using fine-tuned Deep Convolutional Neural Network on trivial dataset.", Sādhanā 45, 243 ,2020.

[40] Simonyan, Karen, and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition", arXiv preprint arXiv:1409.1556 (2014).

[41] Jangid, Mahesh; Srivastava, Sumit. "Handwritten Devanagari Character Recognition Using Layer-Wise Training of Deep Convolutional Neural Networks and Adaptive Gradient Methods" J. Imaging 4, no. 2: 41, 2018.

[42] Guha, R., Das, N., Kundu, M., Nasipuri, M., Santosh, K. C., & IEEE senior member. "DevNet: an efficient CNN architecture for handwritten Devanagari character recognition. International Journal of Pattern Recognition and Artificial Intelligence.",2019.

[43] Mhapsekar M., Mhapsekar P., Mhatre A., Sawant V. "Implementation of Residual Network (ResNet) for Devanagari Handwritten Character Recognition.", Advanced Computing Technologies and Applications. Algorithms for Intelligent Systems,2020.

[44] Narang, S.R., Kumar, M. & Jindal, M.K." DeepNetDevanagari: a deep learning model for Devanagari ancient character recognition." Multimed Tools Appl 80, 20671–20686, 2021.

[45] S. M. Pande and B. K. Jha, "Character Recognition System for Devanagari Script Using Machine Learning Approach," 2021 5th International Conference on Computing Methodologies and Communication (ICCMC), pp. 899-903, 2021.

[46] Mohite and S. Shelke, "Handwritten Devanagari Character Recognition using Convolutional Neural Network," 4th International Conference for Convergence in Technology (I2CT), pp.1-4, 2018.

[47] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. "Rethinking the inception architecture for computer vision.”, In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2818-2826),2016.

[48] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. “Densely connected convolutional networks.” In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4700- 4708),2017.

[49] Shin, H.C., Roth, H.R., Gao, M., Lu, L., Xu, Z., Nogues, I., Yao, J., Mollura, D. and Summers, R.M., 2016. Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. IEEE transactions on medical imaging, 35(5), pp.1285-1298.

[50] Ramya, S., & Shama, K. Comparison of SVM Kernel effect on online handwriting recognition: A case study with Kannada script. In Data engineering and intelligent computing (pp. 75-82), 2018. Springer, Singapore.

[51] Zhang, M. L., & Zhou, Z. H. ML-KNN: A lazy learning approach to multi-label learning. Pattern recognition, 40(7), 2038-2048, 2007.

[52] Song, Y. Y., & Ying, L. U. Decision tree methods: applications for classification and prediction. Shanghai archives of psychiatry, 27(2), 130, 2015.

[53] Bernard, S., Adam, S., & Heutte, L. Using random forests for handwritten digit recognition. In Ninth international conference on document analysis and recognition (ICDAR 2007) (Vol. 2, pp. 1043-1047), 2007. IEEE.

[54] Satange, D. D., Ajmire, D. P., & Khandwani, F. I. Offline Handwritten Gujrati Numeral Recognition Using MLP Classifier. International Journal of Novel Research and Development, 3(8), 2018.

[55] Rahman MH, Shahjalal M, Hasan MK, Ali MO, Jang YM. Design of an SVM Classifier Assisted Intelligent Receiver for Reliable Optical Camera Communication. Sensors. 2021; 21(13):4283.

[56] Atallah, Dalia & Badawy, Mohammed & El-Sayed, Ayman & Ghoneim, Mohamed. (2019). Predicting kidney transplantation outcome based on hybrid feature selection and KNN classifier. Multimedia Tools and Applications. 78. 20383–20407. 10.1007/s11042-019-7370-5.

[57] Zhou, Bei & Li, Zongzhi & Zhang, Shengrui & Zhang, Xinfen & Liu, Xin & Ma, Qiannan. (2019). Analysis of Factors Affecting Hit-and-Run and Non-Hit-and-Run in Vehicle-Bicycle Crashes: A Non-Parametric Approach Incorporating Data Imbalance Treatment. Sustainability. 11. 1327. 10.3390/su11051327.

[58] Akpojaro, Jackson & Bello, Rotimi-Williams. (2020). Image Processing and Supervised Learning for Efficient Detection of Animal Diseases.. 11. 39-48.

[59] Alboaneen, Dabiah & Tianfield, Hua & Zhang, Yan. (2017). Glowworm Swarm Optimisation for Training Multi-Layer Perceptrons. 131-138. 10.1145/3148055.3148075.

[60] McKinney, W. (2011). pandas: a foundational Python library for data analysis and statistics. Python for high performance and scientific computing, 14(9), 1-9.

[61] Harris, Charles R., K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser et al. "Array programming with NumPy." Nature 585, no. 7825 (2020): 357-362.

[62] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. the Journal of machine Learning research, 12, 2825-2830.

[63] Brownlee, J. (2016). Deep learning with Python: develop deep learning models on Theano and TensorFlow using Keras. Machine Learning Mastery.

[64] Chollet, F. (2018). Keras: The python deep learning library. Astrophysics Source Code Library, ascl-1806.

[65] Ari, N., & Ustazhanov, M. (2014, September). Matplotlib in python. In 2014 11th International Conference on Electronics, Computer and Computation (ICECCO) (pp. 1-6). IEEE.