OPTICAL CHARACTER RECOGNITION (DEVANAGARI SCRIPT)

A REPORT SUBMITTED TO

SAVITRIBAI PHULE PUNE UNIVERSITY

TOWARDS PARTIAL FULFILLMENT OF DEGREE

OF

MASTER OF SCIENCE (M. Sc.)

IN STATISTICS

IN THE FACULTY OF SCIENCE AND TECHNOLOGY

SUBMITTED BY

CHANDRAHAS SURESH NHAYADE (2107)

VIPUL ANIL DHAMALE (2112)

JAGDISH ANANDRAO PATIL (2141)

UNDER THE GUIDENCE OF

DR. MADHURI KULKARNI

DEPARTMENT OF STATISTICS

AND

CENTRE FOR ADVANCED STUDIES IN STATISTICS,

SAVITRIBAI PHULE PUNE UNIVERSITY,

PUNE-411007,

INDIA.

MAY, 2023

Certificate

This is to certify that, the following students of M.Sc. Statistics,

(1) Chandrahas Suresh Nhayade (2107)

(2) Vipul Anil Dhamale (2112)

(3) Jagdish Anandrao Patil (2141)

have successfully completed their project titled **Optical Character Recognition (Devanagari Script)** under the guidance of **Dr. Madhuri Kulkarni** and have submitted this project report on **May 22, 2023** as a part of the course ST-402, towards partial fulfillment of requirements for degree of M.Sc. Statistics in Savitribai Phule Pune University during academic year 2022-2023.

Dr. Madhuri Kulkarni
(Project Guide)

Prof. T. V. Ramanathan
(Head of the Department)

Acknowledgments

We would like to express special thanks of gratitude to our project guide, **Dr. Madhuri Kulkarni** Ma'am, for her valuable inputs and continued motivation. Ma'am lent us valuable books, referred us to some great papers, gave new perspectives and ideas and was always accessible for discussion and doubts. We are profoundly grateful for the support.

This project would have been impossible without **Mr. Subhash Kulkarni** Sir. Their guidance on Deep Learning techniques helped us to clear our basics, enabling us to take up this project. Also, a big thanks to GitHub and Cross Validated online communities.

We are thankful to the Department of Statistics, Savitribai Phule Pune University, for giving us this opportunity. This project truly helped us to apply statistics in real life and growing field of Deep Learning. We have learnt much more than we ever imagine.

Contents

List of Tables

List of Figures

Chapter 1

Introduction and Summary

1.1 Introduction

Given the ubiquity of handwritten documents in human transactions, Optical Character Recognition (OCR) of documents have invaluable practical worth. OCR is a science that enables to translate various types of documents or images into analyzable, editable and searchable data. During decade of '01, researchers have used artificial intelligence/machine learning tools to automatically analyze handwritten and printed documents in order to convert them into electronic format.

OCR is a system that converts input text into machine-encoded format. Today, OCR is helping not only in digitizing the handwritten medieval manuscripts, but also helping in converting the typewritten documents into digital form. This has made the retrieval of the required information easier as one does not have to go through the pile of documents and files to search the required information. Organizations are satisfying the needs of digital preservation of historic data, law documents, educational scripts etc.

An OCR system depends mainly, on the extraction of features and dis-

crimination/classification of these features (based on patterns). Handwritten OCR have received increasing attention as a sub-field of OCR. It is further categorized into offline system and online system based on input data. The offline system is a static system in which input data is in the form of scanned images while in online systems nature of input is more dynamic and is based on the movement of pen tip having certain velocity, projection angle, position and locus point. Therefore, an online system is considered more complex and advance, as it resolves the overlapping problem of input data that is present in the offline system.

One of the earliest OCR systems was developed in the 1940s, with the advancement in the technology over the time, the system became more robust to deal with both printed, and handwritten characters and this led to the commercial availability of the OCR machines. In 1965, advance reading machine "IBM 1287" was introduced at the "world fair" in New York. This was the first-ever optical reader, which was capable of reading handwritten numbers. During the 1970s, researchers focused on the improvement of response time and performance of the OCR system.

The next two decades from 1980 till 2000, the software system of OCR was developed and deployed in educational institutes, census OCR and for recognition of stamped characters on metallic bar. In the early 2000s, binarization techniques were introduced to preserve historical documents in digital form and provide researchers with access to these documents. Some of the challenges of binarization of historical documents were the use of non-standard fonts, printing noise and spacing. In mid of 2000, multiple applications were introduced that were helpful for differently-abled people. These applications helped these people in developing reading and writing skills.

In the last decade, researchers have worked on different machine learning approaches which include Support Vector Machine (SVM), Random Forests (RF), k Nearest Neighbor (kNN), Decision Tree (DT), Neural Networks etc.

Researchers combined these machine learning techniques with image processing techniques to increase the accuracy of the OCR system. Recently researchers have focused on developing techniques for the digitization of handwritten documents, primarily based on deep learning approach. This paradigm shift has been sparked due to adaption of cluster computing and Graphic Processing Units (GPUs) and better performance by deep learning architectures, which includes Recurrent Neural Networks (RNN), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM) networks etc.

1.2 Motivation

OCR finds wide applications as a telecommunication aid for the deaf, postal address reading, direct processing of documents, foreign language recognition etc. This problem has been explored in depth for the Latin script. However, there are not many reliable OCR software available for the Indian language Hindi (Devanagari), the third most spoken language in the world. This provides a good starting point for the problem and presents a good overview.

1.3 Abstract

The optical character recognition (OCR) systems for Hindi language were the most primitive ones and occupy a significant place in pattern recognition. The Hindi language OCR systems have been used successfully in a wide array of commercial applications. The basic challenge involved in the OCR systems for Hindi language is investigated in this project. The pre-processing activity of sparcity removal is performed on the dataset considered. The feature extraction is performed through Histogram Oriented Gradient (HOG) method. The feature based classification is performed through important soft computing

techniques viz Support Vector Machine (SVM), k-Nearest Neighbor (k-NN), Random Forest, Naïve Bayes and Convolutional Neural Network (CNN). The superiority of these soft computing techniques is demonstrated through the experimental results.

1.4 Literature Review

1.4.1 Optical Character Recognition for Sanskrit using Convolution Neural Networks

[1]

Introduction

Sanskrit is gaining importance in various academic communities due to the presence of ancient scientific and mathematical research work written in this language. However, the lack of accurately digitized and tagged versions of Sanskrit manuscripts is a major bottleneck. Hence, it becomes essential to digitize such ancient manuscripts.

Most of the recent Indic OCR systems make use of machine learning algorithms such as support vector machines (SVMs) and artificial neural networks (ANNs) to classify letters in the image. These classifier models used in the OCRs are trained with input images that are often down sampled by applying PCA order to reduce the complexity of the data. However, this results in loss of important information necessary to make the classifier robust.

In order to develop a robust OCR system which can digitize soiled and noisy documents with high accuracy, we propose the use of Convolution neural

networks (convnets) as opposed to the popular use of SVMs and ANNs, as convnets possess very high learning capacity and the capability to handle high dimensional data such as images. Popular Convnet architectures such as the GoogLeNet , ResNet, VGG Net have achieved state of the art results in popular image classification challenges like the ILSVRC(imagenet large scale visual recognition challenge) challenge or the ImageNet Challenge.

Main Contribution of Research Paper

1. Developing an OCR framework for Sanskrit which can digitize soiled and poorly maintained documents.
2. The use of CNN as classifiers for Sanskrit OCR.
3. A Sanskrit letter data consisting of 11,230 images belonging 602 classes.

Related work: Early efforts in Indian OCR involve a template based and feature-based approach for letter classification. In the template based approach, each unknown letter or pattern is compared with a standard template pattern and the degree of similarity between the two patterns is used to decide classification.

OCR frameworks make use of various machine learning algorithms instead of template-based and feature-based methods, to classify letters. They employ an artificial neural network to recognize letters by inputting segmented images of letters into the classifier model. They adopt a similar OCR framework, but they make use of support vector machines instead of ANNs to recognize letters. The advantages of over traditional Indic OCR systems are that they identify unique characteristics in patterns without any explicit derivation of features.

The BLSTM (Bidirectional Long Short-Term Memory Networks) OCR system functions by recognizing text at a word level by recording the past and

future context of a word. As a result, these OCR can directly digitize images without any prior letter detection and image segmentation.

Devanagari script

The Devanagari script consists of 47 basic characters including 14 vowels and 33 consonants. Devanagari script occur as modified shapes in words. These modified characters are called modifiers or allographs. In addition to this, several half letter and full letter consonants often combine to form compound characters. In addition to this, several half letter and full letter consonants often combine to form compound characters. A unique feature in the Devanagari script is the presence of a horizontal line at the top of each word. The horizontal line is called the 'Shirorekha' and is used to connect consecutive characters or letters in a word. We refer to the 'Shirorekha' as the header line.

Data Acquisition, Segmentation and Anotation

In this section, we describe the adopted algorithm to segment the letters in the image and highlight the procedure used to annotate the letter images. Scanned pages of Sanskrit magazines such as different editions of Chandamama are used for creating the letter dataset. The font size and font style of the text in the magazines varied depending on the edition of the magazine. The image quality and contrast are also not uniform due to poor maintenance of the magazines.

- **Image Pre-Processing:** The scanned images are converted from RGB to grayscale. This is followed by binary thresholding of the image, with a threshold value equal to the mean pixel intensity of the grayscale image. The grayscale image is thresholded to suppress any irregularities or noise

that may be present. The threshold parameter was set to a different value for different images as each image had varying amounts of brightness and contrast. Moreover, a constant threshold for all images would result in excessive or insufficient thresholding. Further, it could lead to a loss in important data or presence of unnecessary noise respectively.

- **Letter Segmentation:** The processed thresholded images are used to identify and segment letters. Letter segmentation is carried out by identifying the corresponding line and word in the image first. The lines in the text are identified by calculating a row-wise black colored pixel density. Subsequently, a line is bound to exist between any two local minima as the minimas represent the white spaces between the lines. Following this, the words in the segmented line are identified by calculating the column wise black pixel intensity. As a result, the segment of the image between any two consecutive local minima contains a word.Segmenting the letters by directly identifying the local minima of the column-wise black pixel intensity is not feasible due to the presence of the header line. This is because the thickness of the header line varies consistently, making it difficult to identify the local minima. In order to avoid this ambiguity, the header line in each segmented word is removed. This is achieved by calculating a row-wise black color pixel intensity and subsequently removing the region of the image which contained the maximum row-wise pixel density. Following this, the image without a header line is used to segment letters. The letters are identified by locating the local minima of column-wise black colored pixel intensities, as a letter would exist between any two consecutive minima.

- **Data Labeling:** The proposed segmentation methodology considers the region between any two local minima (zero intensity) to contain a letter. As a result, in some cases, a combination of half letter or full and half letter (compound characters) are segmented as one entity (Fig 3). Each

of these possible combinations has been considered to be a unique class while labeling the letters. Even though this method increases the number of classes, the segmentation results were superior as compared to the approach of identifying and segmenting each half letter in the word. The segmentation results were poor or non-uniform while segmenting half letters due to the presence of ambiguities.

Methodology

- **Data Augmentation:** Data augmentation is a technique where, images are subjected to a random amount of distortion (for example shear, shifting, rotation) such that the distorted image also belongs to the same class the original image belongs to. Data augmentation is an artificial way of magnifying the dataset size so that the classifiers do not overfit. In addition to this, the augmentation makes the convnet scale invariant. Since the dataset used in this work is small, data augmentation is used extensively during training. Each image is subjected to a random amount of twisting, shifting, shear, zoom in and out and jittering, before training.

- **Baseline Architecture:** The image is passed through a series of convolution and max-pooling operations. Each convolution operation contains a 3x3 kernel with stride 1 and no spatial padding. Subsequently, the convolutional output is followed by a ReLU function. The ReLU functions do not saturate while training the convnet and eventually help avoid the vanishing gradient problem. A maxpooling operation is used after every 2 or 3 Convolution operations, depending on the architecture design, to reduce the computational intensity of the architecture. A kernel/filter size of 2x2 with a stride of 2 pixels is used in each max pooling operation. The final convolution layer is followed by 2 fully connected layers. Subsequently, the convnet terminates with a softmax layer. The number of channels in each fully connected layer and the number of filters in each

convolution operation varies for different convnets.

- **Traning:** The Convnets are trained by optimizing on a cross-entropy loss function using mini batch gradient descent and back propagation. The batch size is set to 32, with a nestrov momentum of 0.9. The learning rate is set to a constant value of 0.001, i.e. this value is not altered during training. A constant dropout of 0.2 is used, to prevent over-fitting. The weights are randomly initialized with a standard deviation of 0.1. Each convnet is subjected to a variable number of epochs during training (between 120 to 140 epochs). Training is terminated when the validation accuracy started to drop while the training accuracy continued to improve. In other words, each Convnet is trained till just before they started to overfit on the data.

- **Rationale for Proposed Architecture:** Various modifications made on the baseline architecture were based on the philosophy of Karen, i.e. the depth of the convnet is increased while keeping the size of the filters same. In addition to this, the number of channels in the fully connected layers are also altered.

  Convnet A: Initially convnet A is trained on the data using the procedure mentioned in above training section. The training error of convnet A reached a constant value, showing that a more complex model is required to attain better results.

  Convnet B: Convnet B is designed by adding an additional conv3-32 and conv3-64 layer. The training error for convnet B did not stagnate, rather convnet B started to overfit on the data. This proved that slight changes in the architecture would be sufficient to improve the results.

  Convnet C : As a result, convnet C is designed by doubling the number of neurons in each fully connected layer of convnet B. However, Convnet C showed a slight improvement in results before it started to overfit on the data.

In order to prevent overfitting, a dropout layer is added after the first fully connected layer. The resulting architecture, convnet D, achieves the best results on the dataset. Hence convnet D is the proposed classifier for the OCR. In all the convnets, a series of convolution operations with 3x3 filters are used to generate an effective larger receptive field of 5x5 or 7x7, instead of directly using a 5x5 or 7x7 filters. This is because two consecutive 3x3 filters produce an effective receptive field of a 5x5 filter, but at the same time reduce the number of parameters in the network. Similarly, three consecutive 3x3 filters produce an effective receptive field of a 7x7 filter, while reducing the number of parameters in the network [6]. In addition to this, increasing the number of convolution operations would also increase the non-linearity which leads to an increase in the learning capacity of the networks.

Implementation Details

The entire software is implemented in python. Image segmentation and letter localization are carried out with the help of open source libraries like OpenCV and PIL. The convnets are implemented on Keras using Tensorflow as the backend. Training is carried out on an NVIDIA 960M GPU, training a single convnet took about 1 hour. Since the convnets are simple in nature, techniques like Multi-GPU training are not used to speed up the training process.

Results

The test data was generated by randomly selecting scanned images of pages belonging to different Sanskrit magazines. This ensured that the letters in the dataset belonged to different font style and font size. In addition to this,

the image quality and contrast were not uniform. The test data contained 1124 images of letters.

- **Complete Pipeline of OCR:** A scanned image of a Sanskrit page is entered into the Sanskrit OCR. The Sanskrit OCR segments the letters in the page following the procedure mentioned in Section IV B. Subsequently, the segmented image is classified by the OCR using the proposed convnet. Finally, the English class names of the classified letters are used to generate a text document which is then transliterated into Devanagari script with an Itrans converter.

- **Performance of Test Data:** For comparison with a standard classifier, we train a 5-layer artificial neural network (ANN) on the data. The ANN contains 2500, 2000,1000 and 800 neurons in the first, second, third and fourth layer respectively. The ANN finally terminates with 602 neuron softmax layer. The results achieved by the ANN and convnets on the test dataset is depicted in Table III. The ANN is outperformed by all the convnets due to its limited learning capability and high sensitivity to image quality when compared to convnets.

Conclusion and Future Scope

We present an OCR for Sanskrit (Devanagari script). We introduce a novel approach of using convnets as classifiers for Indic OCRs. We show that convnet are more suitable than SVMs and ANNs, for multi-class image classification problems. In addition to this, we show that our OCR is ideal for digitizing old and poorly maintained material as it robust to font size and style, image quality and contrast. To improve the OCR system further, learning can be introduced for letter segmentation and identification. This could be achieved with the help of a selective search algorithm followed by an R-CNN.

1.4.2 Handwritten Hindi Character Recognition: A Review

[2]

Introduction

The ongoing research to improve reading skills of computers gave rise to the field of document analysis and recognition (DAR). It aims at automatic transformation of information presented on paper into machine readable format. DAR has two main objectives: analysing documents and their recognition. automatic script identification is a sub-domain of DAR which facilitates automatic archiving of multilingual documents. Segmentation of document page into regions such as title, headers, keywords, abstract, and sections helps in online data retrieval based on regions of document.

General Steps of Character Recognition

To be computationally and ethically effective, a character recognition process follows few basic steps listed below:

- **Data Acquisition:** Data acquisition is the process of converting handwritten documents into digitised form. Digitisation is done by electronic devices such as tablets, cameras, or scanners. Apart from features and classifiers, database also plays a major role in recognition rate. If the images in database are distorted, or characters are not properly written or they contain noise which cannot be eliminated by preprocessing, then no matter how good features or classifiers are used, the desired accuracy can never be achieved. Therefore, the existence of some standard

database is essential, as they limit the pitfalls of data acquisition and exhilarates the performance of subsequent feature extraction and classification phases. Table below summarises the currently available databases for Hindi handwritten characters.

Database	Description	Writers
ISI Database	30,000 samples of basic Hindi characters	1049
Vijay Dongre	5137 numerals and 20,305 isolated characters	750
HPL online dataset	data collected from digital equipment	100

Table 1.1: Currently available databases for Hindi handwritten characters

Preprocessing

Preprocessing is the process of applying subsequent operations on input image to transform it into a form which is more suitable for further processing. It improves the quality of elements of the digital images called pixels. Here are the techniques used for image preprocessing.

- Noise Reduction
- Normalisation
- Thinning

Feature Extraction

The purpose of feature extraction is to invent feature vector which unambiguously and efficiently recognise patterns. Features can be classified into two categories: structural and statistical. Authors have proposed character recognition technique based on simple components such as curves, loops, lines, and

dots present in Hindi characters. These components act as feature vector and are classified using Multi Layer Perceptron (MLP) and Radial Basis Function (RBF). Authors in used HOG and projection profiles as features. Fast discrete Curvelet transform is used in for handwritten character recognition. Curvelet coefficients are obtained from thin and thick character images. The fusion of these coefficients obtained by inverse FFT acts as features, similarity between tested feature set and reference feature set is measured by Euclidean Distance (ED) and k-Nearest Neighbour (KNN) classifier. Some other classifiers are used for comparison which are Mirror Image Learning (MIL), Projection Distance (PD), subspace Method, Linear Discriminant Function, SVM, Modified Quadratic Discriminant Function (MQDF), ED, k-NN. MIL classifier is reported to give better results above all.

Classification/recognition

Feature extraction phase transforms input images into feature vectors which represent certain characteristics of that image. These n-dimensional feature vectors should be classified into categories for recognition purpose, and this is the job of classifiers. Classifiers assign class labels, probabilities, or membership scores to them. Classification is itself, a vast research area and also an important phase of character recognition. There are large number of classifiers, but we need to limit our scope, thus classifiers which are most relevant to this article are discussed here. The classifiers can be broadly categorised as statistical, structural, artificial neural network (ANN), SVM, and ensemble (combination) of classifiers.

Convolutional Neural Network

Deep learning is a machine learning technique in which model perform

classification task directly from images. Deep learning architectures can have large number of layers, thus the term 'deep' has been used. The most famous deep learning network is Convolutional Neural Network (CNN). The image pixels are fed as input to the network, there is no manual feature extraction. The basic architecture of CNN . The input layer of CNN is raw image which can be grey scale, RGB or binary. The first layer is always convolutional layer. In this layer, kernels of specific sizes convolve over the entire image and form activation maps, which are input for next layer. The next layer can be again convolutional layer or ReLu layer or pooling layer. In this way, these layers can be repeated and new activation maps can produced at each level. These activation maps act as features which extract information of gradients, edges, corners from the image and are fed as input to fully connected layer or classification layer.

Discussion

- **Lack of standard databases:** The first and most important aspect of any character recognition system is database. It is one of the major drawback of Hindi character recognitionTo improve the quality of research work, there is need of developing standard database which should be publicly available.

- **Recognition of confusing characters:** There are many characters in Hindi language which look alike, thus classification becomes a major challenge. These characters may require special attention for recognition.

- **Improved preprocessing steps:** Most of the works on Hindi characters have rarely focused on preprocessing steps. Selection of preprocessing algorithms has high impact on recognition results.

- **Statistical and structural information:** Statistical features provide good estimates of character pixel characteristics but these features fail

while recognising similar characters. Thus, structural features are used to provide shape-related information. However, neither of them can stand alone for representing a pattern, so a system is required which uses combination of both structural and statistical information. Thus, to develop a robust system, it is necessary to use both style of information to cover diverse characteristics of characters.

- **Word Recognition:** The characters in Hindi language are connected by headline. For word recognition, there is a need to detect this headline and remove it. Detection of headline and lower or upper modifier recognition in handwritten characters is also a research area. The words may contain half characters, touching characters, and broken characters. The literature works do not differentiate between half and complete characters. There is a need to recognise them as well because there will be an ambiguity of whether the character is wrongly segmented or it is a half character.

- **Selection of features:** It is evident that single set of features cannot represent complex patterns thus multiple features are selected for easy recognition. However, using numerous features do not guarantee better recognition results. Thus, selected feature set should provide the information which is not redundant in nature. They should minimise intra-class variations and maximise inter-class variations of characters. The widely used CNN reduces the need of manual feature extraction and hence, its usage for Hindi recognition can also be explored.

- **Classifier fusion:** To improve the classification performance, it may be beneficial to combine the outputs of different classifiers.

Conclusion

The techniques discussed in this paper are relevant works of various researchers with a common aim of improvement in recognition rates. From the survey, it can be concluded that implemented recognition techniques correctly recognises characters of good quality, but fails in recognising overlapped, similar shaped, or ambiguous characters. One effective solution for improving recognition rate is using semantic information of characters at post-processing timeThe need of the hour is to develop a system which is out of theoretical bounds and robust enough to handle practical challenges. Tremendous efforts in this field have brought us half-way and we look forward to achieve the ultimate goal of machine simulation of fluent human reading especially for unconstrained offline handwriting.

1.5 Objectives

In various areas, there is need to convert the OCR text (offline handwritten, online handwritten and machine typed) into digital form so that we won't need to go through pile of documents and can have access to the documents in simpler and less exhausting manner. By using various machine learning tools with important inclusion of deep learning techniques we can identify the handwritten text which can facilitate many professional fields such as banking, post offices, academic institutions, and other professional working bodies which require digital processing of paper documents. Some of our objectives are given below.

1. To train a model (Deep Learning) to recognise pattern in the curves of basic handwritten characters of Devanagari Script which are the foundations of derived (complex) characters seen in usual writing.

2. To train a model (Machine Learning) to separate the then recognised characters in righteous way possible.

3. To compare the efficiency of every fitted model and identify the best fit for respective problem.

4. To obtain the structured output for OCR in Devanagari Script and find sources where the fitted models can be useful in solving the affecting problems.

1.6 Data Description

Hindi language is most spoken language of India. It is the standardized and Sanskritized register of the Hindustani language. The Hindi language is written in the Devnagri script which consists of 11 vowels and 33 consonants as shown in fig 1.1. It is written from left to right and is an abugida as well.

Considered data has been made available from online website given below.
Online data site: https://www.kaggle.com/datasets/rishianand/devanagari-character-set?resource=download

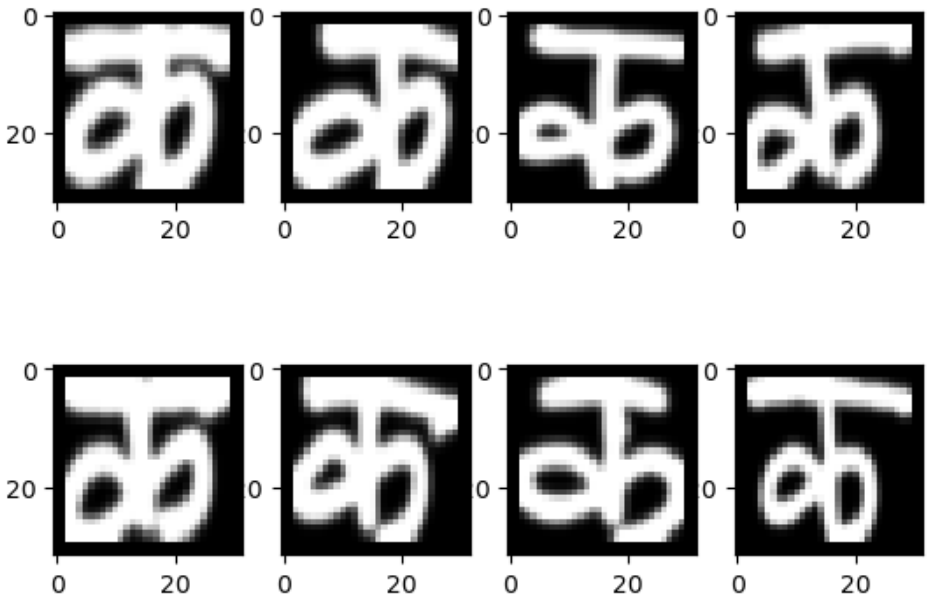


Figure 1.1: Various styles of writing Ka

Context

This is a dataset of Devanagari Script Characters. It comprises of 92000 images [32x32 px] corresponding to 46 characters, consonants "ka" to "gya",

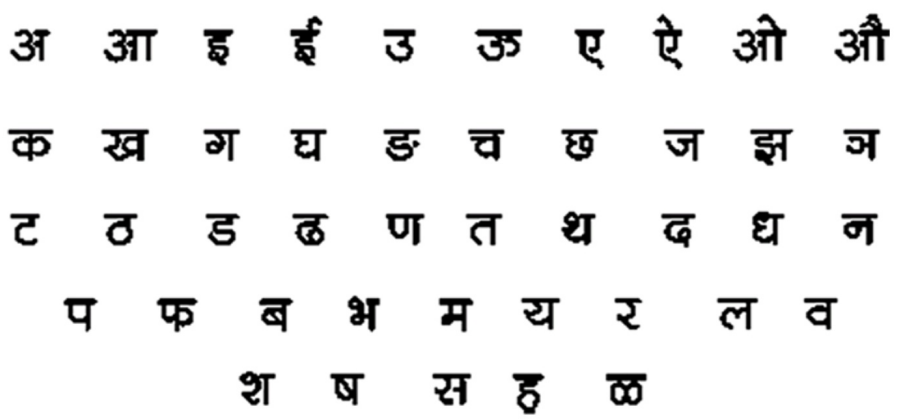


Figure 1.2: Devanagari Script

and the digits 0 to 9. The vowels are missing.

Content

Our training dataset contains 2000 examples of each character, for a total of 92,000 images. Each image consists of 32x32 pixels and 3 color channels. The test set consists of 13800 total images (300 for each character) and the training set consists of 78200 images (1700 per character). This accounts for an 85/15 split.

Also the CSV file is of the dimension 92000 * 1025. There are 1024 input features of pixel values in grayscale (0 to 255). The column "character" represents the Devanagari Character Name corresponding to each image.

1.7 Definitions and Domain Terminologies

Definition 1.7.1. Optical Character Recognition:
Optical character recognition (OCR) is a process that converts printed texts into digital image files. It is a digital copier that uses automation to convert

scanned documents into editable, shareable PDFs that are machine-readable.

Definition 1.7.2. Image:
An image is a visual representation of something. It can be two-dimensional, three-dimensional, or somehow otherwise feed into the visual system to convey information.

Definition 1.7.3. Digitization:
Digitization is the process of converting information into a digital i.e. computer-readable format.The result is the representation of an object, image, sound, document, or signal (usually an analog signal) obtained by generating a series of numbers that describe a discrete set of points or samples.[3] The result is called digital representation or, more specifically, a digital image.

Definition 1.7.4. RGB Colour Model:
The RGB color model is an additive color modelin which the red, green and blue primary colors of light are added together in various ways to reproduce a broad array of colors. The name of the model comes from the initials of the three additive primary colors, red, green, and blue.

Definition 1.7.5. Grayscale Image :
A grayscale image is one in which the value of each pixel is a single sample representing only an amount of light; that is, it carries only intensity information. Grayscale images, a kind of black-and-white or gray monochrome, are composed exclusively of shades of gray. The contrast ranges from black at the weakest intensity to white at the strongest

Definition 1.7.6. Pixel:
A pixel is the smallest addressable element in a image, or the smallest addressable element in a dot matrix display device.

Definition 1.7.7. Pixel Intensity:
Pixel intensity value is the primary information stored within pixels.

Definition 1.7.8. Machine Learning:
Machine learning (ML) is the study of computer algorithms that can improve automatically through experience and by the use of data.

Definition 1.7.9. Deep Learning :
Deep learning is part of a broader family of machine learning methods, which is based on artificial neural networks with representation learning. Learning can be supervised, semi-supervised or unsupervised.

Definition 1.7.10. Neural Networks :
An neural network or artificial neural network or just neural net is a computational learning system that uses a network of functions to understand and translate a data input of one form into a desired output, usually in another form.

Definition 1.7.11. Accuracy:
Accuracy is the degree of closeness between predicted value and the true value.

Definition 1.7.12. Precision:
Precision is defined as the fraction of relevant instances among all retrieved instances.

Definition 1.7.13. Recall:
Recall is the fraction of retrieved instances among all relevant instances.

Definition 1.7.14. F1 Score:
F1-Score or F-measure is an evaluation metric for a classification defined as the harmonic mean of precision and recall. It is a statistical measure of the accuracy of a test or model.

■

Chapter 2

Exploratory Data Analysis

2.1 Data Visualisation

Data visualisation is the graphic depiction of information and data. Data visualisation tools offer a simple approach to examine and comprehend trends, outliers, and patterns in data by utilising visual components like charts, graphs, and maps.

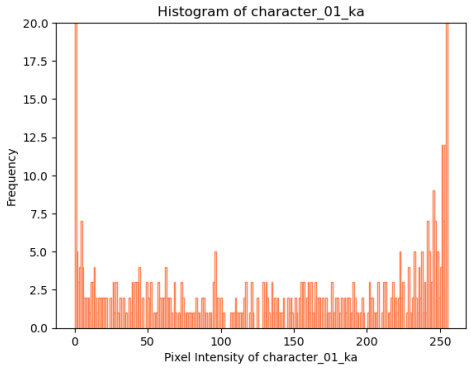
2.1.1 Image Representation

We have retrieved images of devanagri letters and 0 to 9 digits from the pixel intensity data, which are shown below.

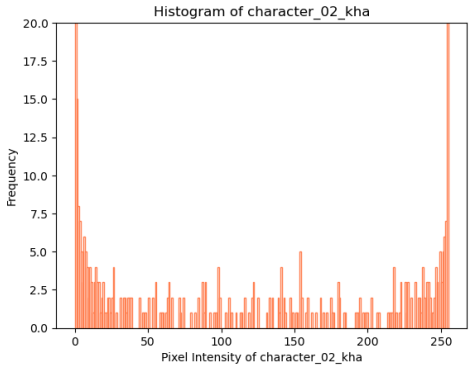
Figure 2.1: Images

2.1.2 Histogram

A histogram is an approximate representation of the distribution of numerical data. A Histogram meaning can be stated as a graphical representation that condenses a data series into an easy interpretation of numerical data by grouping them into logical ranges of different heights which are also known as bins. Basically, it summarizes discrete or continuous data. We can also call it a frequence distribution graph as it is like a plot that lets you discover the underlying frequency distribution.

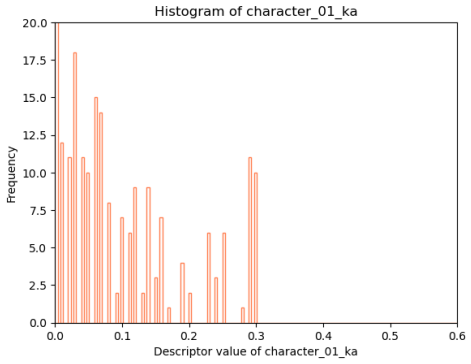


(a) Character Ka Pixel Intensity

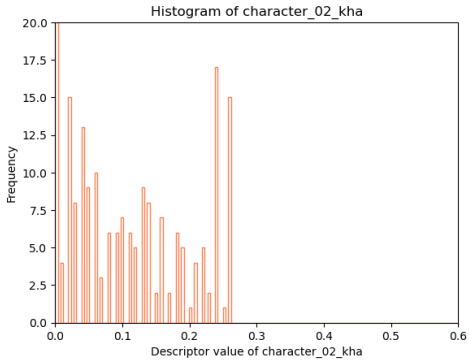


(b) Character Kha Pixel Intensity

Figure 2.2: Histogram of pixel intensity



(a) Character Ka Descriptor Value

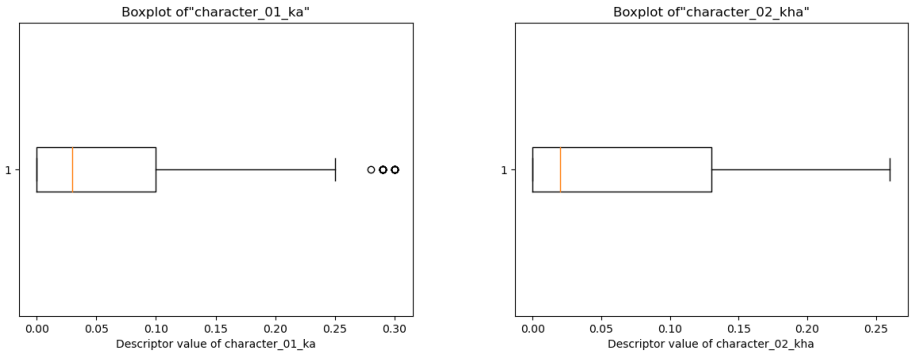


(b) Character Kha Descriptor Value

Figure 2.3: Histogram of descriptor data

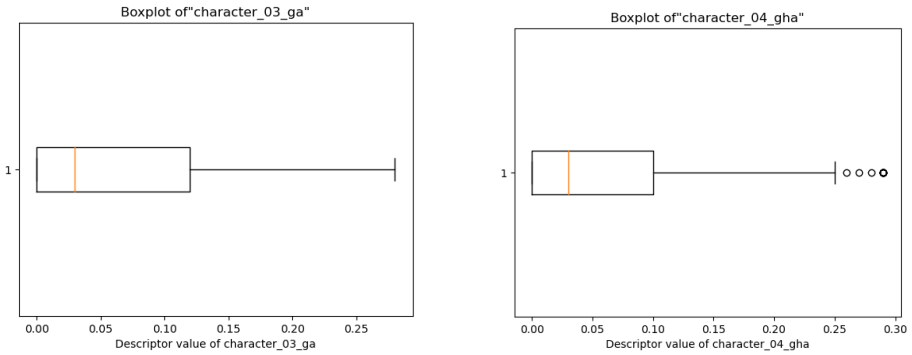
2.1.3 Boxplot

A boxplot is a standardized way of displaying the distribution of data based on a five number summary ("minimum", first quartile [Q1], median, third quartile [Q3] and "maximum").It can tell you about your outliers and what their values are. Boxplots can also tell you if your data is symmetrical, how tightly your data is grouped and if and how your data is skewed.



(a) Character Ka (b) Character Ka

Figure 2.4: Boxplotsof pixel intensity

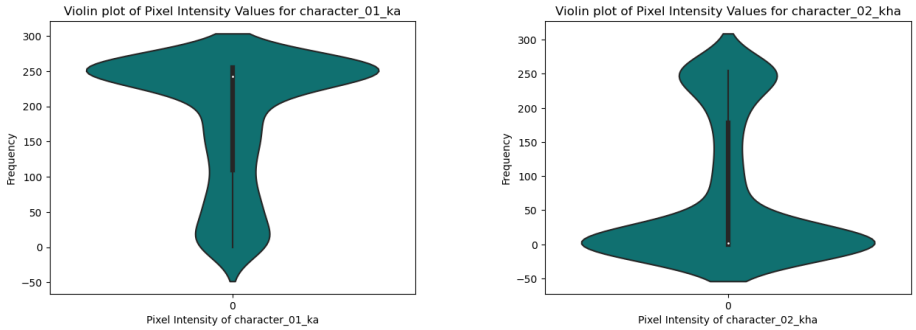


(a) Character ga (b) Character Gha

Figure 2.5: Boxplots of descriptor data

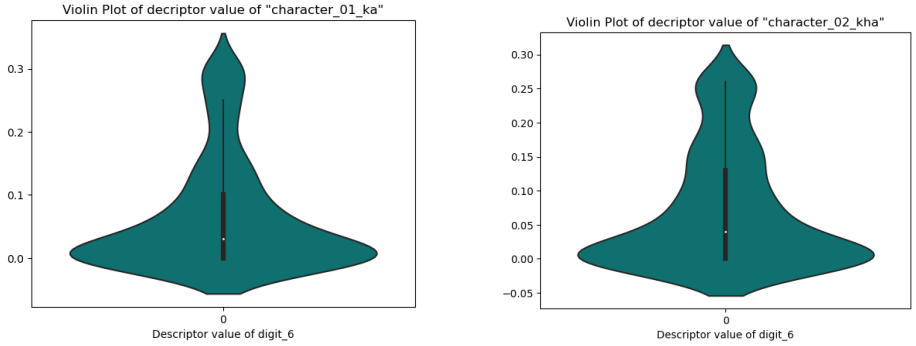
2.1.4 Violin Plot

Violin Plot is a method to visualize the distribution of numerical data of different variables. It is similar to Box Plot but with a rotated plot on each side, giving more information about the density estimate on the y-axis. The density is mirrored and flipped over and the resulting shape is filled in, creating an image resembling a violin.



(a) Character Ka Pixel Intensity (b) Character Kha Pixel Intensity

Figure 2.6: Violin Plot



(a) Character Ka Descriptor Value (b) Character Kha Descriptor Value

Figure 2.7: Violin Plot

■

Chapter 3

Data Pre-processing and Feature Extraction

3.1 Introduction

Once the data has been acquired properly we proceed to pre process the data. In pre-processing stage a series of operations are performed viz binarization, sparcity removal, noise removal, character segmentation, thinning, skew detection. Since the aquired data is already in clean condition, we used sparcity removal and skew detection and correction techniques for pre-processing. The main objective of pre-processing is to organize information so that the subsequent character recognition task becomes simpler. It essentially enhances the image, rendering it suitable for further analysis.

3.2 Data Pre-processsing

3.2.1 Sparcity Removal

Sparsity removal refers to the process of eliminating or reducing the presence of empty or sparse regions in an image before performing character recognition. OCR is a technology that converts printed or handwritten text into machine-encoded text. Sparsity in OCR images refers to areas where the text or characters are missing, faint, or not well-defined. These regions can arise due to various reasons such as poor image quality, low resolution, noise, uneven illumination, or degradation during scanning or digitization.

Removing sparsity is important in OCR because it can significantly affect the accuracy and reliability of the character recognition process. Sparse regions may lead to errors or misinterpretations as the OCR algorithm tries to interpret missing or unclear characters. Therefore, it becomes necessary to pre-process the image and minimize sparsity before feeding it into the OCR system.

3.2.2 Skew Detection and Correction

When a text or written document is fed into scanner either mechanically or manually a few degrees of tilt or skew is unavoidable. In skew angle the text lines in digital image make angle with horizontal direction. A number of methods are available in literature for identifying image skew angles. They are basically categorized on the basis of projection profile analysis, nearest neighbor clustering, Hough transform, cross correlation and morphological transforms. The aforementioned methods correct the detected skew.

3.3 Feature Extraction

Feature extraction attempts to decrease the amount of features in a dataset by generating new features from the ones that already exist (and then removing the original features). The majority of the information in the original set of characteristics should then be able to be summarised by the new, smaller set of features. Through the combination of the original set, a condensed version of the original features can be produced.

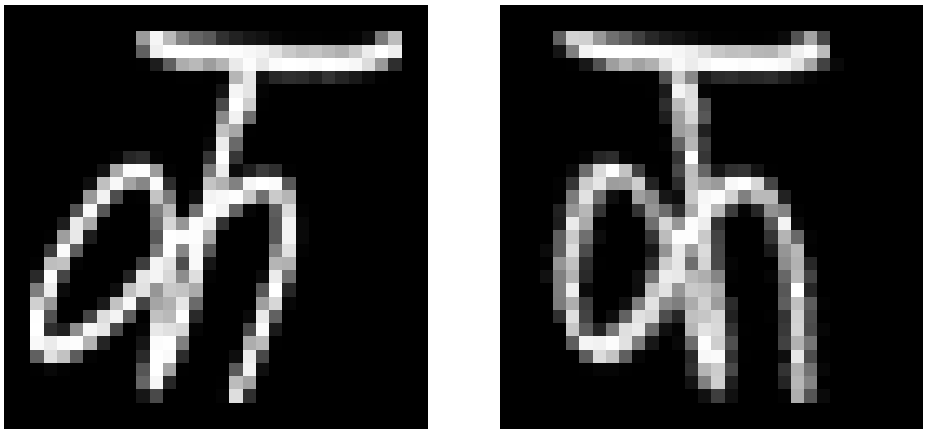
The process of feature extraction begins with an initial set of measured data and creates derived values (features) that are meant to be useful and non-redundant, easing the learning and generalisation processes and, in certain situations, improving human interpretations. Feature extraction is related to dimensionality.

3.3.1 Histogram of Oriented Gradients

Histogram of Oriented Gradients, also known as HOG, is a feature descriptor. It is used in computer vision and image processing for the purpose of object detection. The technique counts occurrences of gradient orientation in the localized portion of an image. This method is quite similar to Edge Orientation Histograms and Scale Invariant Feature Transformation (SIFT). The HOG descriptor focuses on the structure or the shape of an object. It is better than any edge descriptor as it uses magnitude as well as angle of the gradient to compute the features. For the regions of the image it generates histograms using the magnitude and orientations of the gradient.

- **Feature Descriptor:** A feature descriptor is a representation of an image or an image patch that segments away unnecessary information to create a simpler image.

Step 1 : Deskewing Deskewing is the method of removing skew by rotating an image in the opposite direction by the same amount as its skew. This causes the text to run across the page rather than at an angle, creating an image that is aligned both horizontally and vertically.



(a) Original Image (b) Deskewed Image

Figure 3.1: Deskewing

Step 2 : Calculate the Histogram of Oriented Gradients (HOG) descriptor: In this step, we will convert the grayscale image to a feature vector using the HOG feature descriptor.The HOG feature descriptor counts the occurrences of gradient orientation in localized portions of an image.Gradients are the small change in the x and y directions.

```
324
[0.  0.  0.1 0.3 0.7 0.7 0.  0.  0.  0.  0.  0.5 0.5 0.5 0.5 0.  0.  0.
 0.  0.6 0.6 0.4 0.  0.  0.  0.  0.  0.  0.  0.  0.  0.7 0.7 0.3 0.  0.
 0.1 0.2 0.5 0.5 0.5 0.5 0.3 0.1 0.  0.  0.2 0.7 0.7 0.1 0.  0.  0.  0.
 0.1 0.  0.  0.  0.5 0.5 0.5 0.4 0.2 0.4 0.4 0.4 0.2 0.1 0.4 0.4 0.4 0.3
 0.1 0.3 0.6 0.6 0.3 0.  0.  0.  0.  0.1 0.  0.  0.  0.5 0.5 0.5 0.4 0.2
 0.4 0.4 0.4 0.2 0.1 0.4 0.4 0.4 0.3 0.1 0.3 0.6 0.6 0.3 0.  0.  0.  0.
 0.4 0.2 0.2 0.1 0.4 0.4 0.4 0.2 0.4 0.4 0.4 0.2 0.  0.4 0.4 0.4 0.3 0.4
 0.  0.6 0.6 0.6 0.1 0.  0.  0.  0.  0.1 0.  0.  0.  0.7 0.7 0.2 0.2 0.2
 0.4 0.4 0.1 0.  0.4 0.4 0.1 0.2 0.4 0.6 0.6 0.6 0.1 0.1 0.  0.  0.1 0.2
 0.  0.6 0.6 0.4 0.  0.  0.  0.  0.  0.4 0.4 0.4 0.2 0.4 0.4 0.4 0.3 0.4
 0.1 0.  0.  0.3 0.5 0.5 0.5 0.2 0.1 0.  0.2 0.7 0.7 0.1 0.  0.  0.  0.
 0.  0.2 0.6 0.6 0.6 0.1 0.  0.  0.  0.  0.  0.3 0.7 0.7 0.2 0.  0.  0.
 0.1 0.3 0.6 0.6 0.3 0.  0.  0.  0.  0.  0.  0.6 0.6 0.4 0.  0.  0.  0.
 0.  0.  0.6 0.6 0.6 0.  0.  0.  0.  0.1 0.3 0.6 0.6 0.3 0.  0.  0.  0.
 0.  0.  0.6 0.6 0.4 0.  0.  0.  0.  0.  0.  0.6 0.6 0.6 0.  0.  0.  0.
 0.  0.6 0.6 0.6 0.1 0.  0.  0.  0.  0.5 0.3 0.5 0.5 0.2 0.1 0.  0.  0.1
 0.6 0.6 0.2 0.3 0.1 0.  0.1 0.  0.5 0.6 0.6 0.6 0.1 0.1 0.  0.  0.1 0.2
 0.6 0.6 0.3 0.1 0.1 0.1 0.  0.1 0.3 0.7 0.7 0.1 0.  0.  0.  0.  0.  0.2]
```

Figure 3.2: Descriptor Matrix

■

Chapter 4

Classifier Models

4.1 k-Nearest Neighbor

K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and puts the new case into the available category that shows the most similarity. K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm. K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.

The k-NN algorithm calculates the Euclidean distance between test samples and correctly identified samples in the dataset to determine the K-closest neighbors. The Euclidean distance between two samples, $a_1 = (a_{1,1}, \ldots, a_{1,k})$ and $a_2 = (a_{2,1}, \ldots, a_{2,k})$, is defined as:

$$\sqrt{\sum_{j=1}^{k}(a_{1,j} - a_{2,j})^2} \tag{4.1.1}$$

The label of the test sample is then determined based on the majority class of its K-nearest neighbors.

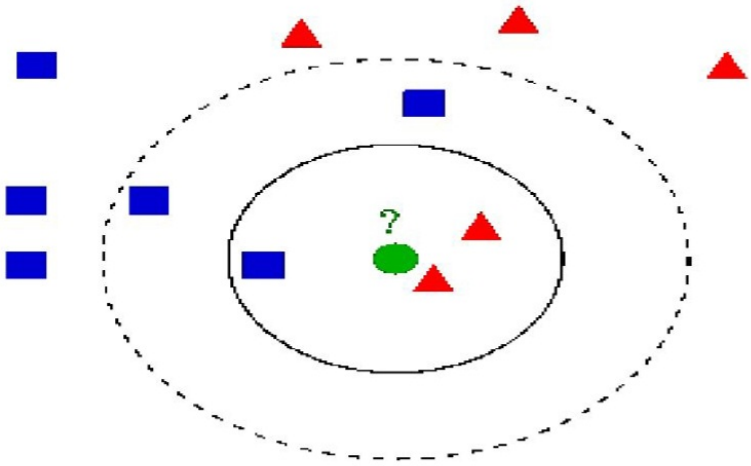


Figure 4.1: k-Nearest Neighbor

In the figure, the test sample is represented as a circle. The first class is represented by squares, and the second class is represented by triangles. If the K-closest neighbors considered are three (i.e., $k = 3$), then the circle is classified as belonging to the class of triangles. If $k = 5$, then the circle is classified as belonging to the class of squares.

KNN has several advantages, including being non-parametric and working well on small sample sizes. However, there are three limitations of KNN:

1. When K is greater than one and the number of training samples of different classes is the same, there will be a tie for the assignment of a specific class.

2. When any input vector (i.e., test sample) is assigned to a class, it does not indicate the intensity of the vector with respect to that class.

3. All classes are considered with equal strength in the assignment of the class label to the test sample.

- **Classification report for descriptor values**

Classification Report for KNN:

	precision	recall	f1-score	support
character_01_ka	0.63	0.76	0.69	584
character_02_kha	0.68	0.81	0.74	563
character_03_ga	0.70	0.84	0.76	471
character_04_gha	0.72	0.79	0.76	510
character_05_kna	0.67	0.67	0.67	500
character_06_cha	0.75	0.77	0.76	562
character_07_chha	0.78	0.71	0.74	560
character_08_ja	0.66	0.74	0.70	525
character_09_jha	0.75	0.74	0.75	555
character_10_yna	0.72	0.75	0.74	535
character_11_taamatar	0.67	0.75	0.71	523
character_12_thaa	0.79	0.83	0.81	587
character_13_daa	0.78	0.93	0.85	503
character_14_dhaa	0.67	0.86	0.75	357
character_15_adna	0.79	0.90	0.84	581
character_16_tabala	0.79	0.89	0.83	510
character_17_tha	0.76	0.87	0.81	579
character_18_da	0.77	0.90	0.83	607
character_19_dha	0.77	0.65	0.71	587
character_20_na	0.75	0.73	0.74	580
character_21_pa	0.79	0.71	0.75	604
character_22_pha	0.72	0.70	0.71	605
character_23_ba	0.74	0.60	0.66	547
character_24_bha	0.71	0.53	0.61	468
character_25_ma	0.78	0.72	0.75	601
character_26_yaw	0.78	0.81	0.79	527
character_27_ra	0.78	0.87	0.83	568
character_28_la	0.82	0.79	0.81	512
character_29_waw	0.77	0.87	0.81	516
character_30_motosaw	0.87	0.88	0.87	561
character_31_petchiryakha	0.91	0.92	0.91	529
character_32_patalosaw	0.81	0.89	0.85	552
character_33_ha	0.83	0.73	0.78	553
character_34_chhya	0.82	0.63	0.71	547
character_35_tra	0.84	0.81	0.83	497
character_36_gya	0.82	0.75	0.78	574
digit_0	0.85	0.75	0.80	573
digit_1	0.82	0.80	0.81	542
digit_2	0.78	0.76	0.77	526
digit_3	0.76	0.78	0.77	515
digit_4	0.82	0.80	0.81	521
digit_5	0.72	0.57	0.63	520
digit_6	0.73	0.71	0.72	491
digit_7	0.76	0.57	0.65	538
digit_8	0.71	0.54	0.61	495
digit_9	0.75	0.68	0.71	537
accuracy			0.76	24798
macro average	0.76	0.76	0.76	24798
weighted average	0.76	0.76	0.76	24798

Figure 4.2: KNN:Classification report for descriptor values

- **Confusuion matrix for descriptor values**

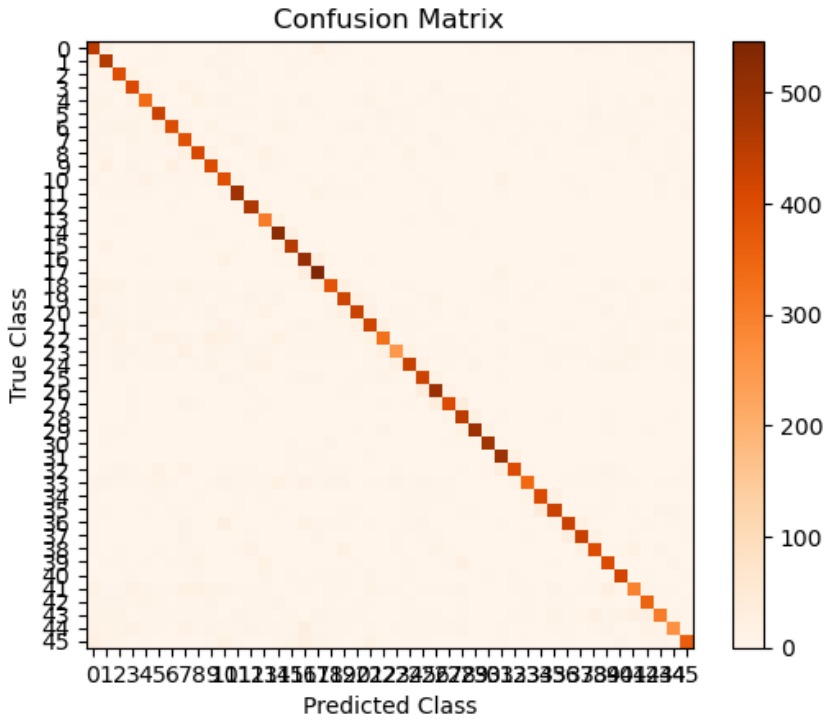


Figure 4.3: KNN:Confusuion matrix for descriptor values

- **Classification report for pixel intensity** report for pixel intensity

Classification Report for KNN:

	precision	recall	f1-score	support
character_01_ka	0.97	0.96	0.97	336
character_02_kha	0.97	0.91	0.94	383
character_03_ga	0.82	0.94	0.87	350
character_04_gha	0.84	0.89	0.86	416
character_05_kna	0.93	0.85	0.89	384
character_06_cha	0.87	0.96	0.91	365
character_07_chha	0.95	0.89	0.92	370
character_08_ja	0.90	0.95	0.93	395
character_09_jha	1.00	0.91	0.95	332
character_10_yna	0.92	0.93	0.92	353
character_11_taamatar	0.77	0.95	0.85	350
character_12_thaa	0.90	0.88	0.89	382
character_13_daa	0.85	0.89	0.87	380
character_14_dhaa	0.88	0.90	0.89	380
character_15_adna	0.90	0.93	0.92	406
character_16_tabala	0.80	0.96	0.87	407
character_17_tha	0.87	0.88	0.87	392
character_18_da	0.92	0.92	0.92	364
character_19_dha	0.94	0.84	0.89	437
character_20_na	0.87	0.85	0.86	379
character_21_pa	0.64	0.94	0.76	378
character_22_pha	0.95	0.84	0.89	267
character_23_ba	0.94	0.74	0.83	385
character_24_bha	0.91	0.89	0.90	380
character_25_ma	0.87	0.81	0.84	372
character_26_yaw	0.83	0.82	0.83	365
character_27_ra	0.95	0.96	0.95	335
character_28_la	0.93	0.93	0.93	401
character_29_waw	0.85	0.85	0.85	402
character_30_motosaw	0.97	0.89	0.93	418
character_31_petchiryakha	0.93	0.81	0.87	383
character_32_patalosaw	0.94	0.87	0.90	406
character_33_ha	0.97	0.84	0.90	361
character_34_chhya	0.99	0.90	0.94	413
character_35_tra	0.97	0.89	0.93	421
character_36_gya	0.97	0.94	0.95	372
digit_0	0.90	1.00	0.95	338
digit_1	0.96	1.00	0.98	216
digit_2	0.95	0.62	0.75	29
digit_3	0.93	0.89	0.91	71
digit_4	1.00	0.98	0.99	411
digit_5	0.94	0.91	0.92	137
digit_6	0.95	0.95	0.95	272
digit_7	0.99	0.97	0.98	361
digit_8	0.87	0.98	0.92	261
digit_9	0.99	0.99	0.99	370
accuracy			0.90	16086
macro average	0.91	0.90	0.90	16086
weighted average	0.91	0.90	0.90	16086

Figure 4.4: KNN:Classification report for pixel intensity

- **Confusuion matrix for pixel intensity**

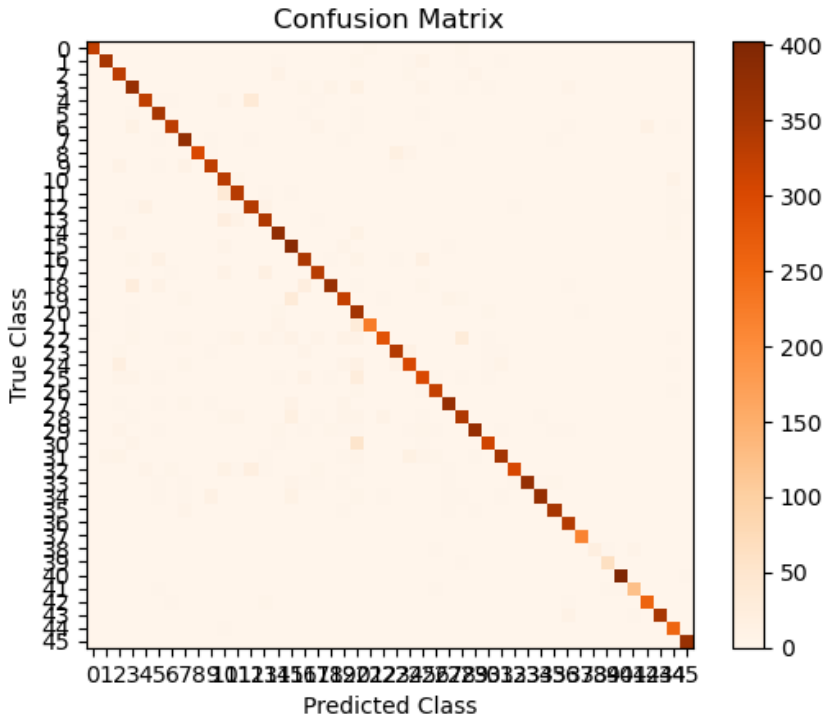


Figure 4.5: KNN:Confusuion matrix for pixel intensity

Result:

The k-Nearest Neighbors (k-NN) classifier achieved an overall accuracy of 76% for descriptor data and 90% for pixel intensity data. It demonstrated effective classification performance by assigning instances to their nearest neighbors based on neighbors. The model showcases its ability to classify the target variables accurately, making it suitable for tasks that require pattern recognition and similarity-based predictions.

4.2 Random Forest

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve complex problem and to improve the performance of the model. As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of over fitting.

Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction. One big advantage of random forest is that it can be used for both classification and regression problems, which form the majority of current machine learning systems.

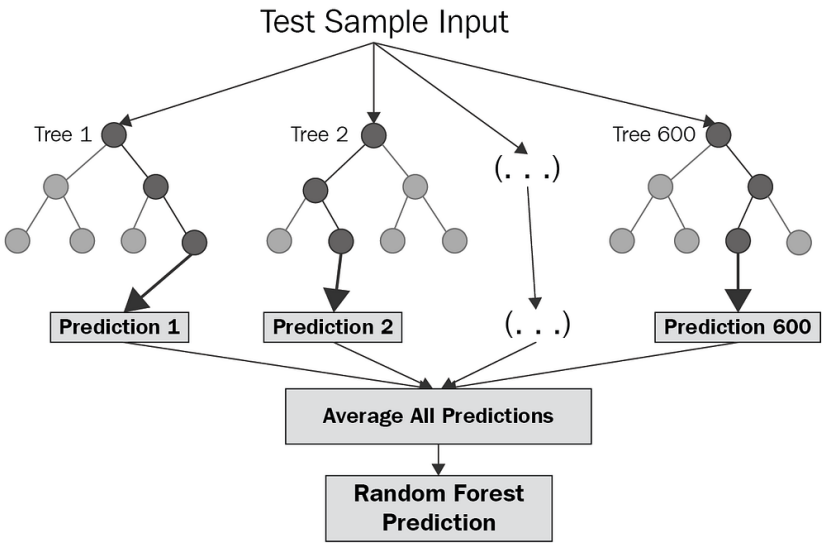


Figure 4.6: Random Forest

In Random Forest each tree is grown as follows:

1. If the number of cases in the training set is N, sample N cases at random but with replacement, from the original data. This sample will be the training set for growing there.

2. If there are M input variables, a number $m << M$ is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing.

3. Each tree is grown to the largest extent possible. There is no pruning.

- **Classification report for descriptor values**

Classification Report for Random Forest:

	precision	recall	f1-score	support
character_01_ka	0.65	0.64	0.65	379
character_02_kha	0.70	0.70	0.70	367
character_03_ga	0.71	0.70	0.71	295
character_04_gha	0.72	0.76	0.74	353
character_05_kna	0.68	0.59	0.63	340
character_06_cha	0.64	0.68	0.66	379
character_07_chha	0.64	0.56	0.60	386
character_08_ja	0.56	0.59	0.57	345
character_09_jha	0.57	0.71	0.63	383
character_10_yna	0.58	0.70	0.64	338
character_11_taamatar	0.63	0.69	0.66	335
character_12_thaa	0.75	0.79	0.77	389
character_13_daa	0.72	0.92	0.81	319
character_14_dhaa	0.72	0.78	0.75	238
character_15_adna	0.66	0.89	0.76	380
character_16_tabala	0.75	0.86	0.81	330
character_17_tha	0.75	0.87	0.81	388
character_18_da	0.77	0.89	0.82	401
character_19_dha	0.68	0.50	0.58	396
character_20_na	0.70	0.59	0.64	400
character_21_pa	0.72	0.64	0.68	401
character_22_pha	0.73	0.65	0.69	398
character_23_ba	0.59	0.47	0.52	368
character_24_bha	0.67	0.43	0.52	304
character_25_ma	0.65	0.67	0.66	397
character_26_yaw	0.75	0.78	0.77	344
character_27_ra	0.75	0.86	0.80	370
character_28_la	0.73	0.76	0.75	352
character_29_waw	0.73	0.85	0.79	355
character_30_motosaw	0.77	0.87	0.81	379
character_31_petchiryakha	0.83	0.88	0.86	336
character_32_patalosaw	0.77	0.90	0.83	387
character_33_ha	0.77	0.65	0.71	368
character_34_chhya	0.78	0.50	0.61	385
character_35_tra	0.71	0.78	0.75	347
character_36_gya	0.76	0.70	0.73	386
digit_0	0.71	0.79	0.75	384
digit_1	0.76	0.76	0.76	361
digit_2	0.66	0.74	0.70	360
digit_3	0.64	0.78	0.70	341
digit_4	0.64	0.73	0.68	346
digit_5	0.64	0.43	0.51	357
digit_6	0.67	0.56	0.61	326
digit_7	0.73	0.46	0.56	360
digit_8	0.68	0.46	0.55	324
digit_9	0.72	0.65	0.68	355
accuracy			0.70	16532
macro average	0.70	0.70	0.69	16532
weighted average	0.70	0.70	0.69	16532

Figure 4.7: Random Forest:Classification report for descriptor values

- **Confusuion matrix for descriptor values**

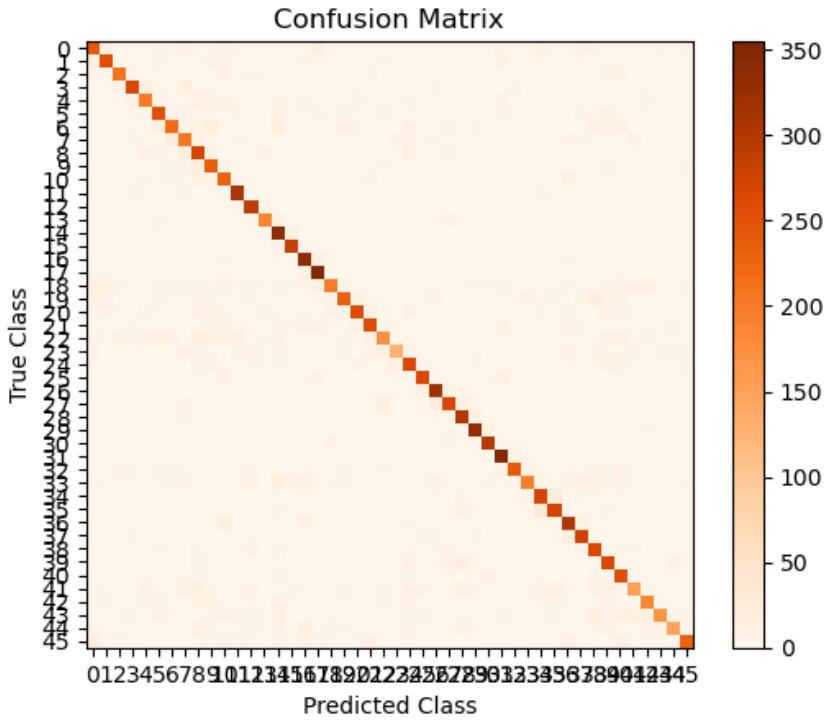


Figure 4.8: Random Forest:Confusuion matrix for descriptor values

- **Classification report for pixel intensity**

Classification Report for Random Forest:

	precision	recall	f1-score	support
character_01_ka	0.91	0.93	0.92	336
character_02_kha	0.90	0.87	0.88	383
character_03_ga	0.89	0.88	0.89	350
character_04_gha	0.82	0.80	0.81	416
character_05_kna	0.89	0.88	0.89	384
character_06_cha	0.86	0.92	0.89	365
character_07_chha	0.90	0.84	0.87	370
character_08_ja	0.93	0.93	0.93	395
character_09_jha	0.95	0.91	0.93	332
character_10_yna	0.92	0.92	0.92	353
character_11_taamatar	0.94	0.93	0.94	350
character_12_thaa	0.90	0.94	0.92	382
character_13_daa	0.90	0.88	0.89	380
character_14_dhaa	0.92	0.93	0.93	380
character_15_adna	0.91	0.94	0.93	406
character_16_tabala	0.93	0.97	0.95	407
character_17_tha	0.84	0.85	0.85	392
character_18_da	0.86	0.90	0.88	364
character_19_dha	0.89	0.83	0.86	437
character_20_na	0.88	0.89	0.89	379
character_21_pa	0.78	0.91	0.84	378
character_22_pha	0.92	0.91	0.91	267
character_23_ba	0.90	0.78	0.83	385
character_24_bha	0.90	0.90	0.90	380
character_25_ma	0.88	0.83	0.85	372
character_26_yaw	0.82	0.84	0.83	365
character_27_ra	0.93	0.94	0.94	335
character_28_la	0.93	0.94	0.93	401
character_29_waw	0.87	0.82	0.84	402
character_30_motosaw	0.85	0.83	0.84	418
character_31_petchiryakha	0.79	0.92	0.85	383
character_32_patalosaw	0.88	0.84	0.86	406
character_33_ha	0.91	0.85	0.88	361
character_34_chhya	0.89	0.89	0.89	413
character_35_tra	0.92	0.90	0.91	421
character_36_gya	0.89	0.94	0.92	372
digit_0	0.96	0.99	0.98	338
digit_1	0.92	0.99	0.95	216
digit_2	0.92	0.41	0.57	29
digit_3	0.94	0.63	0.76	71
digit_4	0.95	0.97	0.96	411
digit_5	0.89	0.86	0.87	137
digit_6	0.96	0.91	0.93	272
digit_7	0.95	0.98	0.96	361
digit_8	0.94	0.95	0.95	261
digit_9	0.95	0.98	0.96	370
accuracy			0.90	16086
macro average	0.90	0.88	0.89	16086
weighted average	0.90	0.90	0.90	16086

Figure 4.9: Random Forest:Classification report for pixel intensity

- **Confusuion matrix for pixel intensity**

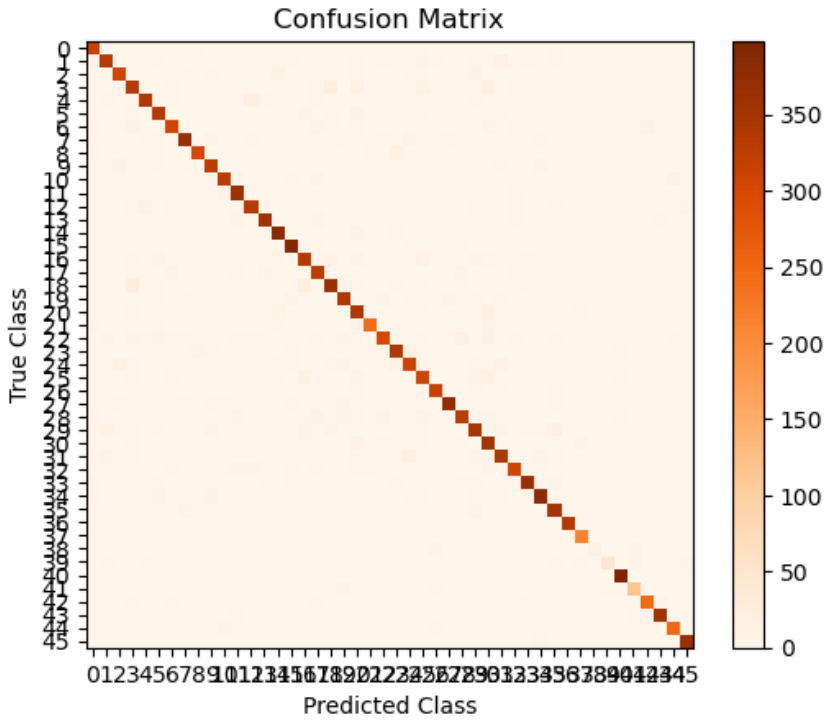


Figure 4.10: Random Forest:Confusuion matrix for pixel intensity

Result:

The Random Forest classifier achieved an overall accuracy of 70% for descriptor data and 90% for pixel intensity data. Through its ensemble approach, combining multiple decision trees, the model exhibited remarkable performance in accurately categorizing the target variables into their respective classes. This accuracy reflects the classifier's ability to capture complex patterns and make reliable predictions.

4.3 Support Vector Machine

Support Vector Machine (SVM) is a supervised machine learning algorithm used for both classification and regression. Though we say regression problems as well it's best suited for classification. The objective of the SVM algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points. The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane. It becomes difficult to imagine when the number of features exceeds three.

Let's consider two independent variables x1, x2, and one dependent variable which is either a blue circle or a red circle.

From figure 4.11 (a) it's very clear that there are multiple lines (our hyperplane here is a line because we are considering only two input features x1, x2) that segregate our data points or do a classification between red and blue circles. So how do we choose the best line or in general the best hyperplane that segregates our data points?

Working of Support Vector Machine

One reasonable choice as the best hyperplane is the one that represents the largest separation or margin between the two classes.

So we choose the hyperplane whose distance from it to the nearest data point on each side is maximized. If such a hyperplane exists it is known as the maximum-margin hyperplane/hard margin. So from figure 4.11 (b), we choose L2. Let's consider a scenario like shown in figure 4.12 (a).

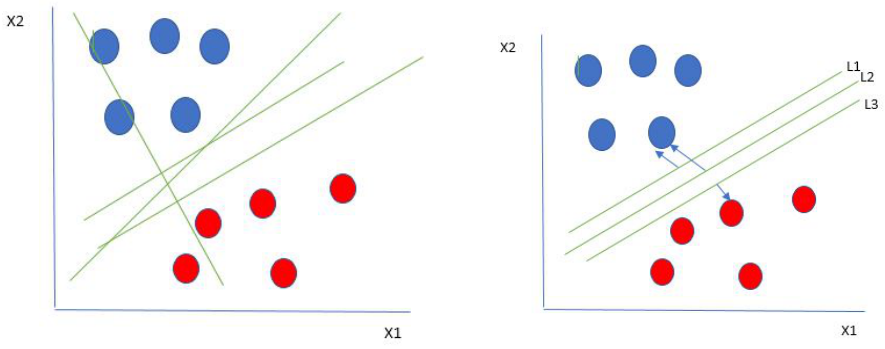
Here we have one blue ball in the boundary of the red ball. So how does

SVM classify the data? It's simple! The blue ball in the boundary of red ones is an outlier of blue balls. The SVM algorithm has the characteristics to ignore the outlier and finds the best hyperplane that maximizes the margin. SVM is robust to outliers.

So in this type of data point what SVM does is, finds the maximum margin as done with previous data sets along with that it adds a penalty each time a point crosses the margin. So the margins in these types of cases are called soft margins. When there is a soft margin to the data set, the SVM tries to minimize $\frac{1}{\text{margin}} + \Lambda(\sum \text{penalty})$. Hinge loss is a commonly used penalty. If no violations no hinge loss. If violations hinge loss proportional to the distance of violation.

Till now, we were talking about linearly separable data(the group of blue balls and red balls are separable by a straight line/linear line). What to do if data are not linearly separable?

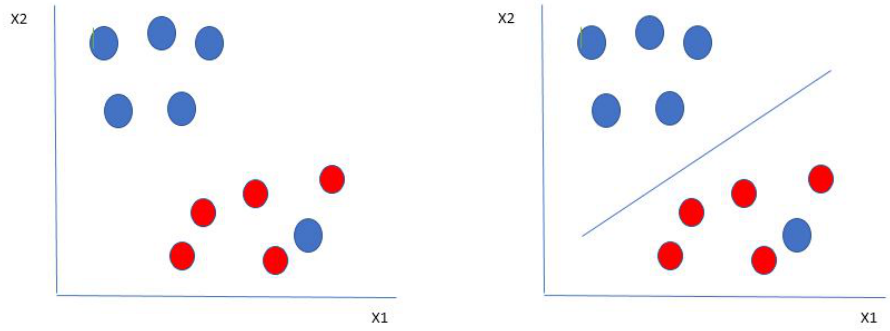
Say, our data is shown in the figure 4.13 (a). SVM solves this by creating a new variable using a kernel. We call a point x_i on the line and we create a new variable y_i as a function of distance from origin O. So if we plot this we get something like as shown in figure 4.13 (b).



(a) Linearly Separable Data points

(b) Multiple hyperplane separate the data from two classes

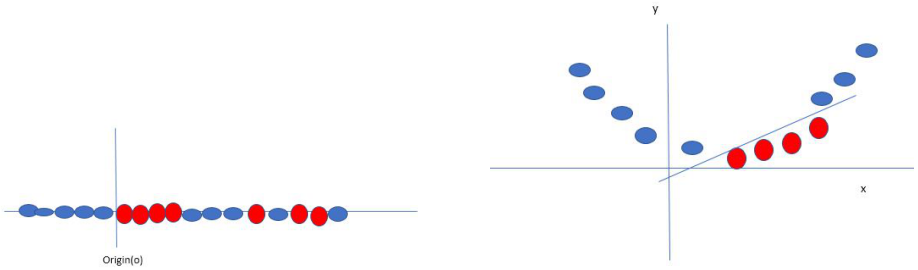
Figure 4.11: Support Vector Machine



(a) Selecting hyperplane for data with outlier

(b) Hyperplane which is most optimized one

Figure 4.12: Support Vector Machine



(a) Original 1D dataset for classification

(b) Mapping 1D data to 2D to become able to separate the two classes

Figure 4.13: Support Vector Machine

- **Classification report for descriptor values**

Classification Report for SVM:

	precision	recall	f1-score	support
character_01_ka	0.40	0.40	0.40	379
character_02_kha	0.40	0.49	0.44	367
character_03_ga	0.50	0.58	0.54	295
character_04_gha	0.48	0.56	0.52	353
character_05_kna	0.39	0.42	0.40	340
character_06_cha	0.41	0.49	0.45	379
character_07_chha	0.37	0.39	0.38	386
character_08_ja	0.44	0.43	0.44	345
character_09_jha	0.47	0.54	0.50	383
character_10_yna	0.39	0.45	0.42	338
character_11_taamatar	0.42	0.46	0.44	335
character_12_thaa	0.54	0.58	0.56	389
character_13_daa	0.65	0.75	0.70	319
character_14_dhaa	0.60	0.64	0.62	238
character_15_adna	0.68	0.80	0.73	380
character_16_tabala	0.59	0.57	0.58	330
character_17_tha	0.72	0.76	0.74	388
character_18_da	0.67	0.67	0.67	401
character_19_dha	0.28	0.26	0.27	396
character_20_na	0.46	0.35	0.40	400
character_21_pa	0.52	0.45	0.48	401
character_22_pha	0.46	0.47	0.46	398
character_23_ba	0.35	0.31	0.33	368
character_24_bha	0.35	0.29	0.32	304
character_25_ma	0.59	0.54	0.56	397
character_26_yaw	0.55	0.56	0.55	344
character_27_ra	0.68	0.78	0.72	370
character_28_la	0.67	0.64	0.65	352
character_29_waw	0.65	0.68	0.66	355
character_30_motosaw	0.70	0.73	0.71	379
character_31_petchiryakha	0.76	0.75	0.75	336
character_32_patalosaw	0.72	0.79	0.75	387
character_33_ha	0.58	0.54	0.56	368
character_34_chhya	0.42	0.34	0.37	385
character_35_tra	0.59	0.59	0.59	347
character_36_gya	0.52	0.52	0.52	386
digit_0	0.55	0.59	0.57	384
digit_1	0.65	0.63	0.64	361
digit_2	0.51	0.56	0.54	360
digit_3	0.58	0.66	0.62	341
digit_4	0.56	0.58	0.57	346
digit_5	0.48	0.26	0.33	357
digit_6	0.36	0.32	0.34	326
digit_7	0.38	0.28	0.32	360
digit_8	0.44	0.31	0.36	324
digit_9	0.46	0.40	0.43	355
accuracy			0.52	16532
macro average	0.52	0.52	0.52	16532
weighted average	0.52	0.52	0.52	16532

Figure 4.14: SVM:Classification report for descriptor values

- **Confusuion matrix for descriptor values**

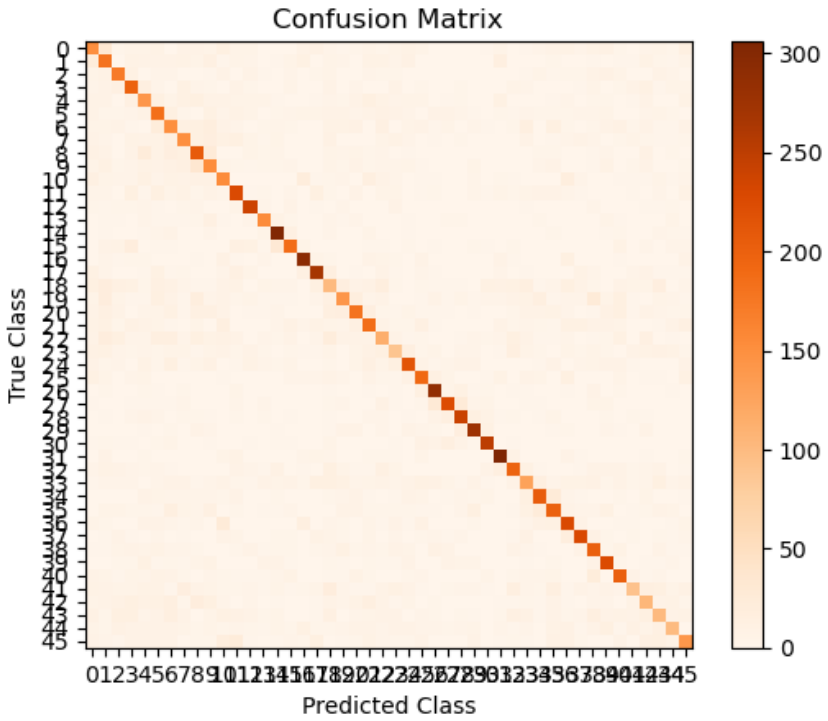


Figure 4.15: SVM:Confusuion matrix for descriptor values

- **Classification report for pixel intensity**

Classification Report for SVM:

	precision	recall	f1-score	support
character_01_ka	0.80	0.91	0.85	336
character_02_kha	0.67	0.73	0.70	383
character_03_ga	0.71	0.77	0.74	350
character_04_gha	0.64	0.72	0.68	416
character_05_kna	0.69	0.77	0.73	384
character_06_cha	0.77	0.88	0.82	365
character_07_chha	0.75	0.70	0.72	370
character_08_ja	0.79	0.90	0.84	395
character_09_jha	0.91	0.86	0.88	332
character_10_yna	0.78	0.84	0.81	353
character_11_taamatar	0.86	0.91	0.89	350
character_12_thaa	0.82	0.87	0.84	382
character_13_daa	0.70	0.72	0.71	380
character_14_dhaa	0.84	0.87	0.86	380
character_15_adna	0.78	0.85	0.81	406
character_16_tabala	0.80	0.85	0.83	407
character_17_tha	0.64	0.72	0.68	392
character_18_da	0.68	0.77	0.73	364
character_19_dha	0.73	0.63	0.68	437
character_20_na	0.73	0.71	0.72	379
character_21_pa	0.68	0.73	0.71	378
character_22_pha	0.79	0.82	0.80	267
character_23_ba	0.72	0.65	0.68	385
character_24_bha	0.78	0.79	0.78	380
character_25_ma	0.64	0.64	0.64	372
character_26_yaw	0.61	0.50	0.55	365
character_27_ra	0.88	0.86	0.87	335
character_28_la	0.83	0.78	0.80	401
character_29_waw	0.74	0.62	0.68	402
character_30_motosaw	0.76	0.65	0.70	418
character_31_petchiryakha	0.79	0.78	0.78	383
character_32_patalosaw	0.67	0.54	0.60	406
character_33_ha	0.83	0.72	0.77	361
character_34_chhya	0.85	0.80	0.82	413
character_35_tra	0.78	0.67	0.72	421
character_36_gya	0.83	0.79	0.81	372
digit_0	0.97	0.99	0.98	338
digit_1	0.90	0.98	0.94	216
digit_2	0.57	0.72	0.64	29
digit_3	0.74	0.69	0.72	71
digit_4	0.95	0.92	0.94	411
digit_5	0.89	0.85	0.87	137
digit_6	0.88	0.88	0.88	272
digit_7	0.96	0.96	0.96	361
digit_8	0.90	0.95	0.93	261
digit_9	0.94	0.92	0.93	370
accuracy			0.78	16086
macro average	0.78	0.79	0.78	16086
weighted average	0.78	0.78	0.78	16086

Figure 4.16: SVM:Classification report for pixel intensity

- **Confusuion matrix for pixel intensity**

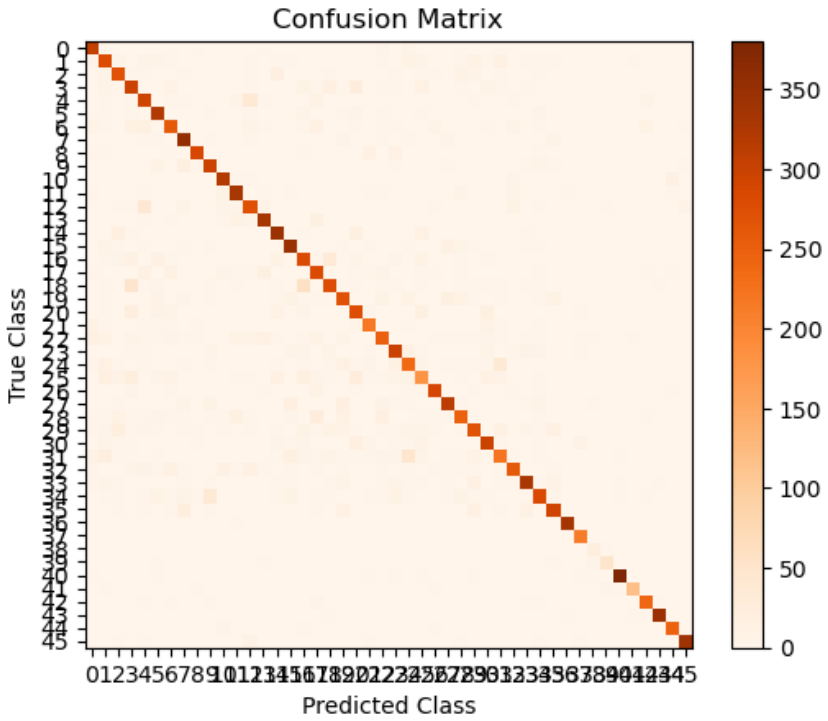


Figure 4.17: SVM:Confusuion matrix for pixel intensity

Result:

The Support Vector Machine (SVM) classifier achieved an overall accuracy of 52% for descriptor data and 78%for pixel intensity data. The results suggest that there may be limitations in accurately classifying the target variables using the provided descriptor value data.With its robust classification performance, the model demonstrated the ability to accurately classify the target variables. The SVM classifier effectively identified decision boundaries and maximized the margin between classes, resulting in a high level of accuracy for pixel intensity data.

4.4 Convolutional Neural Network

Convolutional Neural Network (CNN)is one of the main categories to do image classification and image recognition in neural networks. Scene labeling, objects detections, and face recognition, etc., are some of the areas where convolutional neural networks are widely used.

CNN takes an image as input, which is classified and process under a certain category such as dog, cat, lion, tiger, etc. The computer sees an image as an array of pixels and depends on the resolution of the image. Based on image resolution, it will see as $h * w * d$, where h=height w=width and d=dimension. For example, An RGB image is $6 * 6 * 3$ array of the matrix, and the grayscale image is $4 * 4 * 1$ array of the matrix.

In CNN, each input image will pass through a sequence of convolution layers along with pooling, fully connected layers, filters (Also known as kernels). After that, we will apply the Soft-max function to classify an object with probabilistic values 0 and 1.

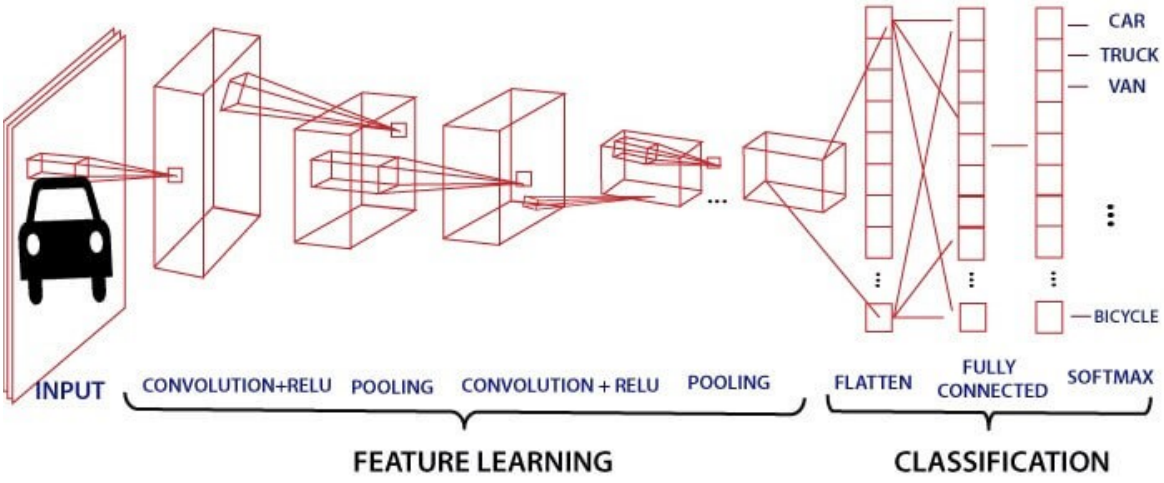


Figure 4.18: CNN

4.4.1 Convolution Layer

Convolution layer is the first layer to extract features from an input image. By learning image features using a small square of input data, the convolutional layer preserves the relationship between pixels. It is a mathematical operation which takes two inputs such as image matrix and a kernel or filter.

1. The dimension of the image matrix is $h * w * d$.

2. The dimension of the filter is $f_h * f_w * d$.

3. The dimension of the output is $(h - f_h + 1) * (w - f_w + 1) * 1$.

Let's start with consideration a $5 * 5$ image whose pixel values are 0, 1, and filter matrix $3 * 3$ as:

The convolution of $5 * 5$ image matrix multiplies with $3 * 3$ filter matrix is called "Features Map" and show as an output.

Convolution of an image with different filters can perform an operation such as blur, sharpen, and edge detection by applying filters.

4.4.2 Strides

Stride is the number of pixels which are shift over the input matrix. When the stride is equaled to 1, then we move the filters to 1 pixel at a time and similarly, if the stride is equaled to 2, then we move the filters to 2 pixels at a time. The following figure shows that the convolution would work with a stride of 2.

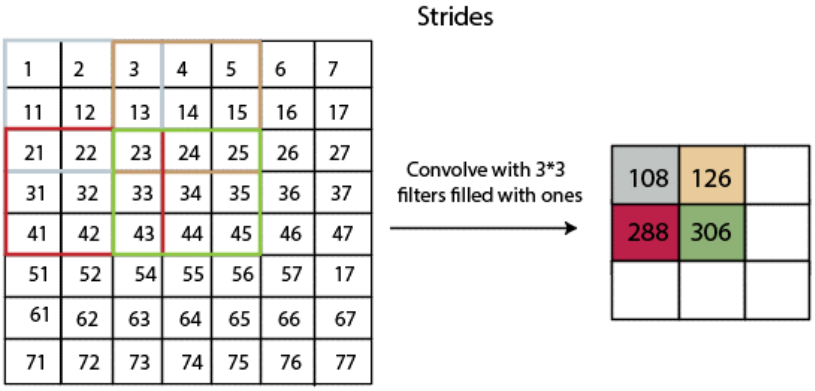


Figure 4.19: Strides

4.4.3 Padding

Padding plays a crucial role in building the convolutional neural network. If the image will get shrink and if we will take a neural network with 100's of layers on it, it will give us a small image after filtered in the end.

If we take a three by three filter on top of a grayscale image and do the convolving then what will happen?

It is clear from the above picture that the pixel in the corner will only get covers one time, but the middle pixel will get covered more than once. It means that we have more information on that middle pixel, so there are two downsides:

- Shrinking outputs.

- Losing information on the corner of the image.

To overcome this, we have introduced padding to an image. "Padding is an additional layer which can add to the border of an image."

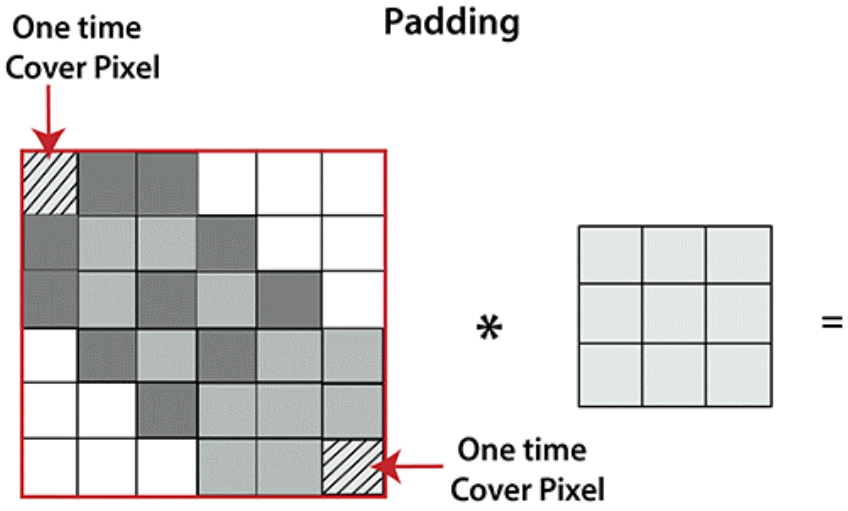


Figure 4.20: Padding

4.4.4 Pooling Layer

Pooling layer plays an important role in pre-processing of an image. Pooling layer reduces the number of parameters when the images are too large. Pooling is "downscaling" of the image obtained from the previous layers. It can be compared to shrinking an image to reduce its pixel density. Spatial pooling is also called downsampling or subsampling, which reduces the dimensionality of each map but retains the important information. There are the following types of spatial pooling:

Max Pooling

Max pooling is a sample-based discretization process. Its main objective is to downscale an input representation, reducing its dimensionality and allowing for the assumption to be made about features contained in the sub-region binned. Max pooling is done by applying a max filter to non-overlapping sub-regions of the initial representation.

Average Pooling

Down-scaling will perform through average pooling by dividing the input into rectangular pooling regions and computing the average values of each region.

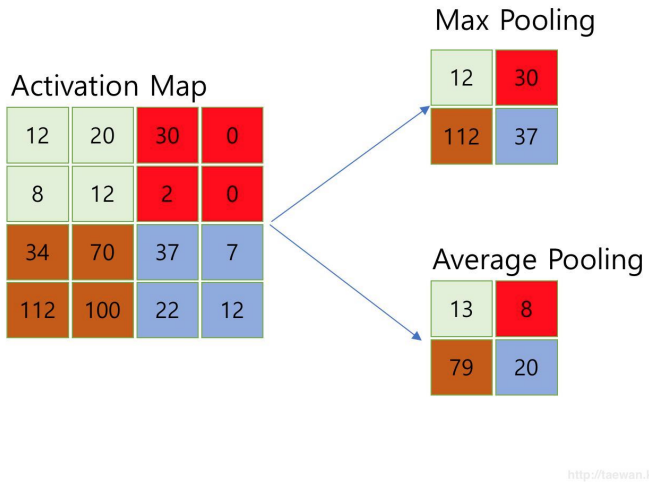


Figure 4.21: Pooling

Sum Pooling

The sub-region for sum pooling or mean pooling are set exactly the same as for max-pooling but instead of using the max function we use sum or mean.

4.4.5 Fully Connected Layer

The fully connected layer is a layer in which the input from the other layers will be flattened into a vector and sent. It will transform the output into the desired number of classes by the network.

In the above diagram, the feature map matrix will be converted into the

vector such as $x_1, x_2, x_3...x_n$ with the help of fully connected layers. We will combine features to create a model and apply the activation function such as softmax or sigmoid to classify the outputs as a car, dog, truck, etc.

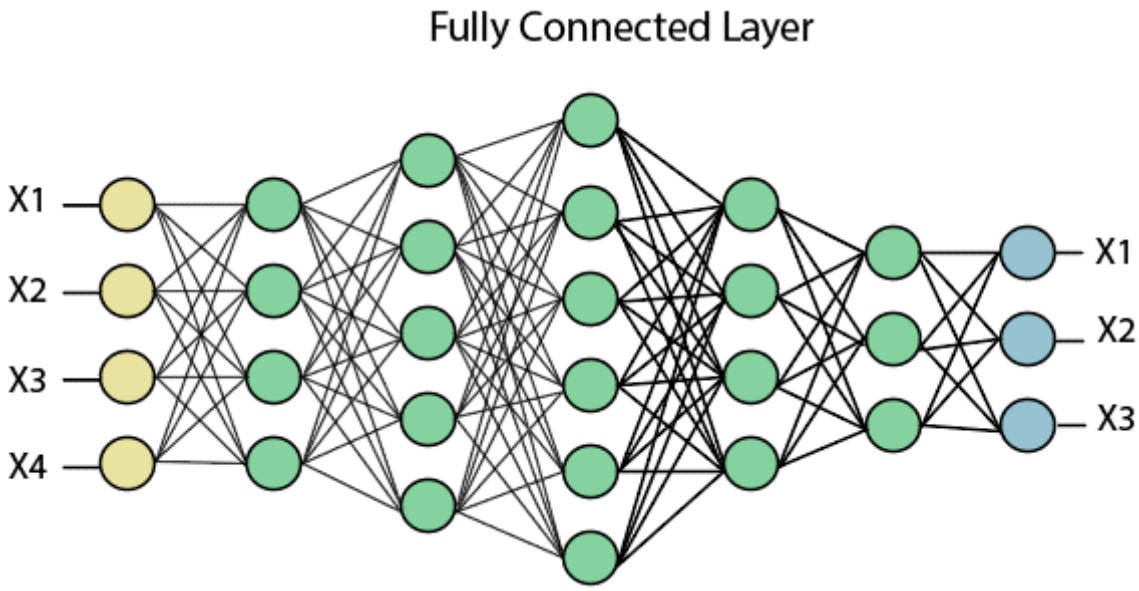


Figure 4.22: Fully Connected Layers

CNN output for descriptor data:

```
Model: "sequential"
_________________________________________________________________
 Layer(type)                 Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 18, 18, 64)         640

 conv2d_1 (Conv2D)           (None, 18, 18, 64)        36928

 Batch normalization         (None, 18, 18, 64)         256

 max_pooling2d               (None, 9, 9, 64)             0

 conv2d_2 (Conv2D)            (None, 9, 9, 128)        73856

 conv2d_3 (Conv2D)            (None, 9, 9, 128)       147584

 batch_normalization_1        (None, 9, 9, 128)          512

 max_pooling2d_1              (None, 4, 4, 128)            0

 conv2d_4 (Conv2D)            (None, 4, 4, 256)       295168

 conv2d_5 (Conv2D)            (None, 4, 4, 256)       590080

 batch_normalization_2        (None, 4, 4, 256)         1024

 max_pooling2d_2              (None, 2, 2, 256)            0

 conv2d_6 (Conv2D)            (None, 2, 2, 512)       1180160

 conv2d_7 (Conv2D)            (None, 2, 2, 512)       2359808

 batch_normalization_3        (None, 2, 2, 512)          2048

 global_max_pooling2d          (None, 512)                0

=================================================================
Total params: 4,688,064
Trainable params: 4,686,144
Non-trainable params: 1,920
```

```
Epoch 1/5
53/53 [==============================] - 314s 6s/step - loss: 4.0868 - accuracy:
0.2111 - val_loss: 4.5717 - val_accuracy: 0.0845
Epoch 2/5
53/53 [==============================] - 299s 6s/step - loss: 2.2115 - accuracy:
0.5181 - val_loss: 2.2404 - val_accuracy: 0.4958
Epoch 3/5
53/53 [==============================] - 303s 6s/step - loss: 1.4821 - accuracy:
0.6581 - val_loss: 1.9649 - val_accuracy: 0.5634
Epoch 4/5
53/53 [==============================] - 319s 6s/step - loss: 1.0811 - accuracy:
0.7409 - val_loss: 1.7799 - val_accuracy: 0.5926
Epoch 5/5
53/53 [==============================] - 331s 6s/step - loss: 0.8004 - accuracy:
0.8046 - val_loss: 1.6502 - val_accuracy: 0.6267
```

Figure 4.23: CNN for descriptor values

CNN output for pixel intensity data data:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d (Conv2D)              (None, 28, 28, 64)          640

 conv2d_1 (Conv2D)            (None, 28, 28, 64)        36928

 batch_normalisation          (None, 28, 28, 64)          256

 max_pooling2d                (None, 14, 14, 64)            0

 conv2d_2 (Conv2D)            (None, 14, 14, 128)       73856

 conv2d_3 (Conv2D)            (None, 14, 14, 128)      147584

 batch_normalisation_1        (None, 14, 14, 128)         512

 max_pooling2d_1              (None, 7, 7, 128)             0

 conv2d_4 (Conv2D)            (None, 7, 7, 256)        295168

 conv2d_5 (Conv2D)            (None, 7, 7, 256)        590080

 batch_normalisation_2        (None, 7, 7, 256)          1024

 max_pooling2d_2              (None, 3, 3, 256)             0

 conv2d_6 (Conv2D)            (None, 3, 3, 512)       1180160

 conv2d_7 (Conv2D)            (None, 3, 3, 512)       2359808

 batch_normalisation_3        (None, 3, 3, 512)          2048

 global_max_pooling2d         (None, 512)                   0

=================================================================
Total params: 4,688,064
Trainable params: 4,686,144
Non-trainable params: 1,920
_________________________________________________________________
Epoch 1/5
49/49 [==============================] - 434s 9s/step - loss: 1.5704 - accuracy:
0.6915 - val_loss: 0.4445 - val_accuracy: 0.9044
Epoch 2/5
49/49 [==============================] - 445s 9s/step - loss: 0.2301 - accuracy:
0.9492 - val_loss: 0.2349 - val_accuracy: 0.9446
Epoch 3/5
49/49 [==============================] - 444s 9s/step - loss: 0.1486 - accuracy:
0.9738 - val_loss: 0.2497 - val_accuracy: 0.9419
Epoch 4/5
49/49 [==============================] - 440s 9s/step - loss: 0.1252 - accuracy:
0.9834 - val_loss: 0.1934 - val_accuracy: 0.9637
Epoch 5/5
49/49 [==============================] - 451s 9s/step - loss: 0.1121 - accuracy:
0.9878 - val_loss: 0.1828 - val_accuracy: 0.9669
```

Figure 4.24: CNN for pixel intensity values

Result:

The Convolutional Neural Network (CNN) achieved an impressive overall accuracy of 80% for descriptor data and 98% for pixel intensity datas. This deep learning model effectively captured complex spatial relationships within the input data, demonstrating its ability to extract meaningful features and make accurate predictions. The CNN's architecture, consisting of convolutional, pooling, and fully connected layers, proved to be highly effective in leveraging hierarchical representations and achieving state-of-the-art performance on the given dataset.

■

Chapter 5

Conclusion

1. Beginning with the data, we have got two types of data. One includes the images (descriptor data) and the other one includes numeric values of pixel intensities of those images in a CSV file.

2. Using feature extraction (Histogram of Oriented Gradients), the extracted features of the image data which are in the form of descriptor matrix can be further used to classify and identify the text characters.

3. Working on the then obtained descriptor matrices using machine learning an deep learning classifiers viz. k Nearest Neighbor, Random Forest, Support Vector Machine (Linear Kernel) and Convolutional Neural Network, the results obtained are summerized as follows.

Classifiers	Accuracy for descriptor data	Accuracy for pixel intensity data
kNN	76%	90%
Random Forest	70%	90%
SVM	52%	78%
CNN	80%	98%

Table 5.1: Classifiers with respective accuracies

4. In conclusion, the experimental results indicate that the accuracy achieved using descriptor matrix is consistently lower compared to that obtained with pixel intensity data. This suggests that the information captured by descriptor matrix alone may not be sufficient to accurately represent and distinguish between the underlying patterns and features in the data. Therefore, when aiming for higher accuracy in data analysis or classification tasks, it is recommended to consider incorporating pixel intensity data alongside or instead of relying solely on descriptor matrix.

■

Chapter 6

Limitations and Future Scope

6.1 Limitations

- **Poor Image Quality:** The accuracy of OCR is highly dependent on the input image's quality. Inaccurate recognition results may be caused by low-resolution scans, blurry or distorted text, and uneven illumination.

- **Ambiguous Words:** Words with ambiguous characters or similar shapes may be difficult for OCR systems to recognise. In certain circumstances, the OCR algorithm may erroneously identify the proper characters or misread the word entirely.

- **Handwritten Text:** Due to the wide range of handwriting styles and personal writing habits, OCR for handwritten text poses significant hurdles. When compared to printed language, handwritten text often has a lower recognition accuracy.

- **Noise and Background Interference:** Background elements in the document, such as patterns, textures, or watermarks, can cause noise or interfere with OCR accuracy. These components have the potential to confuse the OCR algorithm and result in inaccurate character recogni-

tion.

- **Processing Power and Efficiency:** The efficiency of the OCR algorithm and the available computational resources influence how quickly features from an image can be extracted. Slower feature extraction may be caused by insufficient processing power or less effective algorithms, which may have an impact on the performance of OCR as a whole.

6.2 Future Scope

1. **Reduction of Processing Time and Misclassification:** As a future scope for the project, there is potential to focus on reducing the processing time required for OCR feature extraction and analysis.The exploration of algorithmic advancements and the application of effective methods for data augmentation can accomplish this. Furthermore, efforts should be focused on reducing misclassification errors, which can be handled by sophisticated algorithms and techniques like ensemble learning and error correction mechanisms. These improvements will aid in making the OCR system more accurate and efficient, allowing for quicker processing and better recognition capabilities.

2. **Composite Models:** By combining several OCR models or techniques, composite models may increase the recognition accuracy. The OCR system can perform better across a wider range of document types and fonts by utilizing the advantages of various models.

3. **Better Image Extraction Methods:** Better image extraction techniques can result in improved OCR performance. To enhance the quality of the input images prior to feature extraction, these techniques may in-

clude advanced image preprocessing, noise reduction, contrast improvement, and adaptive thresholding.

4. **Ensemble Approaches:** Using ensemble methods, such as aggregating predictions from various OCR models or applying model stacking strategies, can improve the OCR system's capacity to handle answers from numerous classes. This strategy makes use of the advantages of many models and can raise overall recognition accuracy.

■

Appendix A

Computer Programs/Codes

A1 Modules

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
import cv2
from skimage.feature import hog
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from keras.layers import Conv2D, BatchNormalization, MaxPool2D, GlobalMaxPool
```

A2 Image Representation

```
char_names = df.character.unique()
rows =10;columns=5;
fig, ax = plt.subplots(rows,columns, figsize=(8,16))
for row in range(rows):
    for col in range(columns):
        ax[row,col].set_axis_off()
        if columns*row+col < len(char_names):
            x = df[df.character==char_names[columns*row+col]].iloc[0,:-1].values.res
            x = x.astype("float64")
            x/=255
            ax[row,col].imshow(x, cmap="binary")
            ax[row,col].set_title(char_names[columns*row+col].split("_")[-1])
plt.subplots_adjust(wspace=1, hspace=1)
len(char_names)
```

A3 Different style of witing

```
X = df.iloc[:,:-1]
Y_d = df.iloc[:,-1]
num_pixels = X.shape[1]
num_classes = 46
img_width = 32
img_height = 32
img_depth = 1
X_images = X.values.reshape(X.shape[0], img_width, img_height)
for i in range(1, 9):
```

```
    plt.subplot(240+i)
    #plt.axis('off')
    plt.imshow(X_images[i-1], cmap=plt.get_cmap('gray'))
plt.show()
Y_d.head()
import sys
sys.getsizeof(X_images)
X.head()
```

A4 Data Preprocessing and Feature Extraction

A4.1 Deskewing

```
def deskew(img):
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    m = cv2.moments(gray)
    if abs(m['mu02']) < 1e-2:
       return gray.copy()
    skew = m['mu11'] / m['mu02']
    M = np.float32([[1, skew, -0.5 * img.shape[0] * skew], [0, 1, 0]])
    deskewed = cv2.warpAffine(img, M, (img.shape[1], img.shape[0]), flags=cv2
    return deskewed
```

A4.2 Implementation of the deskew function

```
input_dir= 'C:/Users/jagdish/PROJECT/Images/digit_9'
output_dir = 'C:/Users/jagdish/PROJECT/output'
```

```
for filename in os.listdir(input_dir):
    if filename.endswith('.jpg') or filename.endswith('.png'):
                img = cv2.imread(os.path.join(input_dir, filename))
        deskewed = deskew(img)
       output_filename = os.path.join(output_dir, filename)
        cv2.imwrite(output_filename, deskewed)
        cv2.imshow('Output Image',deskewed)
        cv2.waitKey(0)
        cv2.destroyAllWindows()
```

A4.3 HOG

```
winSize = (20,20)
blockSize = (10,10)
blockStride = (5,5)
cellSize = (10,10)
nbins = 9
derivAperture = 1
winSigma = -1.
histogramNormType = 0
L2HysThreshold = 0.2
gammaCorrection = 1
nlevels = 64
hog = cv2.HOGDescriptor(winSize,blockSize,blockStride,
cellSize,nbins,derivAperture,
winSigma,histogramNormType,L2HysThreshold,
gammaCorrection,nlevels)
```

A4.4 Descriptor Matrix

```
descriptor_list = []
for i, filename in enumerate(os.listdir(input_dir)):
    if filename.endswith('.jpg') or filename.endswith('.png'):
        img = cv2.imread(os.path.join(input_dir, filename))
        descriptor = hog(img, channel_axis=-1)
        rounded_matrix = np.round(descriptor, decimals=2)
        descriptor_list.append(rounded_matrix)
print(f"Descriptor matrix for image {i}: length = {len(rounded_matrix)}, cont
 descriptor_array = np.array(descriptor_list)

df = pd.DataFrame(descriptor_array)
df.to_excel('ocr.xlsx', index=False, header=False)
```

A5 Classifier Models

A5.1 kNN

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
X_train, X_test, y_train, y_test = train_test_split(X, Y_d, test_size=0.3, ra
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
```

```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

A5.2 Random Forest

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

X_train, X_test, y_train, y_test = train_test_split(X, Y_d, test_size=0.3, random_st
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
y_pred = rf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

A5.3 SVM

```
from sklearn.svm import SVC
X_train, X_test, y_train, y_test = train_test_split(X,Y_d, test_size=0.2, random_sta
svm = SVC(kernel='linear')
svm.fit(X_train, y_train)
y_pred = svm.predict(X_test)
# Evaluate accuracy of model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

A5.4 CNN

```
    x_train, x_test, y_train, y_test = train_test_split(X,Y_d, test_size=0.2, random
```

```
x_train, x_test = x_train.values.reshape(x_train.values.shape[0], 18, 18), x_
print(f'Shape of Training Image array: {x_train.shape}')
print(f'Shape of Testing Image array: {x_test.shape}')
from sklearn.preprocessing import LabelEncoder
labelencoder = LabelEncoder()
Y_train = labelencoder.fit_transform(y_train)
X_train = x_train.reshape(x_train.shape[0], x_train.shape[1], x_train.shape[2
Y_test = labelencoder.fit_transform(y_test)
X_test = x_test.reshape(x_test.shape[0], x_test.shape[1], x_test.shape[2], 1)
try:
    tpu = tf.distribute.cluster_resolver.TPUClusterResolver() # TPU detection
except ValueError: # If TPU not found
    tpu = None
if tpu:
    tf.config.experimental_connect_to_cluster(tpu)
    tf.tpu.experimental.initialize_tpu_system(tpu)
    strategy = tf.distribute.experimental.TPUStrategy(tpu)
    print('Running on TPU ', tpu.cluster_spec().as_dict()['worker'])
else:
    strategy = tf.distribute.get_strategy() # Default strategy that works on
    print('Running on CPU instead')

print("Number of accelerators: ", strategy.num_replicas_in_sync)
if tpu:
    tf.config.experimental_connect_to_cluster(tpu)
    tf.tpu.experimental.initialize_tpu_system(tpu)
    strategy = tf.distribute.experimental.TPUStrategy(tpu)
    print('Running on TPU ', tpu.cluster_spec().as_dict()['worker'])
else:
    strategy = tf.distribute.get_strategy() # Default strategy that works on
```

```
    print('Running on CPU instead')
print("Number of accelerators: ", strategy.num_replicas_in_sync)
from tensorflow.keras import datasets, layers, models
momentum = .9
cnn_model = models.Sequential()
cnn_model.add(Conv2D(64, (3,3), input_shape=(18, 18, 1),
    padding='same', activation='relu'))
cnn_model.add(Conv2D(64, (3,3), padding='same', activation='relu'))
cnn_model.add(BatchNormalization(momentum=momentum))
cnn_model.add(MaxPool2D())
cnn_model.add(Conv2D(128, (3,3), padding='same', activation='relu'))
cnn_model.add(Conv2D(128, (3,3), padding='same', activation='relu'))
cnn_model.add(BatchNormalization(momentum=momentum))
cnn_model.add(MaxPool2D())
cnn_model.add(Conv2D(256, (3,3), padding='same', activation='relu'))
cnn_model.add(Conv2D(256, (3,3), padding='same', activation='relu'))
cnn_model.add(BatchNormalization(momentum=momentum))
cnn_model.add(MaxPool2D())
cnn_model.add(Conv2D(512, (3,3), padding='same', activation='relu'))
cnn_model.add(Conv2D(512, (3,3), padding='same', activation='relu'))
cnn_model.add(BatchNormalization(momentum=momentum))
# flatten...
cnn_model.add(GlobalMaxPool2D())

cnn_model.compile(optimizer='adam',
          loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
          metrics=['accuracy'])
# Display the architecture
cnn_model.summary()
cnn_model.fit(X_train, Y_train, epochs=5, validation_split=0.2, batch_size=1000)
```

A6 Classification Reports

```
    from sklearn.metrics import classification_report, confusion_matrix
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

A7 Confusion Matrix

```
    conf_mat = confusion_matrix(y_test, y_pred)
plt.imshow(conf_mat, interpolation='nearest', cmap=plt.cm.Oranges)
plt.title('Confusion Matrix')
plt.colorbar()
plt.xlabel('Predicted Class')
plt.ylabel('True Class')
plt.xticks(np.arange(conf_mat.shape[1]))
plt.yticks(np.arange(conf_mat.shape[0]))
plt.show()
```

Bibliography

[1] Meduri Avadesh and Navneet Goyal. Optical character recognition for sanskrit using convolution neural networks. In *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*, pages 447–452. IEEE, 2018.

[2] Madhuri Yadav, Ravindra Kumar Purwar, and Mamta Mittal. Handwritten hindi character recognition: a review. *IET Image Processing*, 12(11):1919–1933, 2018.