TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

A Major Project Report

On

Digitization of Devanagari Handwritten Text

Submitted By:

Bishwash Gurung [40660]

Manish Thapa [40669]

Ram Shrestha [40678]

Regan Sthapit [40679]

Submitted To:

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

March, 2023

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

A Major Project Report
On
Digitization of Devanagari Handwritten Text

Submitted By:

Bishwash Gurung [40660]
Manish Thapa [40669]
Ram Shrestha [40678]
Regan Sthapit [40679]

Submitted To:

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

In partial fulfillment for the award of the Bachelor's Degree in Electronics and Communication Engineering.

Under the Supervision of

Er. Praches Acharya

March, 2023

DECLARATION

We hereby declare that the report of the project entitled **"Digitization of Devanagari Handwritten Text"** which is being submitted to the **Department of Electronics and Computer Engineering, IOE, Thapathali Campus**, in the partial fulfillment of the requirements for the award of the Degree of Bachelor of Engineering in **Electronics and Communication Engineering** is a bonafide report of the work carried out by us. The materials contained in this report have not been submitted to any University or Institution for the award of any degree and we are the only author of this complete work and no sources other than the listed here have been used in this work.

Bishwash Gurung (Class Roll No: THA075BEI012) _____

Manish Thapa (Class Roll No: THA075BEI021) _____

Ram Shrestha (Class Roll No: THA075BEI030) _____

Regan Sthapit (Class Roll No: THA075BEI031) _____

Date: March, 2023

CERTIFICATE OF APPROVAL

The undersigned certify that they have read and recommended to the **Department of Electronics and Computer Engineering, IOE, Thapathali Campus,** a major project work entitled **"Digitization of Devanagari Handwritten Text"** submitted by Bishwash Gurung, Manish Thapa, Ram Shrestha, and Regan Sthapit in partial fulfillment for the award of Bachelor's Degree in Electronics and Communication Engineering. The project was carried out under special supervision and within the time frame prescribed by the syllabus.

We found the students to be hardworking, skilled, and ready to undertake any related work to their field of study and hence we recommend the award of partial fulfillment of Bachelor's degree of Electronics and Communication Engineering.

_____

Project Supervisor
Mr. Praches Acharya
Department of Electronics and Computer Engineering, Thapathali Campus

_____

External Examiner
Er. Shristi Heuju
Telecom Engineer, Nepal Doorsanchar Company Limited

_____

Project Coordinator
Mr. Umesh Kanta Ghimire
Department of Electronics and Computer Engineering, Thapathali Campus

_____

Mr. Kiran Chandra Dahal
Head of the Department,
Department of Electronics and Computer Engineering, Thapathali Campus

March, 2023

COPYRIGHT

ACKNOWLEDGEMENT

We wish to express our heartfelt appreciation to everyone who helped us with our major project in our final year. Firstly, we wish to convey our appreciation to Thapathali Campus for giving us the chance to pursue our academic goals and providing us with access to the materials and facilities we needed to finish our project. Our sincere gratitude goes out to Er. Praches Acharya for his important leadership and assistance throughout the project. His knowledge and input were crucial in determining the course of our project, and we are appreciative of the time and effort he invested in assisting us in reaching our objectives.

We would also like to extend our thanks to the instructors who have given us a solid foundation in our field of study and motivated us to work harder. We are grateful for the information and skills they have taught us, and their enthusiasm and commitment to teaching have been inspiring. Lastly, we want to express our heartfelt gratitude to our friends and family for their constant support and encouragement throughout our academic career. Their presence in our lives and the love and support they have given us have been invaluable. We feel fortunate to have had such a strong network of support, and we look forward to continuing to make them proud in our future endeavors.

Bishwash Gurung (Class Roll No: THA075BEI012)

Manish Thapa (Class Roll No: THA075BEI021)

Ram Shrestha (Class Roll No: THA075BEI030)

Regan Sthapit (Class Roll No: THA075BEI031)

ABSTRACT

Handwriting recognition has attracted a lot of attention in the field of pattern recognition and machine learning. Handwritten Text Recognition (HTR) is a framework that can translate the snap shots from manually handwritten or published shape to machine-editable shape. Nepali language is written in Devanagari script from left to right fashion having a straight line on top. We preprocess the images, extract features, and train a Convolutional Neural Network (CNN) model using a large dataset of handwritten Devanagari characters. We evaluate the performance of our model using various metrics such as accuracy, precision, and recall. Our results show that our model can achieve high accuracy in recognizing handwritten Devanagari characters. The development of a reliable Devanagari HTR system has been a challenging task due to the complex nature of the Devanagari script and the variability in handwriting styles. However, recent advancements in deep learning techniques have shown promising results in improving the accuracy of Devanagari HTR systems.

In this project, we use a large dataset of handwritten Devanagari characters to train our CNN model. We preprocess the images by normalizing and resizing them to a standard size to reduce the effect of variability in handwriting styles. We extract features using various convolutional and pooling layers in our model, which helps in capturing important patterns and features of the characters.

Overall, our project contributes to the development of Devanagari HTR and has the potential to make various industries more efficient, accurate, and accessible. The proposed system can be further improved by using more advanced deep learning techniques such as Recurrent Neural Networks (RNNs) and attention mechanisms to better capture the temporal and spatial dependencies of the characters.

Keywords: AI, Handwriting Recognition, Machine learning, Nepali Handwritten Datasets, Neural Network

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
CNN	Convolution Neural Network
DHCD	Devnagari Handwritten Character Dataset
DHTR	Devnagari Handwritten Text Recognition
IT	Information Technology
ML	Machine Learning
MNIST	Modified National Institute of Standards and Technology
NN	Neural Network
OCR	Optical Character Recognition
PR	Precision Recall
RMSP	Root Mean Square Propagation
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristics

1. INTRODUCTION

1.1 Optical Character Recognition (OCR)

Several digital writing tools are available nowadays, but there is still a presence of traditional writing using pens and paper. There are several such traditional documents which need to be typed manually in order to store and process them in digital format.

Some institutions still employ such traditional data storage methods. This is caused by the lack of technical expertise required to operate a computer. Also, the fear of security relating to digital data is a concern, especially without an IT department dedicated to them. The consequence is a time-consuming search for documents when they are required and the possibility of loss of the documents.

OCR is a branch of computer science concerned with transforming text from pictures of printed and handwritten papers into text which can be read by a computer. OCR converts text in picture into textual information, allowing for editing, searching, and republishing without having to retype the entire manuscript. Any printed or handwritten document that is to be digitally recreated must be scanned. A cloned document cannot be changed in terms of spelling, words, font style, or font size. It is also highly time-consuming to type a complete document to copy it. An OCR system is required to address the above-mentioned difficulties.

The technique of optical character recognition (OCR) recognizes text characters in digital photographs or scanned documents and turns them into editable and searchable text. OCR is utilized in many applications, including document scanning, indexing, and digitizing old books and manuscripts. The procedure entails evaluating the text picture, detecting individual letters, and transforming them into machine-encoded text. With the application of deep learning algorithms, OCR technology has advanced dramatically in recent years and has become a crucial tool for automating different business operations.

1.1.1 General Architecture of OCR

The block diagram for general architecture of OCR is shown below:

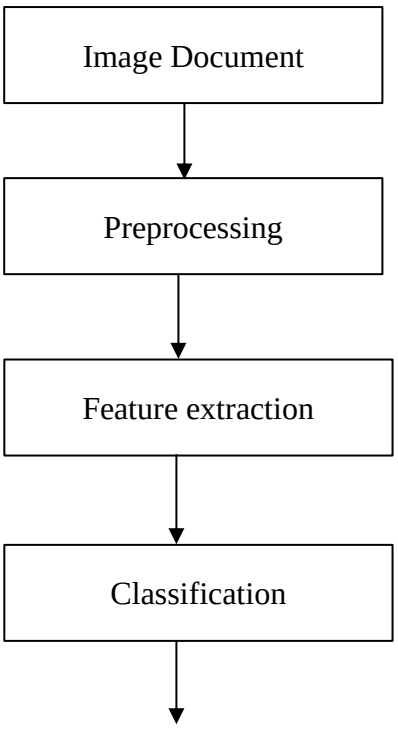


Figure 1- 1: General architecture of OCR

Preprocessing

The graphics file is imported into the OCR program, which then preprocesses it to increase recognition. Various tasks are carried out during pre-processing, including:

- De-skewing
- Binarization
- Page Layout Analysis
- Text lines and words detection
- Character segmentation
- Normalization

Feature Extraction

This component extracts features from an input picture and saves them in a feature vector.

Classification

An Artificial Neural Network (ANN) classifies the test image's segmented characters.

Post-Processing

This subsystem maps the results of classification to current form.

1.1.2 Uses and Limitations of OCR

OCR is commonly used to recognize and search text in electronic records, in addition to posting content on a website. It has made it possible for scanned material to expand beyond simply picture files, transforming them into making documents fully searchable with computer-readable text content. A variety of applications have made use of optical character recognition. Among them are the following:

- Digital Libraries and Institutional Repositories
- Banking: form completion, check collecting, and so on.
- Healthcare: Processing of basic paperwork, insurance documentation, and prescription paperwork
- Handwriting Recognition
- Automatic Identification of License Plates

The following are some examples of OCR's benefits:

- Less expensive than hiring someone to independently enter large amounts of text.
- Much quicker than hiring someone to independently input large amounts of text.

Some of the constraints of OCR are as follows:

- Limited Documents: It does not work well with documents that contain both photos and text, tables, noise, or dirt.
- Correctness: The accuracy of a document is determined by its quality and type, especially the typeface used. Misreading letters, skipping over illegible characters, and combining text from neighboring columns or picture captions are all examples of OCR errors.
- Additional Work: OCR is not mistake proof; it makes mistakes as well. The original image must be carefully compared.

1.2 Devanagari Script

The Devanagari script is multilingual and is used by over 5 million people around the world. The most common languages represented in Devanagari Script are Sanskrit, Nepali, Hindi, and

Marathi. Nepali and Hindi are the official languages of Nepal and India respectively. The term "Devanagari" ("Deva" + "Nagari") comes from Sanskrit. "Deva" denotes divinity, while "Nagari" signifies a city dweller. Almost all Hindu scriptures were written in Devanagari script, with the implication that Sanskrit was the language of the gods. The Devanagari Character Recognition system is a challenging and complex topic of research. Due to the extreme complexity of the segmentation and identification method, development in automatic recognition of the Devanagari script has been extremely sluggish.

The Nepali language is written in Devanagari script. Devanagari is written from left to right along the horizontal line. A single vertical line called (Purnaviram) indicates the end of phrase and sentence in Devanagari. Devanagari script contains a horizontal line, which connects all the characters of a word. This line is called 'Shiro Rekha' or headerline. Based on these horizontal lines, Devanagari scripts can be divided into three zones: top zone which is above the header line ,the core/middle zone just below the headerline and bottom zone below the baseline of core zone. The core and top zone are separated by the header line. The header line also helps to separate the different words from each other.

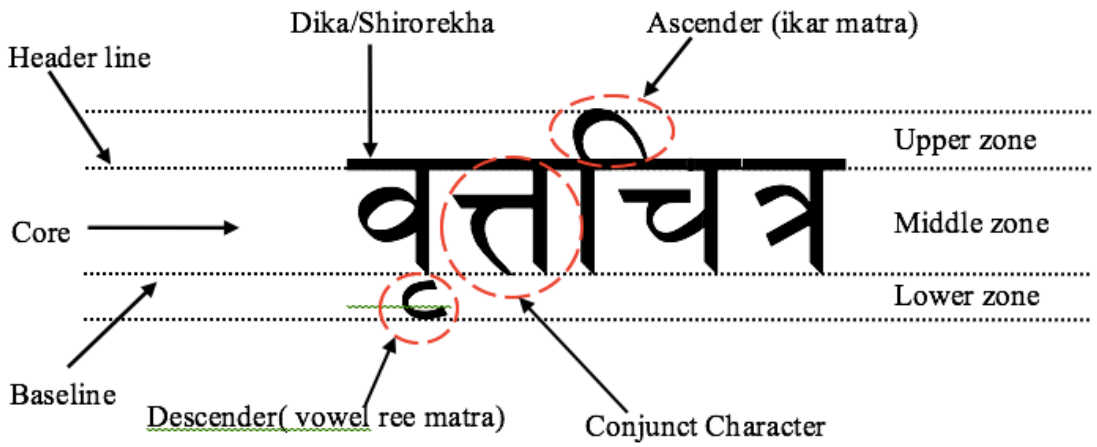


Figure 1- 2: Structure of Devanagari word

The symbols which are written above or below the core zone are considered as upper modifier and lower modifier respectively. A conjunct character is a composite character made up of one or more half consonants, followed by a consonant. In each conjunct character, the right part is

a full consonant, and the left part is always a half consonant.

This project aims to digitize Devanagari handwritten text. Devanagari script contains 36 consonant (vyanjanvarna) characters, 13 vowel (swarvarna) characters and 10 numbers which is shown in Table 1- 1, Table 1- 2 and Table 1-3 respectively. Devanagari script contains different types of upper and lower modifiers which is shown in Table 1- 4. Also the presence of 'Shiro Rekha', 'Aakar','Ekar','Ukar','Rasho','Dirgha','Chandrabindu' make it challenging for pattern recognition.

Table 1- 1: Nepali Consonant Characters

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

Table 1- 2: Nepali Vowel Characters

अ	आ	इ	ई	उ	ऊ	ऋ	ए	ऐ	ओ	औ	अं	अः

Table 1- 3: Nepali Numeric Characters

०	१	२	३	४	५	६	७	८	९

Table 1- 4: Modifiers

ा	ि	ी	ु	ू	े	ै	ो	ौ	ृ	ं	ँ	ः

Table 1- 5: Consonants and their half forms

Consonant	Half form	Consonant	Half form	Consonant	Half form	Consonant	Half form	Consonant	Half form
क	क्	ख	ख्	ग	ग्	घ	घ्	ङ	
च	च्	छ		ज	ज्	झ	झ्	ञ	ञ्
ट		ठ		ड		ढ		ण	ण्
त	त्	थ	थ्	द		ध	ध्	न	न्
प	प्	फ	फ्	ब	ब्	भ	भ्	म	म्
य		र		ल	ल्	व	व्	श	श्
ष	ष्	स	स्	ह	ह्				

In Devanagari script, we can find different letter variants in printed or handwritten documents. They are particularly used in older text. While certain older types of letters, including letter अ and letter ण are no longer utilized, they are still present in many older manuscripts. Several letter variations are seen in Table 1- 6.

Table 1- 6: Letter Variations

standard	ancient
अ	ॳ
झ	झ
ण	ण
ल	ल

1.3 Machine Learning

Machine learning (ML) is the scientific study of the algorithms and statistical models that

computer systems use to do a certain task efficiently without explicit instructions, instead relying on patterns and inference. It is considered as a subset of artificial intelligence. Machine learning algorithms construct a mathematical model using sample data, referred to as "training data" to make decisions without being expressly programmed to do so. ML algorithms are utilized in a broad range of applications, including email filtering and computer vision, when developing an algorithm containing particular instructions for accomplishing the task is impractical. ML is connected to computational statistics, which is concerned with making predictions using computers.

Some common terms used in ML contexts are:

- **Supervised Learning:** This algorithm creates a mathematical model from a set of data that includes both inputs and expected outputs. If the aim was to determine whether a picture contained a specific object, the training data for a supervised learning system would comprise images with and without the object (the input), and each image would have a label (the output) indicating whether it had the object.

- **Unsupervised Learning:** The technique constructs a mathematical model using a collection of data that just comprises inputs and no desired output labels. Unsupervised learning techniques are used to discover structure in data, such as data point grouping or clustering. Unsupervised learning, like feature learning, may detect patterns in data and categorize inputs. The practice of lowering the number of "features," or inputs, in a piece of data is known as dimensionality reduction.

- **Optimization:** Optimization is the most important element in the machine learning algorithm. It begins with the definition of some form of loss function/cost function and finishes with the loss being minimized using one or more optimization routines. The selection of the optimization algorithm might be the difference between high accuracy in hours or days. Optimization has several applications and is a highly explored issue in business and academia. SGD (Stochastic Gradient Descent), ADAM (Adaptive Momentum), Gradient Descent, and so on are some examples.

- **Loss:** Loss is a non-negative variable that enhances the reliability of the model as the value of the loss function decreases. MSE (Mean Squared Error), Categorical Cross Entropy, and

other loss functions are used.

- **Underfitting and Overfitting:** Underfitting occurs when there are few train datasets and the loss for the test set is large. Overfitting occurs when there are few datasets and the model is trained for a long time, and the model finally recalls the dataset. Both of these issues result in significant loss and decreased precision. These issues may be solved by employing generalization techniques, an appropriate loss function, train data, and a high learning rate.

1.4 Motivation

Digital papers have ingrained themselves into daily life. Anyone may benefit from scanning their papers to make them easier to reference, organize files, protect, and save. Document types that can be digitized are unlimited. Thus, the heightened curiosity urges us to cope with the Nepali handwritten text.

In the case of Nepali OCR, modifiers, compound characters, and touching characters, and variations in typefaces prevent the segmentation process from achieving complete accuracy. Successful recognition is directly impacted by these issues, which degrade performance. Therefore, it is indeed to focus on a better segmentation strategy and performance enhancement.

1.5 Problem Definition

All of us desire an OCR system for Nepali that can detect various document formats and papers made up of diverse typefaces, but the primary thing we want is accuracy in the recognition. The OCR projects for Sanskrit, Hindi and Nepali have all just been released. Their performance, though, has not been up to par.

Character segmentation is one of the main issues while trying to recognize Nepali characters. This issue develops as a result of modifiers, merged characters, variable character sizes, and other factors. The conjuncts and compound characters are not handled properly, which is the issue. The other causes for difficulties in recognizing Devanagari characters are:

- Some Devanagari characters have similar shapes.
- Varying writers, or even the same writer, might write at different times depending on the pen or pencil used.
- The character may be written in several locations on paper. Characters can be written in a

variety of typefaces.

This problem needs to be solved to create an OCR for Nepali language that performs well.

1.6 Project Objectives

The main purpose of our project is listed :

- To digitize Devanagari handwritten text using CNN.

1.7 Project Scope and Applications

The technique of identifying handwritten characters in the Devanagari script, which is used to write various Indian languages including Hindi, Sanskrit, Marathi, Nepali, and others, is known as Devanagari Handwritten Text Recognition (HTR). Devanagari HTR's uses span a wide range of fields, including banking, government, education, and the media. The following are some uses for Devanagari HTR:

Banking and finance: Handwritten characters and numbers on checks, bank forms, and other financial documents may be read using Devanagari HTR. This can assist banks and other financial organizations in automating document processing while increasing accuracy and minimizing manual labor.

Government and administration: The digitalization of government records and archives is possible using Devanagari HTR. This can enhance accessibility, preserve historical records, and make it simpler to search for and retrieve information.

Education: Handwritten notes and documents may be recognized using Devanagari HTR, which is useful when digitizing textbooks, academic papers, and other educational resources. This may contribute to making education more inexpensive and accessible, particularly in rural regions.

Media and publishing: Handwritten manuscripts, books, and articles may be digitally translated using Devanagari HTR. This can enhance readability, preserve literary works, and make it simpler to disseminate and exchange material.

In conclusion, Devanagari HTR has a wide range of uses and may change many sectors by increasing their accessibility and efficiency.

1.8 Project Organization

The report is divided into 8 chapters. Each chapter discusses different sections of the project. Chapter 1 is the introduction chapter, which gives the basic introduction to Digitization of Devanagari handwritten text, along with its scope and applications. Chapter 2 contains the existing and previous works related to the field of Digitization of Devanagari handwritten text with references from various journals, articles, and research papers. Chapter 3 explains the requirement in our project and their usage. The methodology and working principle of the system is described in Chapter 4 along with the system block diagram. Chapter 5 gives a clear insight into the functioning and implementation details of our project. In Chapter 6, the obtained result is analyzed clearly. Chapter 7 explains further works to be carried out on the project, for further future enhancements. Finally, conclusions were drawn from the project which is covered in Chapter 8.

2. LITERATURE REVIEW

2.1 Segmentation

Character segmentation is a process that aims to break down a series of characters' pictures into smaller representations of individual symbols. The type and quality of the content to be read affect how challenging it is to segment it accurately.

For optical character recognition, character level segmentation of printed or handwritten text is a crucial preprocessing step (OCR). It has been observed that languages with a cursive writing style make the segmentation issue significantly more challenging. Handling the inherent variation in writing styles of many people presents the fundamental problem in handwritten character segmentation.

Various approaches of segmentation have been employed on segmentation and classification in the OCR system. The approaches are:

1. **The conventional method**, where segments are chosen based on "character-like" characteristics “Dissection” is the name given to the process of breaking down the picture into meaningful parts.
2. **Recognition-based segmentation**, where the computer system looks for parts of the image that correspond to classes in its alphabet.
3. **Holistic approaches**, where the machine tries to identify words as a whole rather than breaking them up into characters.

Conventional Technique

Dissection is the process of breaking down a picture into a series of smaller images utilizing valid character attributes like height, breadth, distinction from surrounding components, placement along a baseline, etc. Dissection is a clever procedure since it analyzes the image; yet, at this moment, there is no symbol categorization going on.

Three phases made up the segmentation stage:

1. Recognizing the beginning of a character.
2. The choice to start character end testing

3. End-of-character detection

The segmentation of non-cursive writing has been based on an investigation of a line of print's projection. The projection frequently has a minimum at the appropriate segmentation column when printed characters contact or overlap horizontally. This approach has been enhanced using a peak-to-valley function. The projected value is indicated together with the location of the minimum. The slanted letters are less satisfactorily projected vertically. An effective method for separating non-touching characters in data that has been manually or mechanically printed is to analyze projections or bounding boxes. To consistently separate connected characters, nevertheless, further in-depth analysis is required. Two characters can overlap to create unique picture characteristics. Dissection techniques have been created as a result to find these characteristics further to employing them to divide a character string picture into smaller ones. Only picture components that fail specific dimensional checks are examined in depth.
Challenges faced in this technique are:

1. To perform segmentation in Devanagari is complicated.
2. Recombination of individual recognized character should be done.

Recognition Based Segmentation Technique

This method breaks down words into their component letters. It differs significantly from a dissection-based method. In this instance, no dissection based on features approach is used. Instead, the graphics is deliberately split up into several overlapping sections without respect for the image's substance. These are categorized in an effort to get a segmentation and recognition result that is cohesive. Letter segmentation's consequence is letter recognition, which may also be impacted by contextual analysis. The primary benefit of this class of approaches is that they do not address the segmentation issue: Recognition problems are mostly the result of incorrect categorization; therefore, no complicated "dissection" technique has to be created.

Kompalli, Suryaprakash et al. [1] described a method which used graph representation to segment individual characters. With the help of this technique, we divide characters that overlap horizontally or vertically, those linked with non-linear borders, and others into smaller, more primal components.

Holistic Technique

The holistic approach contrasts with the traditional dissecting method. This method is employed to identify words as a whole. Consequently, the characterization of words is skipped. The characteristics of the unsegmented word picture are compared to the features or descriptions of the words in the database.

The application of this class of methods is typically restricted to predetermined terms because a holistic approach does not directly deal with letters or alphabets. To increase or alter the range of potential words, a training phase is thus necessary. This characteristic makes this sort of technique more appropriate for uses like check recognition when the vocabulary is statically specified. They can be applied to a specific user as well as the relevant terminology.

B. Shaw, S. K. Parui, et al. [2] described a holistic method for Devanagari recognition using hidden Markov model (HMM). As the feature vector, a sliding window scans the image-strip histogram of chain-code directions from left to right. The recognition of a word picture is suggested using an HMM with constant density. Each word's HMM is built, the conditional probability for each HMM is calculated and the class that has the highest probability is finally selected.

Challenges faced in this technique are:

1. Huge datasets are required to get a reasonable accuracy
2. Number of classes are extravagant
3. Unlike in English vocabulary, where 800 words are enough to hold a basic conversation whereas in Devanagari no such research has been done.

2.2 Recognition

Different techniques of recognition are used for different segmentation techniques. For instance, in Holistic method, word is recognized as whole. However, Conventional segmentation technique recognizes individual segmented character. For Devanagari optical character recognition, several feature extraction techniques and classifiers have been put forth. They have all concentrated on the enhanced performance.

Rao, N. Venkata, et al. [3] described different approaches of recognition as below:

1. **Matrix Matching :**

In matching, each character is transformed into a pattern within a matrix, and the pattern is then examined against an index of recognized characters. On uniform, single-column pages with monotype, its recognition is strongest.

2. **Fuzzy Logic :**

A multi-valued logic known as fuzzy logic enables the definition of intermediate values between traditional evaluations such as yes/no, true/false, black/white, etc. There is an effort to give computer programming a more human-like logic-thinking style. When there is uncertainty and there are no clear true or false values for the replies, fuzzy logic is utilized.

3. **Feature Extraction :**

Each character is described using the presence or absence of important characteristics such as height, width, density, loops, lines, stems, and other characteristics. The feature extraction method is ideal for laser printing, high-quality pictures, and magazine OCR.

4. **Structural Analysis:**

Characters are identified via structural analysis by looking at their sub-vertical and horizontal histograms and sub-feature shapes in the image. Its character repair feature is excellent for newspaper and low-quality text.

5. **Neural Network:**

This method samples the pixels in each image and compares them to a database of character pixel patterns in order to mimic how the human brain system functions. For fixed documents and damaged text, the capacity to distinguish the characters through abstraction is quite useful. Neural Networks are more effective than others in all of these strategies.

Pant, Nirajan. [4] has presented approach utilizing the Random Forest algorithm, which combines two methods for printed Nepali text: The Holistic methods and the Character Level Recognition method. The 78.87 percent accuracy rate was achieved for character level recognition approach and 94.80 percent accuracy rate for the hybrid approach. When 5 comparing the two techniques, he found out that hybrid approach achieved the best performance than character level recognition approach which is shown in below graph:

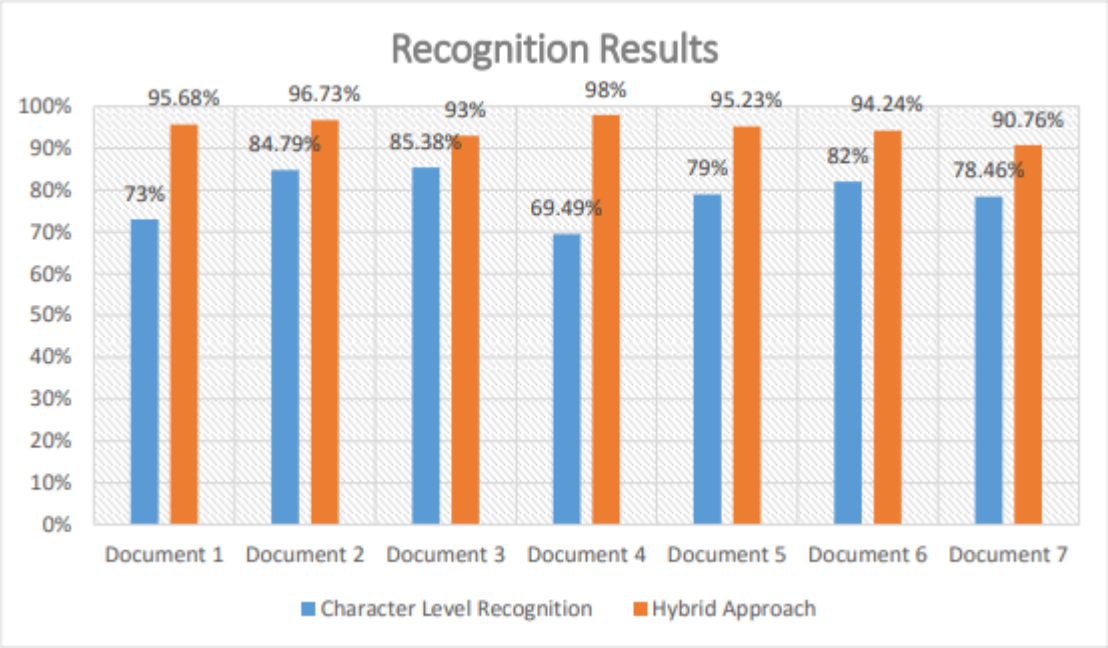


Figure 2- 1: Recognition Results *[4]*

Puri, Shalini, et al. [5] have presented an effective Devanagari character classification model using SVM. The model initially preprocesses the picture before segmenting it with projection profiles, removing Shiro Rekha, extracting characteristics, and categorizing the Shiro Rekha-less letters into predetermined character groups. The approach outperformed current methods by producing average classification accuracy of 99.54 % for printed photos and 98.35 % for handwritten images, respectively, and demonstrated higher performance than existing methods.

Pande, Dwarka Nath, Sandeep, et al. [6] have used the finest methodologies for increasing recognition rates and built a Convolutional Neural Network (CNN) to successfully recognize Devanagari handwritten text (DHTR). The system makes use of the Devanagari handwritten character dataset (DHCD), a large open dataset that includes 46 classes of Devanagari characters, each with 2,000 images. After recognition, conflict resolution is crucial and this technique gives the user a way to address conflicts. In terms of accuracy and training duration, this method yields encouraging results.

Sayyad, Jadhav et al. [7] have performed to recognize Devanagari characters using multilayer perceptron with hidden layer. Various patterns of characters are created in the matrix (n*n) with the use of binary form and stored in the file. The back propagation neural network has used for efficient recognition and rectified neuron values were transmitted by feed forward method in

the neural network.

2.3 Recombination

Python supports devanagari string processing. The **Nisaba Brahmic library** [8] supports critical Brahmic script functions like NFC, visual normalization, reversible transliteration, and validity checks inside a modular and extendable framework.

2.4 Related works

OCR for languages using the Devanagari script has previously undergone extensive investigation, with Hindi in particular claiming to have a performance accuracy rate of up to 93% at the character level. [9]

Shakya, Shanti, et al. [10] have described the general overview of the Nepali OCR system. In their paper they have discussed about the five major methods involved in their projects. They are Preprocessing, Segmentation, Feature Extraction, Training and Recognition. They claimed to have achieved about 90% and 80% accuracy rates for segmentation and recognition respectively.

The project developed by **Ghimire, Ashutosh, et al.** [11] focuses on Nepali character recognition with template matching technique in machine learning with the specific implementation of Back Propagation Algorithm and Gradient Descent Algorithm. The methods in this paper includes the collection of datasets from Computer Vision Research Group, Nepal.

Dawadi, Babu Ram, et al. [12] have demonstrated the solution for Nepali text recognition. They have explored the segmentation technique and revealed its pros and cons. They have developed the system utilizing feed forward back propagation Artificial Neural Network (ANN).

Prajapati, Sudan, et al. [13] have analyzed the efficiency of character recognition of printed text in Nepali script using Tesseract engine and Artificial Neural Network. During the training phase, the total accuracy was 96%, while during the testing phase, it was 69%. Similarly, an ANN accuracy of 98% in training and 81% in testing was attained.

Ingroj [14] provides a way to generate Unicode for Nepali characters. Nepali texts are written in Devnagari script. Characters in the Devnagari script have Unicode code points between u0900 and u097F. Nepali texts are encoded using utf-8 encoding. One of the most used character encodings is this one. A character is represented by an 8-bit block in utf-8 encoding.

S. Bag ,et al. [15] have described an effective character segmentation approach for handwritten Hindi words in this research. Segmentation is carried out based on structural patterns detected in this language's writing style. As input, the suggested approach can handle wide variances in style of writing and skewed header lines. For both handwritten and printed words, the approach has been tested on their own database. The success percentage is 96.93% on average.

Table 2- 1: Comparative study of methodologies applied on Devanagari Handwritten Text Recognition

Topic	**Methodology**	**Accuracy**
Devanagari OCR using a recognition driven segmentation framework and stochastic language models.[1]	Recognition driven segmentation using a graph	The recognition driven BAG segmentation has an accuracy range of 72 to 90%
Offline Handwritten Devanagari Word Recognition: A holistic approach based on directional chain code feature and HMM.[2]	HMM (Hidden Markov Model)	80.2%
Improving Nepali OCR performance by using hybrid recognition approaches.[4]	Character Level Recognition and Hybrid approach.	78.87% & 94.80% respectively

An efficient Devanagari Character classification in printed and handwritten document using SVM.[5]	SVM (Support Vector Machine)	99.54% for printed & 98.35% for handwritten images
A Complete OCR for printed Hindi text in Devanagari script.[9]	Collapsed Horizontal Projection Technique	93% at character level
Dictionary Based Nepali Word Recognition using Neural Network.[12]	Multi-layer Feed Forward Back Propagation Artificial Neural Network (ANN)	About 90% for simple word, 60% for complex word, and near about 50% for handwritten word
Character Segmentation of Hindi Un-constrained Handwritten Words.[15]	Character segmentation method	Average accuracy rate of 96.93 %.

2.5 OCR tools developed for Devanagari

Numerous businesses and individuals in India and Nepal have started the creation of Devanagari OCR software (Sanskrit, Hindi, Marathi, and Nepali). Chitrankan, a Hindi and Marathi OCR system has been created by C-DAC in India. Based on HTK tool and the Tesseract Open Source OCR engine, Nepalese Madan Puraskar Pustakalaya (MPP) has also created OCR initiatives for the Nepali language. Dr. Oliver Hellwing established Ind.Senz, which is creating OCR software for languages written in the Sanskrit, Hindi, and Marathi scripts. Sanskrit/Hindi-Tesseract OCR are the other initiatives.
Below is a detailed description of these tools:

Sanskrit / Hindi - Tesseract OCR (Trained data files for Devanagari fonts for Tesseract OCR 3.02+):

Hih.trained data is available in OCR Tesseract 3.02 for identifying texts written in Devanagari scripts. The lack of training manuals of texts, photos, and file box makes it challenging to increase accuracy by modifying the learned data. It has been observed that recognition is quicker and more accurate when training with a font that is similar to or the same as that used in the text that will be OCRed. This trained data is maintained by Shreeshrii with the intention of producing trained data for multiple Devanagari fonts so that Tesseract OCR may be used to recognize documents written in various Devanagari fonts**.** [16]

Google Drive OCR:

Google has made Nepali OCR available within Google Drive. Users of Google Drive can use the OCR software without charge. In papers with a single column, OCR performs well. Some formatting, including bold, line breaks, font size, and font type, may be retained. Tables, columns, footnotes, endnotes, and lists, however, are probably not going to be seen**.**

Chitrankan

C-DAC created the Hindi and other Indian language OCR (Optical Character Recognition) system. Along with embedded English text, it works with Hindi and Marathi. It has features including a spell checker, the ability to export text as. RTF for editing in any word processor, and the ability to save recognized text in ISCII format. Additionally, implemented are powerful DSP (Digital Signal Processing) techniques to reduce noise and reflection on the back page, automated text and picture region identification, and skew detection and correction up to 15°. Since the identified text needs to be manually edited, it is not very accurate.

Ind.senz OCR Programs:

Sanskrit, Marathi, and Hindi OCR applications are all accessible. These are the only Devanagari OCR applications created and accessible for business usage. If there is a need to digitize a lot of Sanskrit and Hindi text, Ind.senz discusses the usefulness of programs in universities, publishing houses, and data entry firms. The software automatically converts text pictures into computer-editable text in Unicode format. High accuracy rates were achieved using common Devanagari fonts, according to Ind.senz.

A Step Towards development of Nepali OCR:

HTK Toolkit Based OCR

Phase I of the PAN Localization project saw the development of this OCR project (2004-2007). Project management was done by Madan Puraskar Pustakalya. The Bangladesh team provided direct instruction and direction for the development of Nepali OCR. The beta version's release marked the end of the OCR project.

Tesseract Based Nepali OCR

In Phase II of the PAN Localization Project, attempts were undertaken to construct a Nepali OCR built on Tesseract under the direction of MPP and Kathmandu University (KU). Tesseract 2.04 was used to train 202 Nepali characters for this project, including several derivative characters (characters with ukar, ekar, and aikar).

In 2009, Using Tesseract, Google's Nepali OCR was created in response to the beta release of the HTK-based Nepali OCR. Following that, work on improving and developing Nepali OCR was put on hold. These tools have not been updated in a very long time. New platforms and operating system versions have been produced in the current situation. The tools created do not satisfy the needs of the latest Operating System iterations, such as Windows 8.1 and Windows 10. Additionally, OCR solutions for other operating systems like Linux and Android are required.

In order to improve OCR performance, several approaches and techniques for correctly segmenting conjuncts, compound characters, and shadow characters are explored in this chapter. Additionally, a few character recognition algorithms and feature extraction techniques were briefly explained. The majority of studies concentrate on enhancing the conjuncts/compound character segmentation procedure to increase Devanagari OCR performance. The methodology uses segmentation strategies based on recognition, collapsed horizontal projection, and projection profile algorithms.

For the effective recognition of Devanagari letters, various feature extraction techniques and classifiers are also described. Finally, a variety of Devanagari OCR tools created for Hindi, Sanskrit, Marathi, and Nepali languages are shown.

3. REQUIREMENT ANALYSIS

3.1 Hardware Requirements

The hardware requirement for our project to be honest is quite simple. As our project is a software-based project so the hardware required for our project is a computer which can handle the below-mentioned software. Therefore, a computer or a laptop with minimum of 4GB RAM and enough storage capacity is needed for our project.

3.2 Software Requirements

Python tools will be used for feature extraction. It provides the building block to create the necessary system. Pandas is a software library for data manipulation and analysis and NumPy which is used for adding support for large collection of high-level mathematical functions to operate on these arrays will be used for data analysis.

- **TensorFlow**

TensorFlow Federation is a free framework for performing machine learning and other computations on distributed data. TensorFlow was created to enable open analytics and experiment with distributed Learning, a machine learning technique in which a common worldwide model is built across numerous cooperating clients that store their training data locally. Federated learning, for example, has been utilized for training models of prediction for mobile keyboards without transmitting confidential typing information to servers.

- **NumPy**

It is a Python library that adds support for huge, multidimensional arrays and matrices, as well as a plethora of high-level mathematical methods for working with these arrays. Numeric, NumPy's progenitor, was built by Jim Hugunin including cooperation from several different people. Travis Oliphant built NumPy in 2005 by heavily modifying Numeric and combining features from the competitor Numarray. NumPy is a software package that is open-source with a large number of contributors.

- **Visual studio IDE**

A feature-rich tool called an integrated development environment (IDE) supports numerous different aspects of software development. We may edit, debug, develop and publish an app

using the Visual Studio IDE as our creative launchpad. Compilers, code completion tools, graphical developers and numerous other capabilities are included in Visual Studio in addition to the conventional editor and debugger that are offered by the majority of IDEs to improve the software development process.

- **Streamlit**

  Streamlit is a Python library that enables data scientists and machine learning engineers to effortlessly develop web applications that are interactive. Streamlit's built-in components, such as sliders, text inputs, and charts, make it simple to create interactive widgets and visualizations, and its intuitive syntax makes it easy to write and edit code. We utilized Streamlit to create a web-based interface for our data analysis project, which allowed users to engage with the data and explore various visualizations in real-time. This made it easier for us to communicate our findings and allowed us to reach a broader audience. Notably, Streamlit makes it feasible to create such web applications even if one lacks expertise in HTML, CSS, or JavaScript.

3.3 Feasibility Study

3.3.1 Economic Feasibility

Our project is being developed with the aim of minimizing installation costs for the end-user. This is achievable because our project is entirely software-based, which makes it cost-effective. In addition, there is no requirement for purchasing any software licenses to run the system, thereby making it economically feasible to operate.

3.3.2 Operational Feasibility

The system utilizes Convolutional Neural Networks (CNN) to recognize characters with a certain level of error. The system is designed to be a user-friendly Python program with a straightforward interface, which can be easily used by average users who have installed Python. Hence, we consider our project to be operationally feasible.

3.3.3 Technical Feasibility

To ensure that the system works effectively, various technical challenges had to be addressed. One significant difficulty was the lack of an appropriate dataset, which was overcome by

manually collecting character images and performing augmentations to increase the volume of training data. Our system was developed using the python language, enabling it to be used on any operating system. Consequently, the project can be deemed technically feasible.

4. SYSTEM ARCHITECTURE AND METHODOLOGY

4.1 Block Diagram

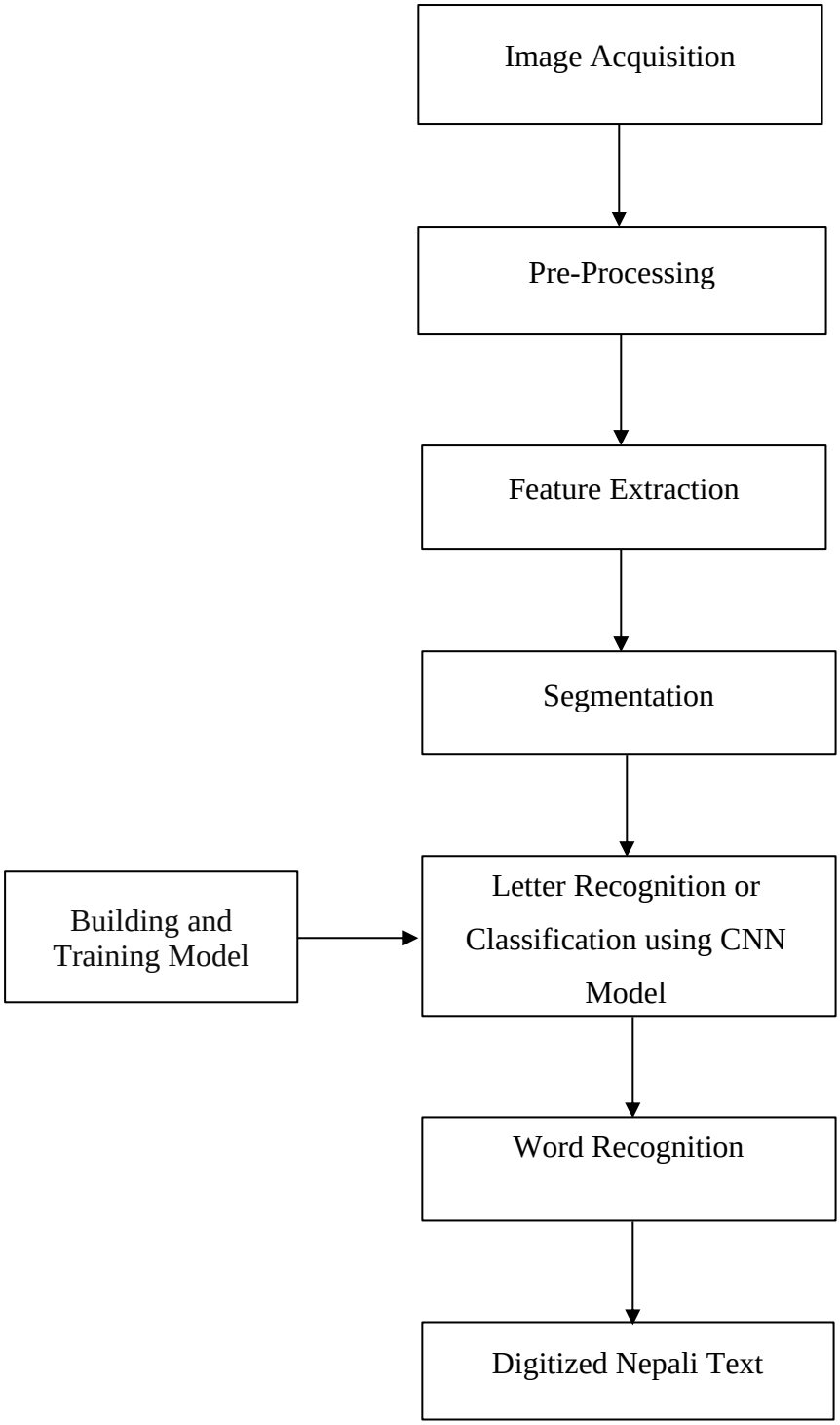


Figure 4- 1: Block Diagram for Digitization of Nepali Text

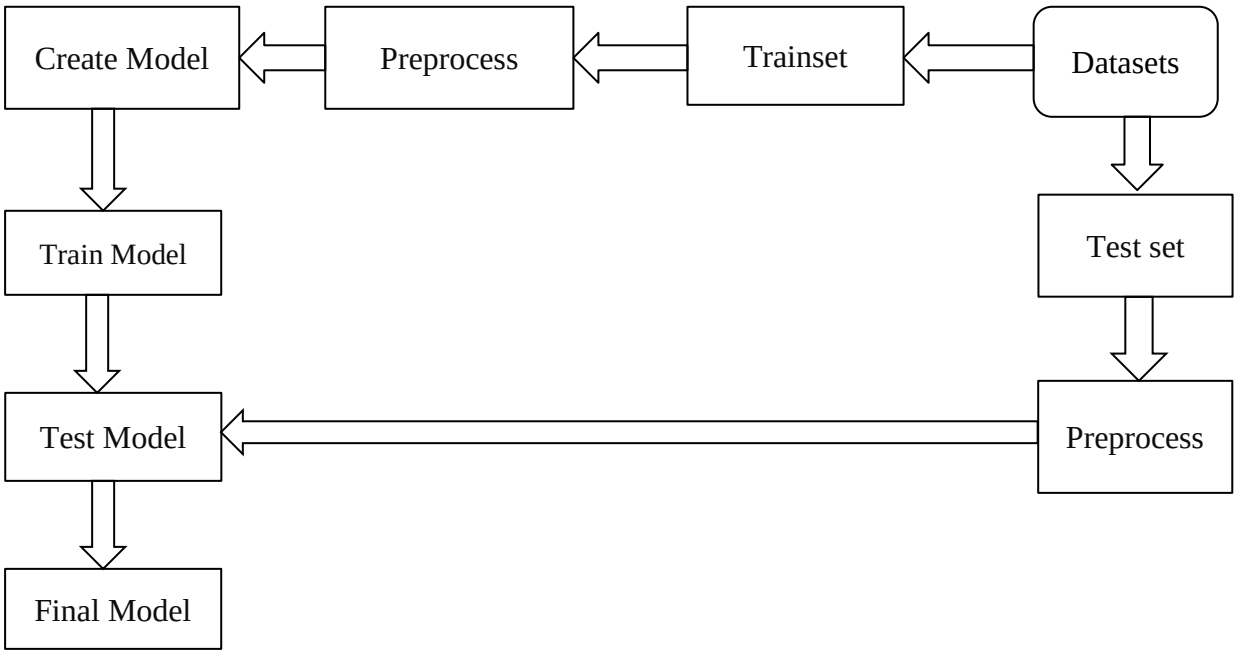


Figure 4- 2: Model Training

4.2 Dataset

On the internet, the dataset of consonant characters, vowel characters and numerals are accessible to everyone. Test and Train portions of the dataset are separated. The dataset should be transformed to the best format before being downloaded from the web. There are several sets of grayscale train and test pictures. From the total photos, 20% of which are removed to be used for validation, while the remaining 80% are from training.

Dataset used in our project contains 60 classes for model used for recognizing middle zone characters(i.e. 36 consonant characters, 6 vowels characters (अ, इ, उ, ऊ, ऋ, ए), 1 aakar (ा), 7 special characters (क्र, ट्र, त्त, द्य, प्र, श्र, रु) and 10 numerals where each class has 2000 images except numerals where each class has 1250 images. Dataset also contains 4 classes for model used for recognizing lower modifier(i.e. ु, ू , ृ , ॅ), 8 classes for model used for recognizing upper modifier(i.e. ि, ी, े, ै, ो, ौ, ँ, ं) , 19 classes for model used for recognizing half form characters and 2 classes for model used for recognizing the ending sentence (i.e. purnabiram and question mark). These all classes have 2000 images. So,

technically this dataset is larger both in terms of samples and classes than the famous MNIST dataset.

For collection of datasets for the modifiers in the Devanagari script, we had one primary approach: collecting handwritten data from school students and family members. In this approach, we requested many students and our family members, where we distributed the leaflet for writing the alphabets, and we primarily collected the handwriting of persons. Thus, obtained data was captured by the mobile phone's camera, scanned and cropped for the individual characters. The foreground of images are white color and backgrounds are black color. Each image is 32 * 32 pixel.

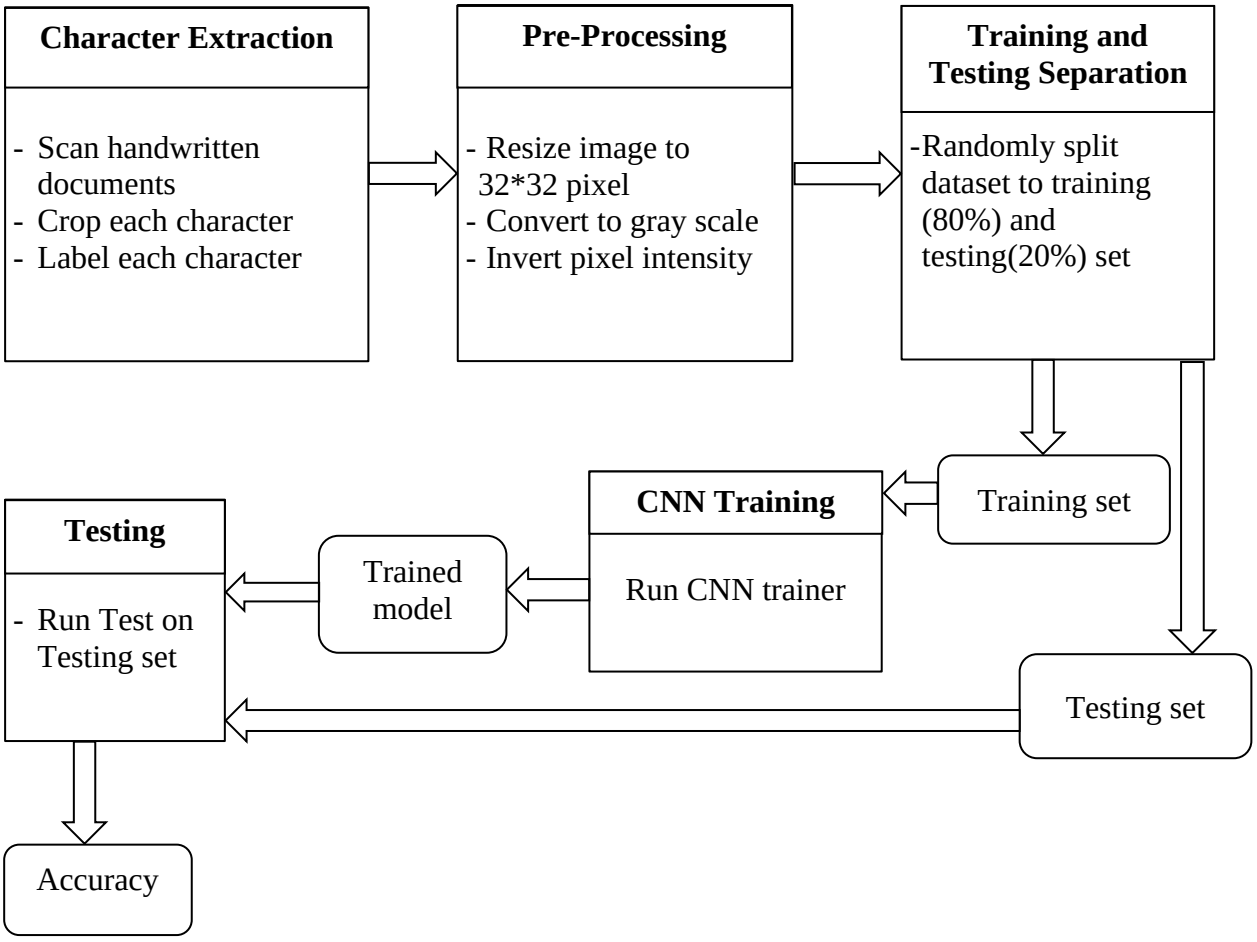


Figure 4- 3: General Flow Diagram of Dataset preparation and Character classification

4.3 Working Principle

First, we begin with the most common text recognition technique, it is quite challenging, it enables us to convert different types of documents, such as scanned paper documents, PDF files, or converting digitally recorded photos into editable, searchable data.

The text recognition system is focused on replicating natural human recognition. Integrity, purposefulness, and flexibility are the system's three primary tenets. Initially, the seen thing must always be viewed as a single object made up of several interconnected elements. In our situation, the diploma is one of these things. Second, there must always be a goal behind any interpretation of facts. And finally, the program has to be capable of self-learning.

Text extraction, sometimes referred to as keyword extraction, uses machine learning to automatically scan text and extract important terms and phrases from unstructured data, including news articles, surveys, and customer service complaints.

4.3.1 Image Acquisition

Image acquisition is accomplished by directly inputting an image file from the folder which is saved in a device .

4.3.2 Pre-processing

The document's picture will be used as an inputs query picture. First, border identification, border cropping, a page straight transformation, elimination of noise, skew adjustment, size normalization, and sharpened kernel application are employed. For preparing it for future processing, the first picture that is input is scanned and examined. The noise is eliminated after identifying the boundary of a specific page using various techniques. The picture is then transformed to a greyscale image.

4.3.3 Feature Extraction

Each character has several characteristics that are crucial for recognizing patterns. Characters from Devanagari scripts have numerous distinctive traits. In order to make the process of categorizing the pattern simple, feature extraction defines the pertinent shape information that is included in a pattern. High dimensional data visualization for cluster analysis and improved

comprehension of data structures is made possible by feature extraction for exploratory data projection. High discriminative reduced dimensionality features should be extracted for feature extraction in classification since they demand less processing power. By transforming the pre-processed picture to a bit mapped version, the vector of features is computed. The bit maps version keeps the main elements of the input image while taking up less space/data length. As a result, the time required for NN Training is reduced without compromising.

4.3.4 Segmentation

The picture is segmented into words as the next significant step following feature extraction. For segmentation, an open-source library called Tesseract-OCR is used, which produces a stream of text that is transmitted on to the next stage. The character segmentation is carried out in the next stage by splitting the input into three zones: the top, middle, and lower zones. The actual word is represented by the center zone. This step aids in determining whether or not words have been connected.

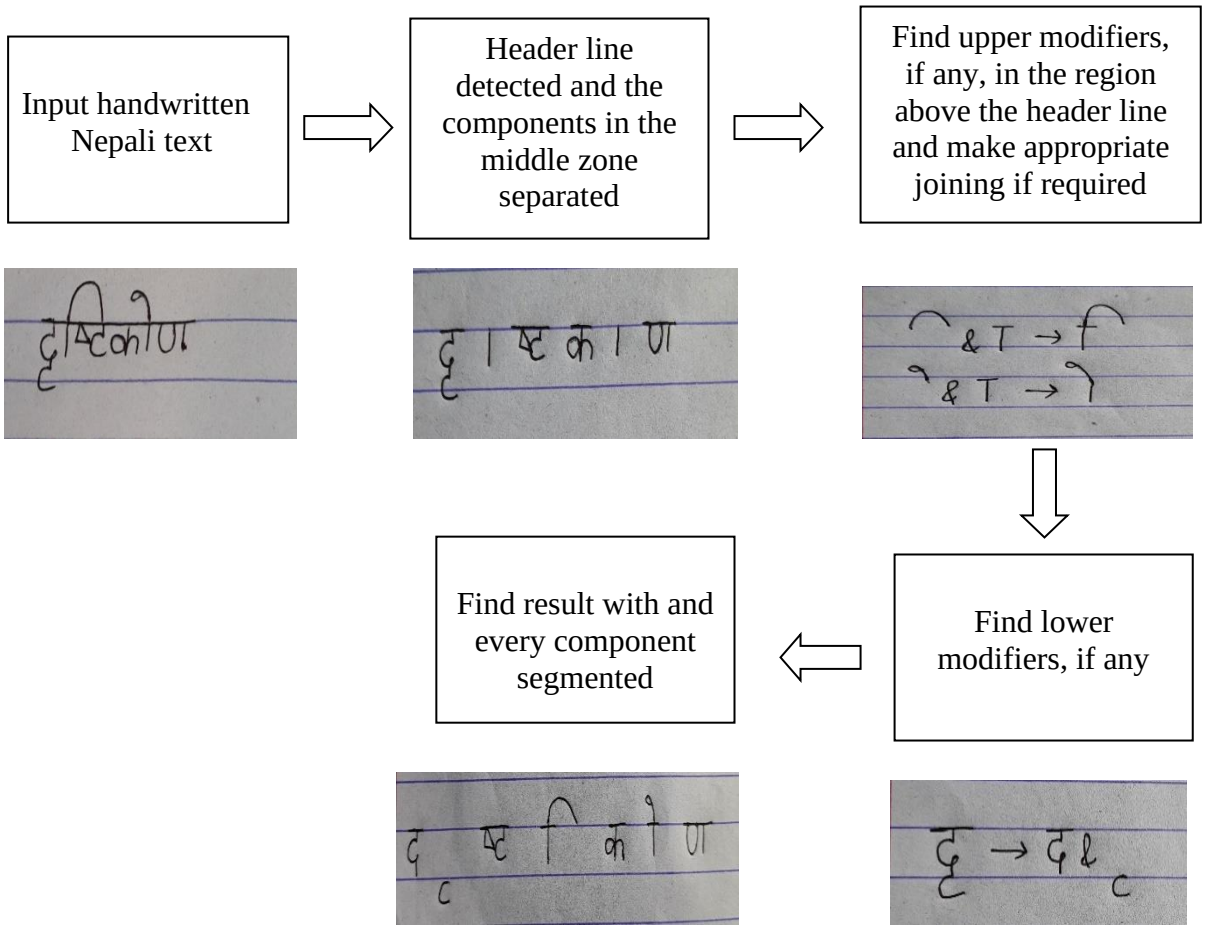
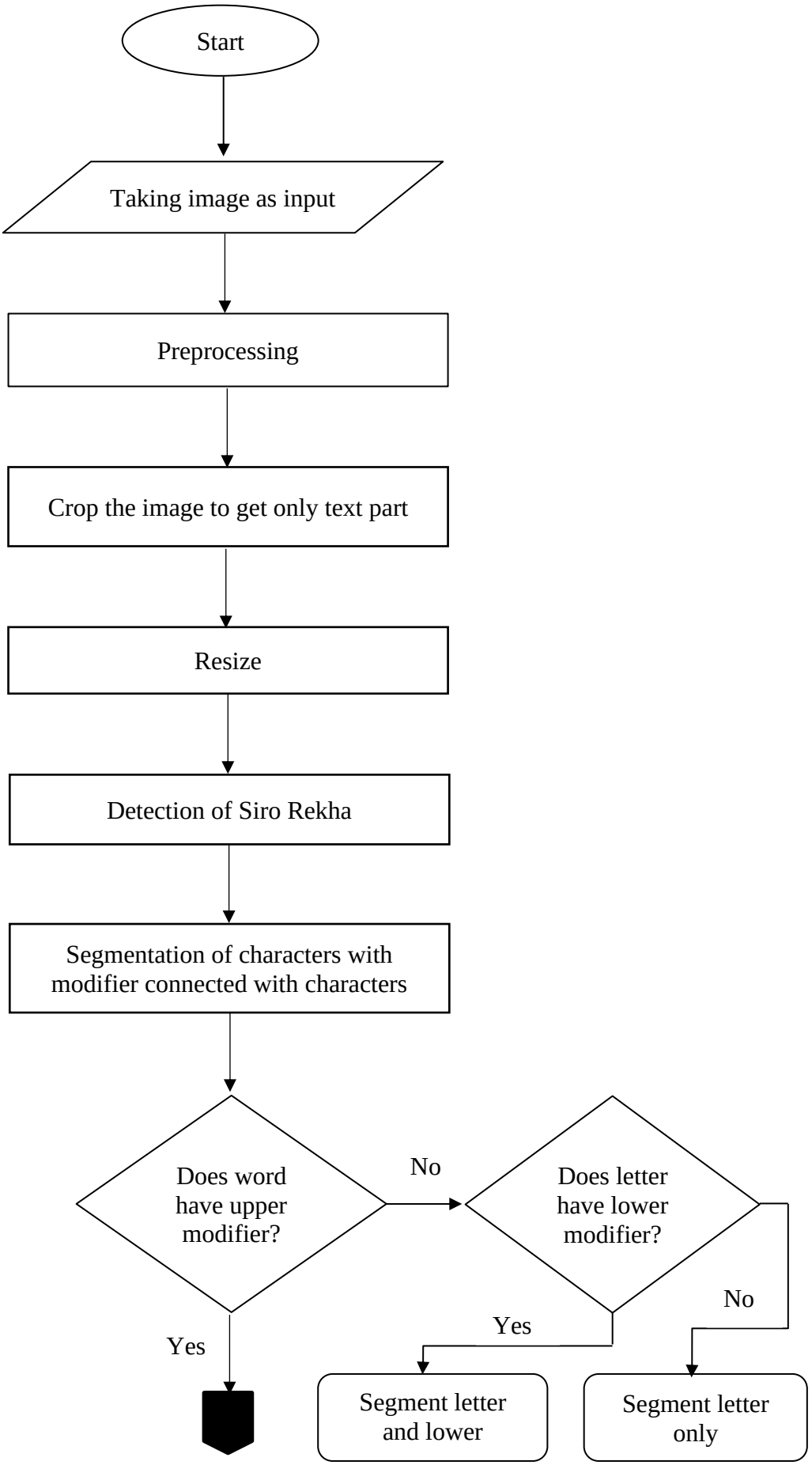
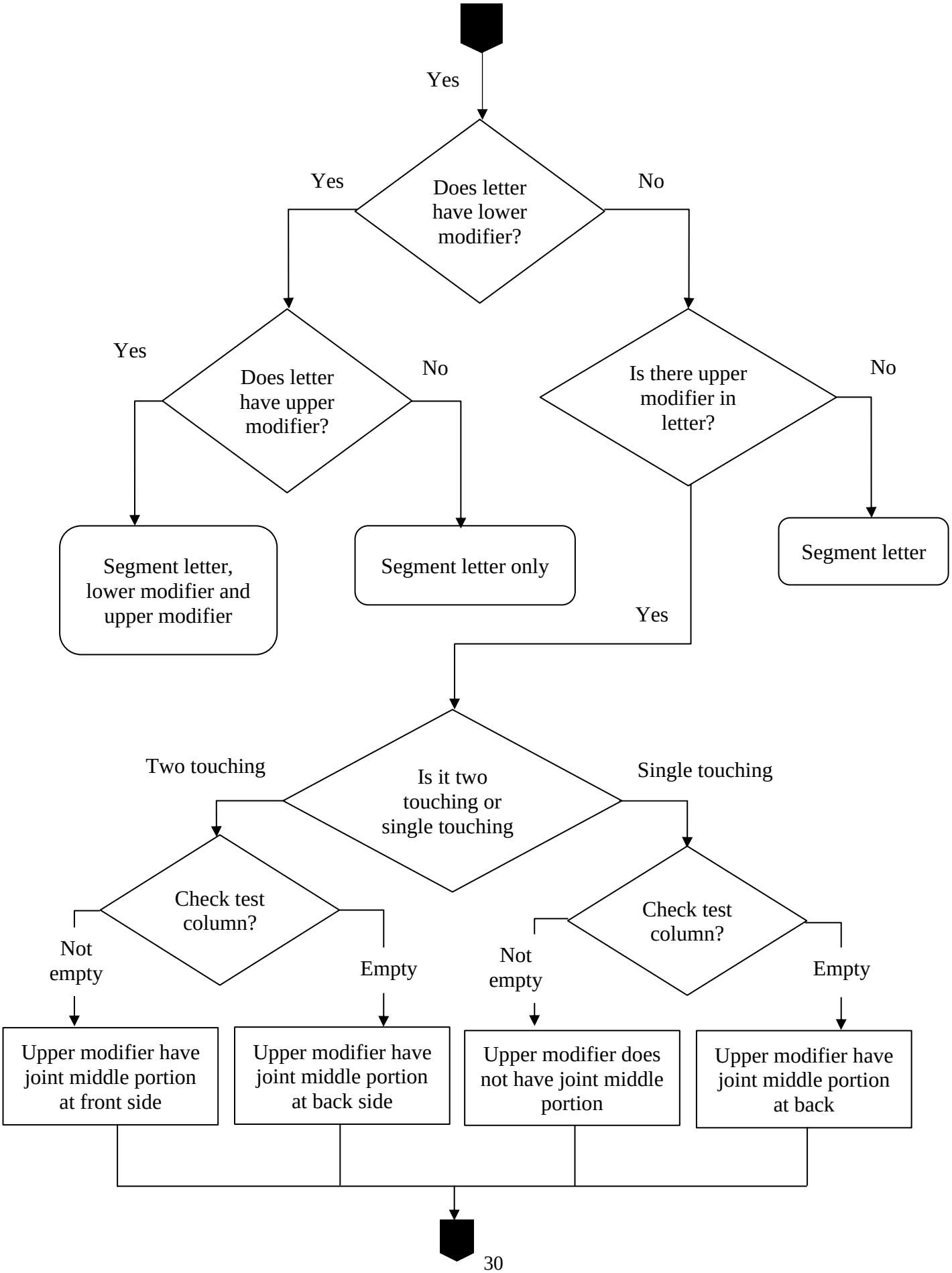


Figure 4- 4: Segmentation process


Start
Taking image as input
Preprocessing
Crop the image to get only text part
Resize
Detection of Siro Rekha
Segmentation of characters with modifier connected with characters
Does word have upper modifier?
No
Does letter have lower modifier?
Yes
No
Yes
Segment letter and lower
Segment letter only


Yes
Does letter have lower modifier?
Yes
No
Does letter have upper modifier?
Yes
No
Is there upper modifier in letter?
No
Segment letter, lower modifier and upper modifier
Segment letter only
Segment letter
Yes
Is it two touching or single touching
Two touching
Single touching
Check test column?
Not empty
Empty
Check test column?
Not empty
Empty
Upper modifier have joint middle portion at front side
Upper modifier have joint middle portion at back side
Upper modifier does not have joint middle portion
Upper modifier have joint middle portion at back

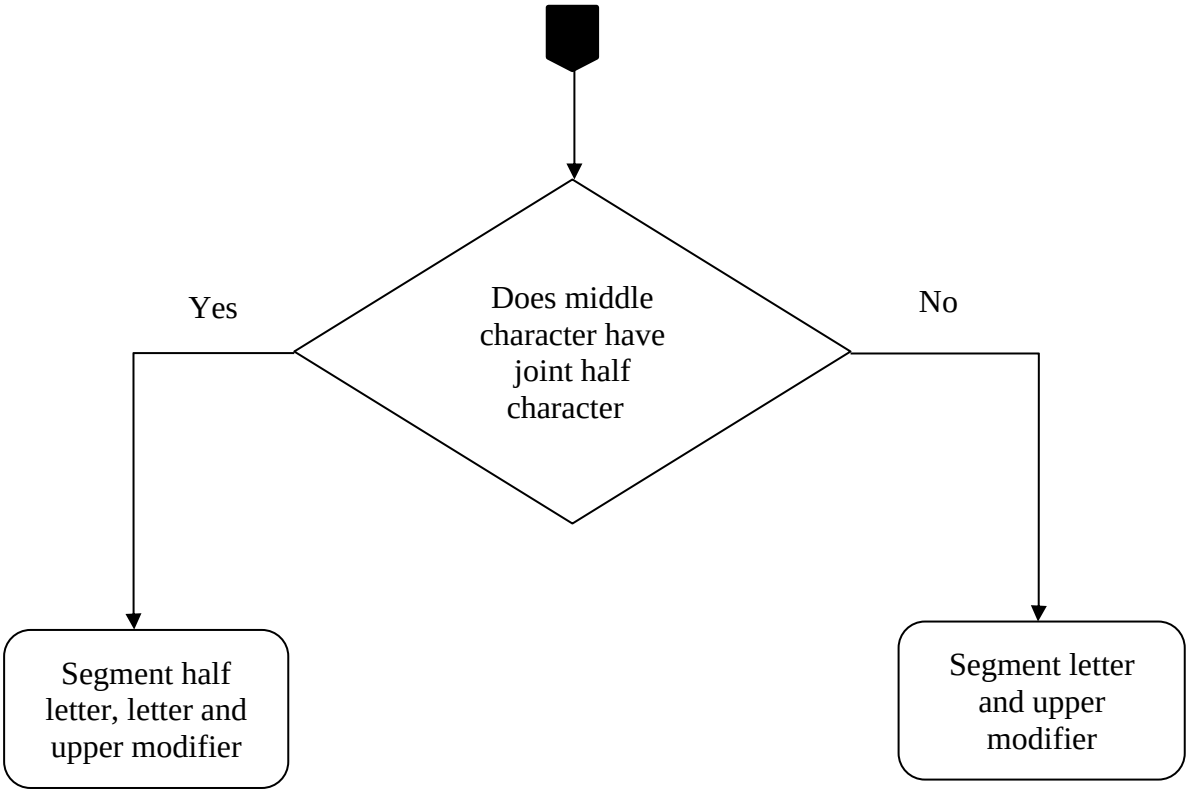


Figure 4- 5: Flowchart for Segmentation

4.3.5 Building Neural Network (CNN)

A CNN is a multiple layer neural system with a specialized design for identifying complicated data characteristics. The framework of a generic CNN is seen below. It may be used to categorize the subject matter of various photographs. The photos might be fed into the model as input. CNN, like ANN, is impacted by the inner functioning of the human mind. CNNs can categorize pictures by extracting characteristics, similar to how a person's brain looks for patterns to identify things.

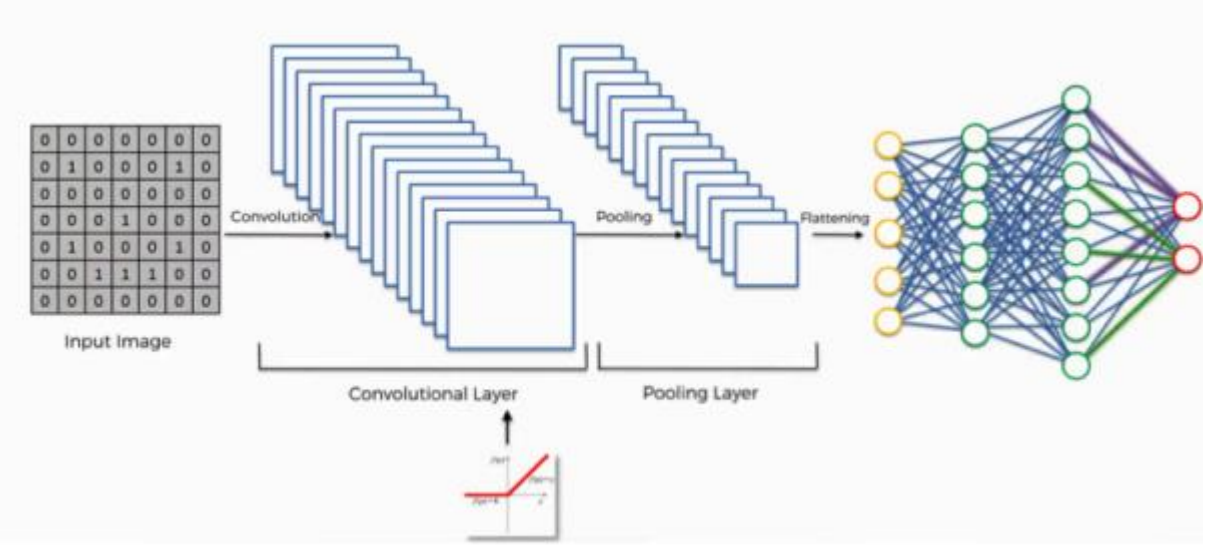


Figure 4- 6: Basic CNN Architecture *[17]*

The CNN has set of layers:

1. **Convolution Layer**

   The fundamental components of convolutional neural network are convolutional layer. A convolution is the straightforward process of applying a filter to an input to produce an activation. A feature map, which shows the positions and strength of a detected feature in an input, such as an image, is produced by repeatedly applying the same filter to an input.

   Convolutional neural networks are novel in their capacity to extract knowledge a large number of parallel filters tailored to a training dataset while adhering to the limitations of a particular predictive modeling issue, like image classification. As a result, input visuals could have extremely specialized qualities found anywhere.

2. **Max-Pooling Layer**

   The Max-Pooling process is used to down sample (pooled) a feature map, which calculates the highest possible value for feature map patches. It is commonly employed after a convolutional layer. It adds a smidgeon of translating invariance, which implies that adjusting the image's size has minimal effect on the results of the majority of pooled outputs.

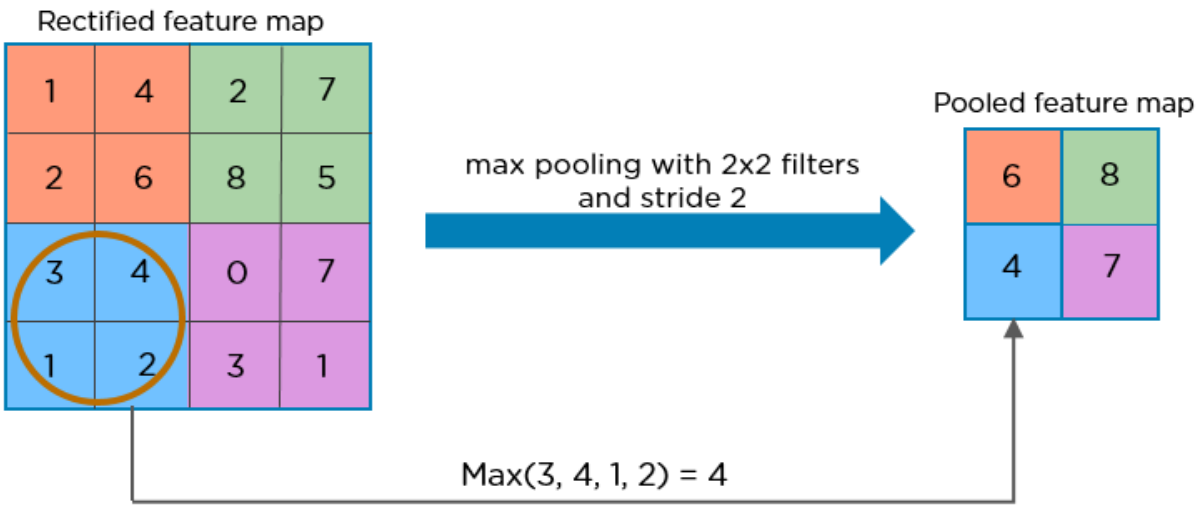


Figure 4- 7: Max Pooling

3. Dropout Layers

It is a layer that is really employed in the neural network to prevent over-fitting. This layer randomly slices two neurons from different layers apart. In order to avoid overfitting, the Dropout layer randomly sets input units to 0 at a frequency of rate at each step during training. Not-zero inputs are scaled up by 1/ (1 - rate), keeping the sum of all inputs constant.

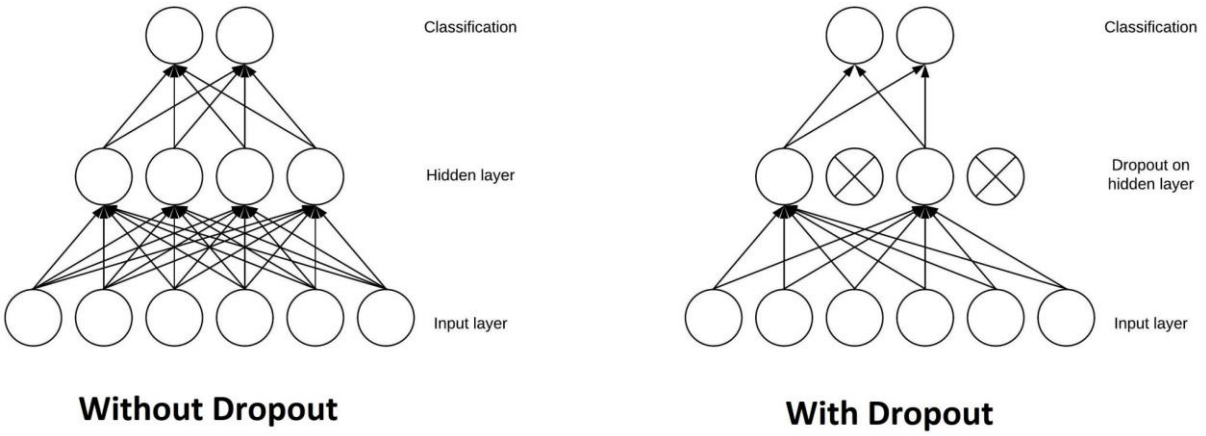


Figure 4- 8: Dropout layer

4. Flatten Layer

The resultant 2-Dimensional arrays from pooled feature maps are all flattened into single, extended continuous linear vector.

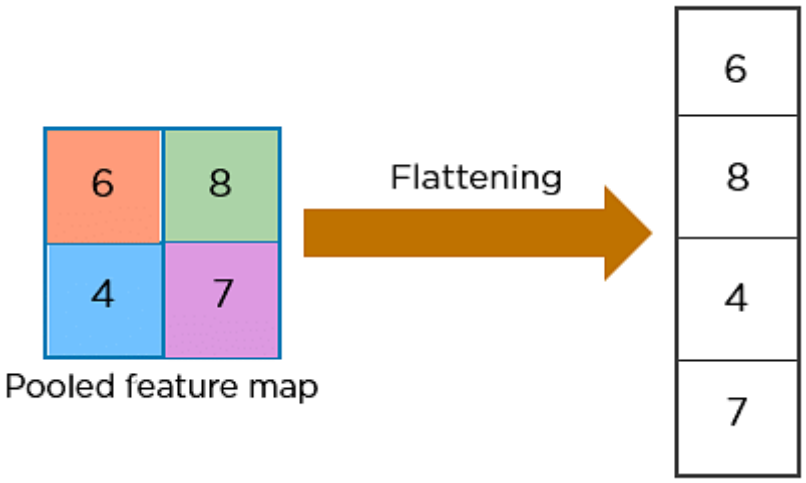


Figure 4- 9: Flattening Layer

5. Dense Layer

In any neural network, a layer that is densely linked to its preceding layer indicates that every neuron in the layer is connected to every other neuron in the layer above it. In artificial neural network networks, this layer is the one that is most frequently utilized.

6. Relu Layer

The piecewise linear function known as ReLU, or "rectified linear activation," outputs zero if the input is negative and the input directly if it is positive. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance.

Compared to sigmoidal functions like sigma(x) and tanh(x), this function offers two significant benefits.

- ReLU may be calculated extremely easily because all that is required is to compare the input to the value 0.
- Additionally, depending on whether or not its input is negative, it has a derivative of either 0 or 1

While sigmoidal functions have derivatives that tend to 0 as they approach positive infinity, ReLU always remains at a constant 1. As a result, even with large input values to the activation function, backpropagation of the mistake and learning can continue.

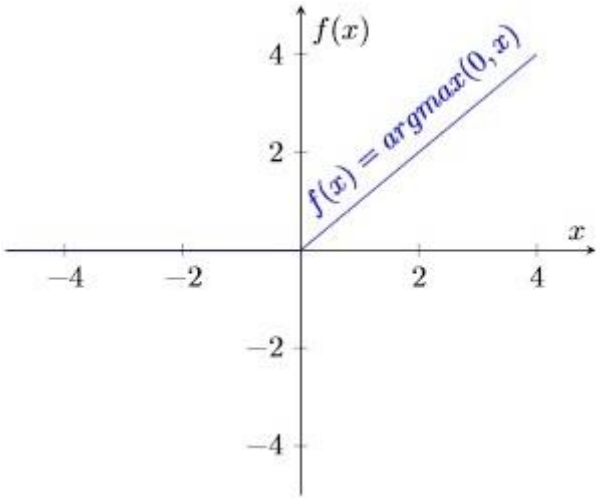


Figure 4- 10: ReLu layer

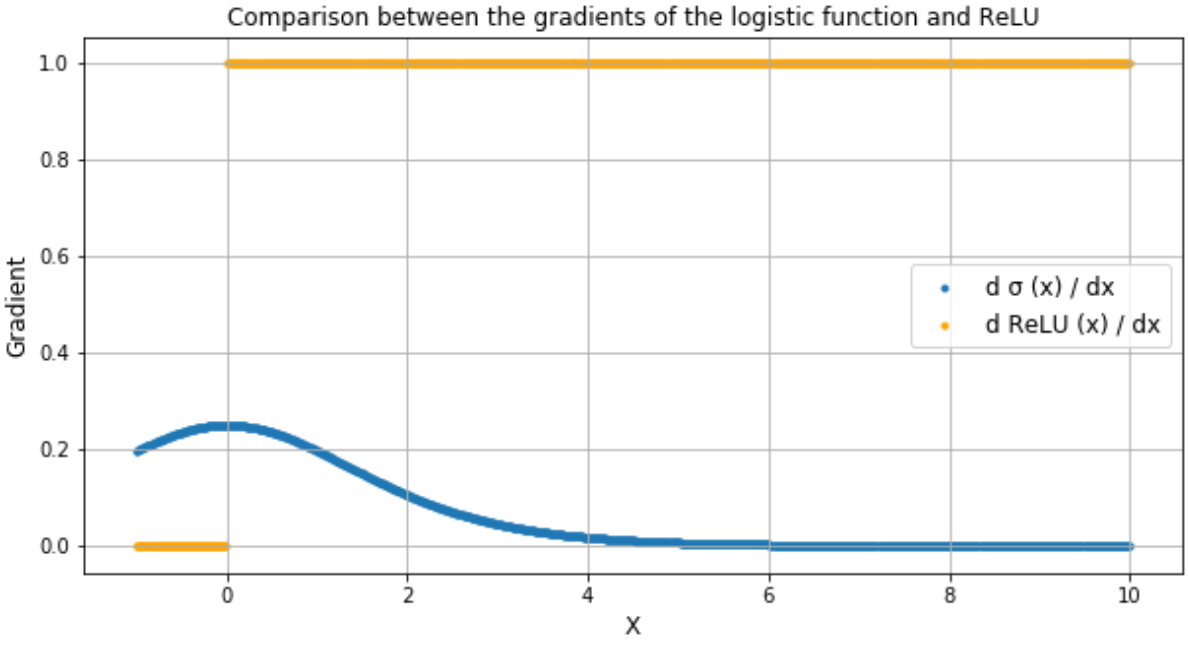


Figure 4- 11: Comparison between Logistic function and ReLU

Adam Optimizer

A new development in deep learning applications for computer vision and natural language processing is the Adam optimization technique, a stochastic gradient descent extension. It

integrates the "gradient descent with momentum" method and the "RMSP" algorithm, intuitively.

The Adam optimizer combines the following two methods of gradient descent:

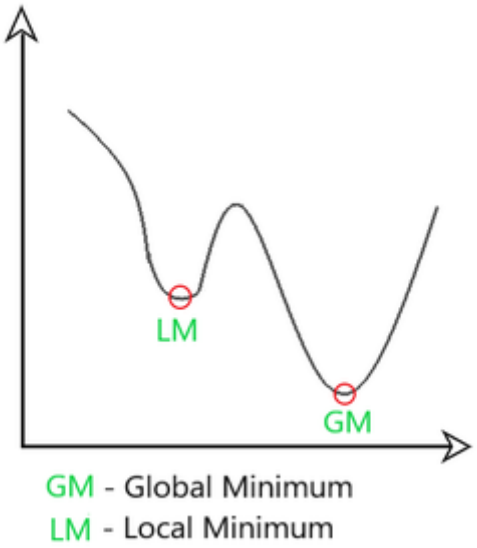
Momentum:

By using the "exponentially weighted average" of the gradients, this approach is used to speed up the gradient descent algorithm. The technique converges quicker towards the minima when averages are used.

RMSP (Root Mean Square Propagation):

An adaptive learning system called Root Mean Square Prop, or RMSprop, aims to enhance Adaptive Gradient Algorithm (AdaGrad). It uses the "exponential moving average" as opposed to AdaGrad's method of computing the cumulative sum of squared gradients.

The above two approaches have their own strengths, and Adam Optimizer relies on those strengths to provide a gradient descent that is better optimal.



Here, we regulate the gradient descent rate to attain the global minimum with the least amount of oscillation while taking large enough steps (step-size) to get through the local minima obstacles. In order to effectively attain the global minimum, combine the advantages of the proposed solutions.

Categorical Cross entropy

Multi-class classification problems employ the loss function categorical cross entropy. In these

tasks, the model must determine which one an example belongs to as there are many alternative categories.

The categorical cross entropy loss function calculates the loss of an example by computing the following sum:

$$L_{CE} = -\sum_{i=1}^{n} t_i \, log(pi) \ldots\ldots\ldots\ldots\ldots\ldots (4.1)$$

where,

$t_i =$ truth level

$p_i =$ Softmax probability for the i^{th} class

Sparse Categorical Cross entropy

The loss function described above applies to both categorical cross entropy and sparse categorical cross entropy. The only distinction between the two is how well labels are defined.

- When dealing with labels that are one-hot encoded, categorical cross-entropy is utilized. For instance, the following values for a 3-class classification issue are [1,0,0], [0,1,0], and [0,0,1].
- Labels are integer encoded in sparse categorical cross-entropy, for instance [1], [2], and [3] for a 3-class issue.

SoftMax Function

The SoftMax function transforms a vector of K real numbers into a probability distribution with K alternative outcomes. It is often referred to as softargmax or the normalized exponential function. It is applied in multinomial logistic regression and is an extension of the logistic function to many dimensions. The SoftMax function is frequently employed as a neural network's final activation function to normalize output to a probability distribution over expected output class.

4.3.6 Word Recognition

The characters will be divided up into several Devanagari characters that are included in the training dataset, DHCD, during the classification step. Following the scanning of the words, every written or printed symbol is compared to the comparable character stored inside the pre-defined class for the categorization of each character. This level employs CNN for classification, with 46 Devanagari script classes. As a result of the cropped picture, each character is categorized.

4.3.7 Digitized Nepali Text

Finally the relevant structured digitized text is obtained throughout this process.

5. IMPLEMENTATION DETAILS

5.1 Implementation Details in Softwares

5.1.1 Implementation in Python

All the coding was done on python 3.10 along with popular libraries like NumPy, Matplotlib, OpenCV, Keras, TensorFlow and other built-in libraries.

NumPy: Large, multi-dimensional arrays and matrices are supported. Additionally, it offers many of sophisticated mathematical operations for using arrays.

OS: Offers assistance for employing operating system-dependent features including the ability to read, write, or view files and directories.

OpenCV (CV2): OpenCV is a machine learning and computer vision library. It provides tools for reading, writing, and manipulating pictures.

Keras: Layers, cost functions, optimizers, and activation functions are some of the widely used neural-network components that are implemented. It offers RNN and CNN implementations.

Matplotlib: Offers interactive visualizations and plotting tools.

TensorFlow: A complete open-source framework for building machine learning applications. It is a symbols math toolbox that uses information flow and distinguished programming to perform numerous operations aimed for neural network training as well as inference. It allows programmers to create machine learning applications by employing a variety of frameworks, tools, and community resources.

5.1.2 Implementation in Jupyter Notebook

In addition to displaying/editing/running notebook documents, Jupyter Notebook was used as major component during the development of digitization system in this project. It was also used for the reporting and visualization of train/test.

5.1.3 Implementation on Google Colab

Google Colab provides nearly 12gb of GPU, and CPU for model train/test online. Training the large dataset on our system was challenging due to lack of resources and Google Colab is nearly 8 times faster than our system so Colab is our best choice. But creating a working environment on Colab is really a challenging task.

5.2 Image Processing Model

The model training is finished and the intention was real world character recognition. Character Detection is an electronic conversion of images of typed, handwritten or printed text into machine-encoded text. Here input image is provided from directory. The input image is thus processed and segmented and provided to NN for prediction. In most of the cases, input image may contain lots of noise which make difficult for our NN to predict character on the image. The provided image may also include less brightness and not well written characters which make difficult or sometimes unable to predict characters. So provided images here are assumed as high quality and only these images are easier or possible to predict character.

Image acquisition along with cropping and resizing

The challenge of recognition is mostly affected by the image capture phase. Because the real-world picture will never be the same, regardless of how precise the model was during testing. There will be a lot of noise, blur, and other quality issues in a real-world photograph. The property of the picture captured is also affected by image acquisition tools like cameras. In the first stage, we scribbled a word on a blank piece of paper.

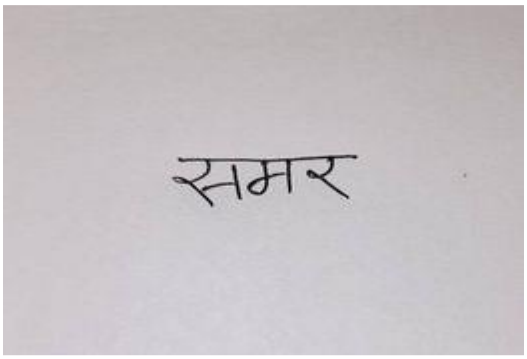


Figure 5- 1: Characters without any modifiers

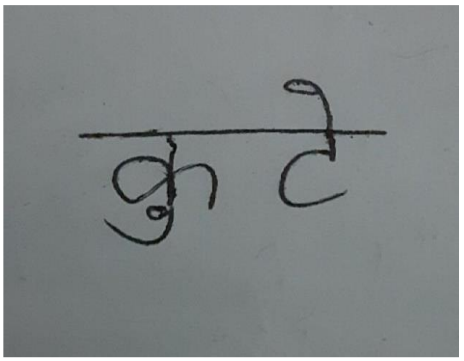


Figure 5- 2 : Characters with modifiers

Figure 5- 3: Half-form characters

Initially after the image is taken from the camera, we input to our system. Before developing the model, we need to devote a significant amount of effort to data pre-processing, often known as data purification. Outlier identification, missing value treatments, and removing undesired or noisy data are a few examples of data pre-processing.

Similar to that, "image pre-processing" refers to actions on pictures that are performed at the most basic level of abstraction. If entropy is a measure of information, then these actions diminish rather than enhance the information content of the picture. Pre-processing aims to improve the picture data by reducing unwanted distortions or enhancing specific visual properties that are important for subsequent processing and analysis tasks.

Initially, we crop the image into a specific part where our text is located. At this stage, the image was converted to grayscale, and the image pixels were organized into a NumPy array. Choosing the hues for the background and foreground was the next stage. If some noise is eliminated and thresholding is carried out, it is easier for an image to distinguish text and detect foreground color.

We define some threshold value as minimal percentage of total rows. Noise is any texture with a row value lower than this. From each row on scanning from top left corner, we count the number of pixels with value other than background color. If this value is greater than threshold, top part of text is found. Removing the part where the value is less than the threshold defined. Similarly, on scanning from bottom right corner to count the number of pixels with value other than background color. If this value is greater than threshold, bottom part of text is found.

After that we transpose the image, and do this process to find the left and right most part of it.

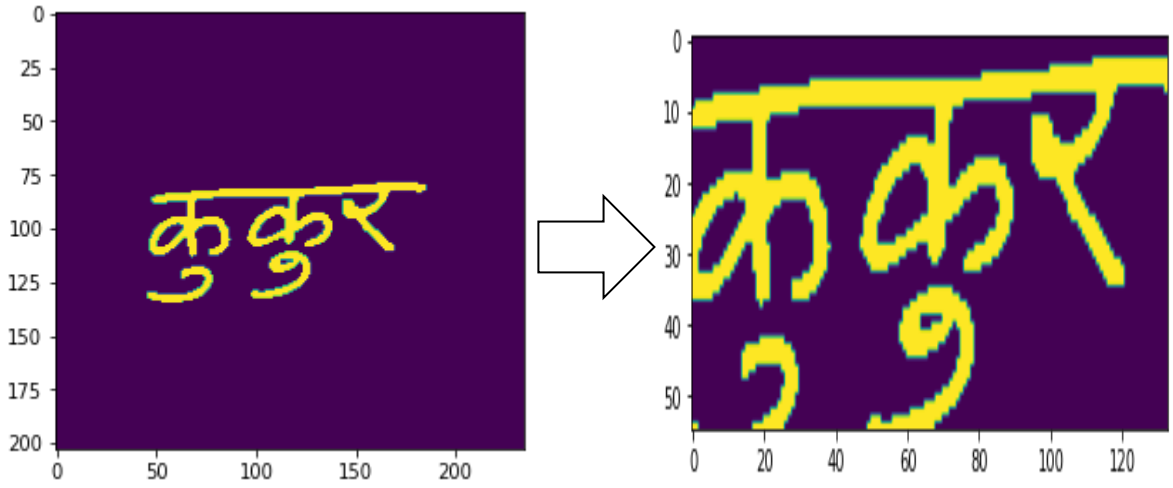


Figure 5- 4: Cropping of image

Resize the image

Images that are stretched or compressed into unreal forms teach a model that things don't seem like they do in reality, which we predict to have a negative impact on accuracy. The majority of neural network models require input images with square shapes; thus, each image must be evaluated to see if it is square or not and then cropped correctly. Cropping is the process of choosing a square area of the image, as demonstrated. We often pay attention to the center of the image while cropping.

In image preprocessing for neural networks, resizing images while maintaining aspect ratio ensures that the original proportions of the image are preserved. This is important as it helps prevent distortion in the image and ensures that important features of the image are not lost during the resizing process. To resize an image while maintaining aspect ratio, you first need to calculate the aspect ratio of the original image and then use that value to determine the new height and width while preserving the proportionality. This can be achieved by either choosing the longest edge (width or height) of the initial picture and using its value to calculate the value for the other edge, or by setting a maximum value for one of the dimensions and calculating the value for the other dimension based on the aspect ratio.

Header line (Shiro Rekha) detection and removal:

The element of a word that is most noticeable and distinctive is the header line. We may acquire the top and core-bottom sections of a word by dividing the header line. The word's horizontal density, or the number of pixels in each row, is computed to separate the header line, and the

area with the highest density, located in the top 55% of the word, is recognized. We have taken into account the top 55% of a word for two reasons:

- By focusing on the upper 65% of the word, high horizontal density sections in certain of the words (bottom 35% of the word) are segregated.
- The upper modifiers can occasionally be found in the top part of the word, such samples are taken care of by using the top 65% area.

The area with the maximum pixel density will provide us with the line's position because the header line spans the full word. In contrast to printed characters where the header line only covers one row, handwritten characters have a header line that spans numerous rows. Once the header line's location has been established, it is deleted by switching the background gray level for three rows (the header line, one row above it, and one row below it).

Figure 5- 5: Character with header line(Shiro Rekha)

Figure 5- 6: Image with header line(Shiro Rekha) removed

Character level segmentation

One of the main goals of our system is the segmentation of characters. It follows a set of processes, such as input, preprocessing, and segmentation, to complete the segmentation process. Both texts that were created by machines and those that were authored by hand undergo segmentation. There will be quite simple to segment text in any script if it has only one font, one alignment, and one width. as every character has a specific distinctive behavior. However, handling the procedure when the material is produced by hand will be challenging. Because several complications like overlapping, touching, skewness, breaking, and non-alignment happen while writing by hand. These factors will decrease the segmentation's success rate.

After the removal of header line, we scan the image through each row horizontally and determine values which are different from the background pixels of that image. We define a certain threshold space that separates character from each other on a word. After going through each row, if the space between two characters is less than the threshold defined then we crop it off.

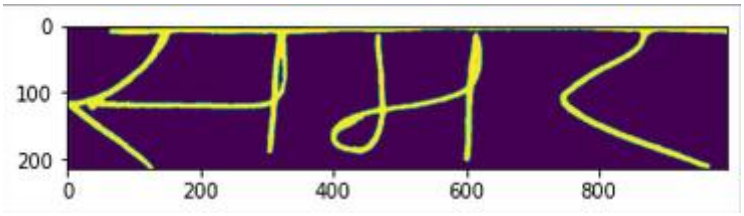


Figure 5- 7: Non-Segmented Characters

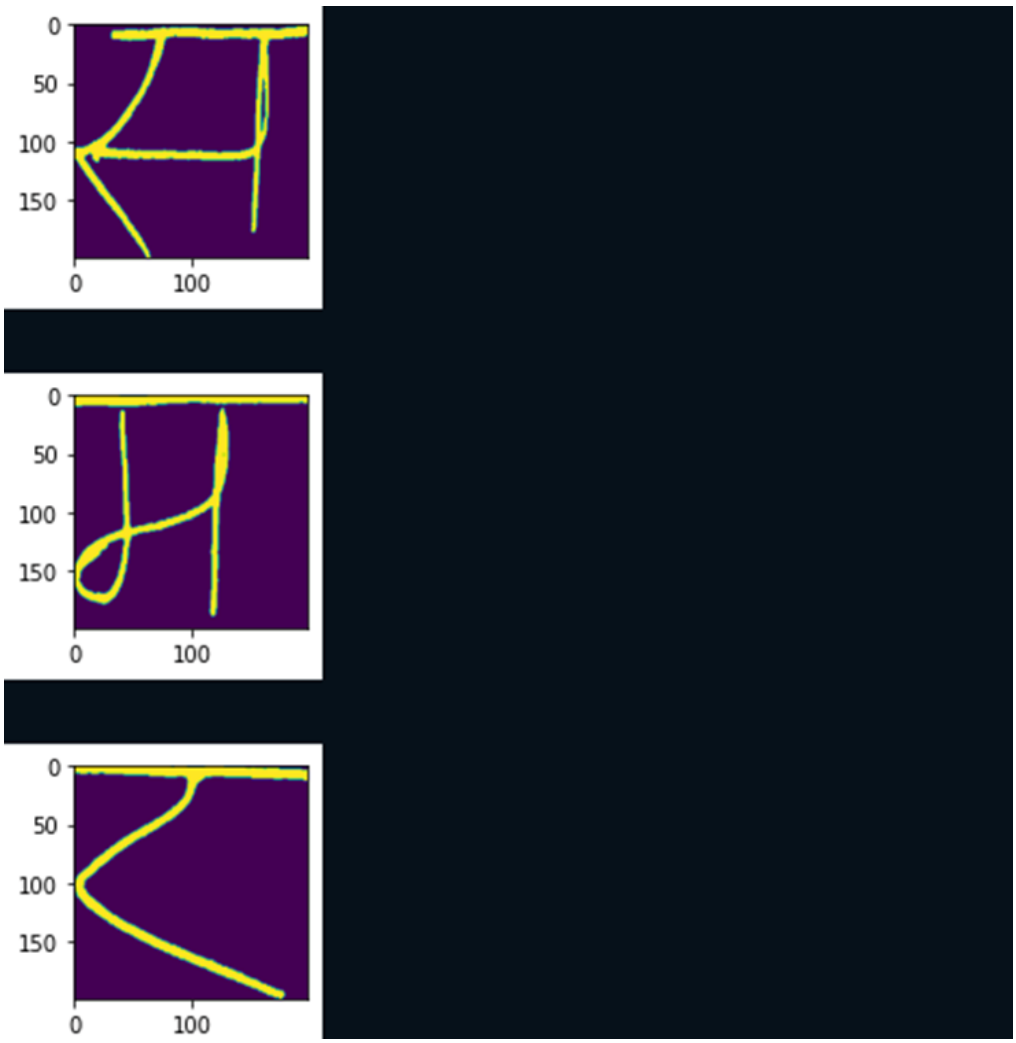


Figure 5- 8: Segmented Characters

Segmentation of Upper modifiers

The process of identifying and separating the numerous elements (such as matras and vowels) that modify the fundamental consonant letters in the script is known as the segmentation of upper modifiers in Devanagari script.

The upper modifiers in Devanagari often appear above the consonant letters to alter how they are pronounced. Vowels, half-vowels (matras), and different diacritical signs can all be used as these modifiers.

Many natural language processing (NLP) activities, including optical character recognition (OCR) and text-to-speech (TTS) systems, need the segmentation of upper modifiers. It makes it possible for a more accurate depiction of the text and aids in maintaining word pronunciation

and meaning.

We use an algorithm to extract each upper modifier independently and deal with it accordingly. The suggested method for upper modifier segmentation is shown in Figure.

Following are the steps:

1. The upper modifiers can be divided into two categories. Based on how frequently they contact the header line at various points, they are categorized. Depending on the writing style qualities, they either touch once or twice. The touching point is shown by red circle .

(a)

Figure 5-9: (a) One touch upper modifiers

(b)

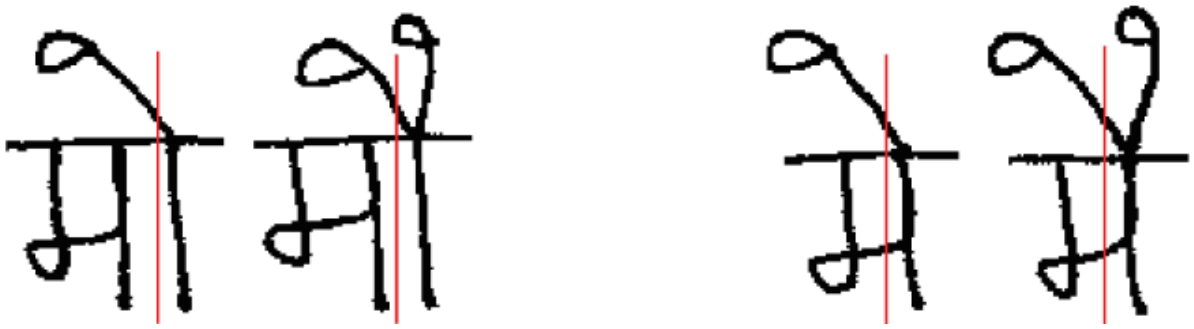
Figure 5-9: (b) Two touch upper modifiers

2. If just one touch was made, look up the class of the next modifier, if any, in the order from left to right. Go to step 3 if there is no subsequent modifier or if the subsequent modifier is not a single touching modifier. However, if there is just one touching modifier, see if the touching points of the current and subsequent modifiers are closer together or not. If not, proceed to step 3 instead. If not, we apply to join them and go to the next round. The whole range of upper modifier options is displayed here.

(a) (b) (c) (d)

Figure 5- 10: (a) No additional modifier. Figure 5- 10: (b) The next modifier does not involve only one touch. Figure 5- 10: (c) Single touching is the next modifier, but the touching locations are not closer together. Figure 5- 10: (d) Single touching is the next modification, and the touching spots are closer together.

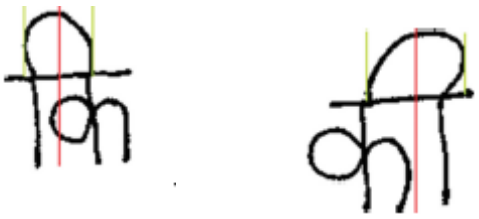
3. Search for the column that is some threshold values (column values) away from the touching point. Connect it to the component in the middle zone that is directly below from the computed column above, if it is over a blank space in the middle zone. If it sits on top of another component in the center zone, just show it separately.



(a) (b)

Figure 5- 11: (a) Calculated column (shown in red) is located over a space in the middle zone that is empty. Figure 5- 11: (b) Red column is located on a component in the middle zone.

4. Search for the column that is some threshold values (column values) away from the back touching point. Connect it to the component in the middle zone that is directly below from the computed column above, if it is over a blank space in the middle zone then the middle portion of the upper modifier is at the back. If it sits on top of another component in the center zone, the middle portion of the upper modifier is at the front.



(a) (b)

Figure 5- 12: (a) The middle column is on a component in the middle area. Figure 5- 12: (b) The middle column is on a gap in the middle area.

Figure 5- 13: Character with an upper modifier

Figure 5- 14: Segmented image

Segmentation of lower modifiers

The lower complement segment in the Devanagari script refers to the process of identifying and separating the different components that help modify the basic consonant characters in the script.

In Devanagari, lower modifiers often appear below consonant characters to modify their pronunciation. These modifiers can be in the form of virama (removing vowels inherent in consonants), anal (representing nasal sounds), and various diacritics.

The following steps are employed

- Initially each element is scanned to check the lowest point value, and stored in an array
- Range of the character values is identified, and we indicate certain threshold value
- After that, we compare those range value with certain threshold values applied
- If the range is greater than the threshold values, we get the average between them
- Else, the average value can be stated as the highest low point value

After that we compare the value of each character element with the lowest point average from the specific range, if the lowest point of each element is greater than the average value, it satisfies that the lower modifier is present in the character, if not we leave it in its actual state.

Figure 5- 15: Character with a lower modifier

Figure 5- 16: Segmented image with lower modifier

Segmentation of half-form characters

In each half-joint character, the right part is a full consonant and the left part always a semi-consonant. Now we want to determine if a character is single or half-form. While looking for this, we don't use the header row. For separation, the header row is present. Initially we check the dimension of the following image, we usually suspect that the width of the image is comparatively greater than the height which points out that the half-form characters is present within the character. After that we scan vertically(column-wise) moving from left to right to locate the pixel value accordingly on the basis of width of the middle portion just below the header line. We count the pixel value and store it in an array.

After the array is loaded, we take out the first few five or six stored content array values and pick out the middle pixel value where the half-form characters touching point is present. The difference is calculated to get the touching point of it. The highest stored value in that array is indicated as the maximum change in pixel value. The obtained value is used to crop the image

from the original image.

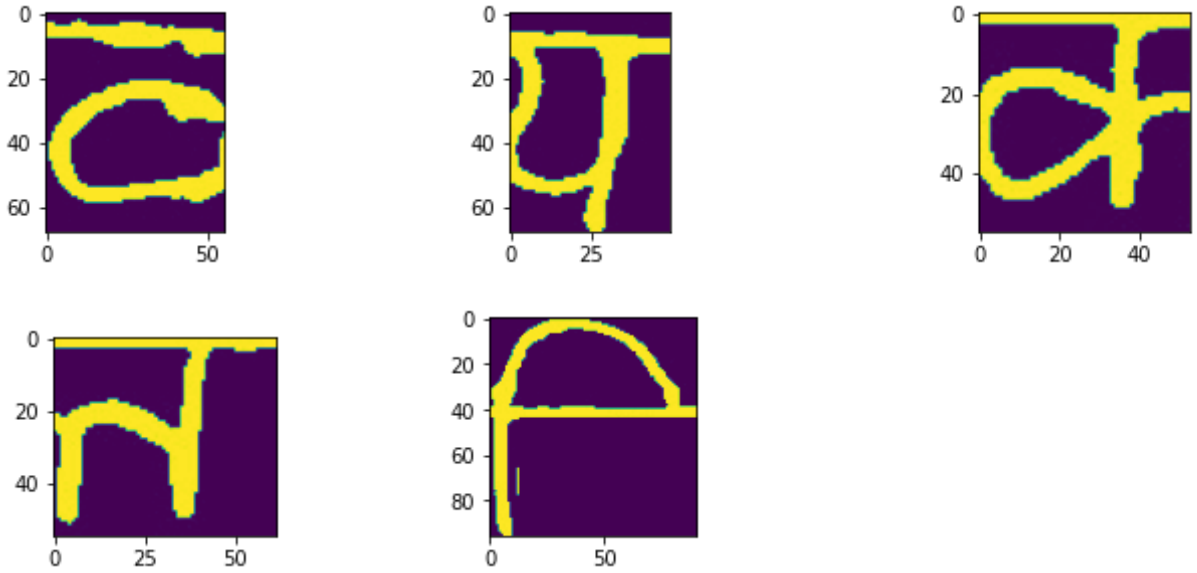


Figure 5- 17: Segmentation of half-form characters

Figure 5- 18: Image containing half-form, lower and upper modifiers

Figure 5- 19: First character segmented from the given image

Figure 5- 20: Lower modifier connected to the first character in a segmented form

Figure 5- 21: Segmented second character (half-form)

Figure 5- 22: Third character segmented

Figure 5- 23: Upper modifier segmented

Figure 5- 24: Fourth Character segmented

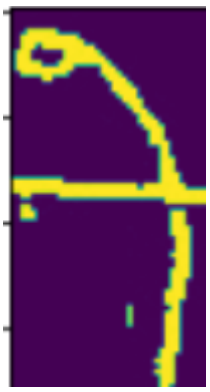


Figure 5- 25: Next upper modifier segmented

Figure 5- 26: Final Character segmented

5.3 Classification

The prediction technique was applied to each section. Prior to making a prognosis, the segment's form must be adjusted to match the neural network input. As a result, each segment is transformed into a 30 by 30 sized picture, and we also put 1-pixel background-colored borders all around it. The intake shape for our model will then be a 32 by 32 form for our segments. The neural network is next. It is considered that the character should be presented if the section has a high forecast. Depending on the image quality resulting to erroneous segmentation, predictions may also be incorrect.

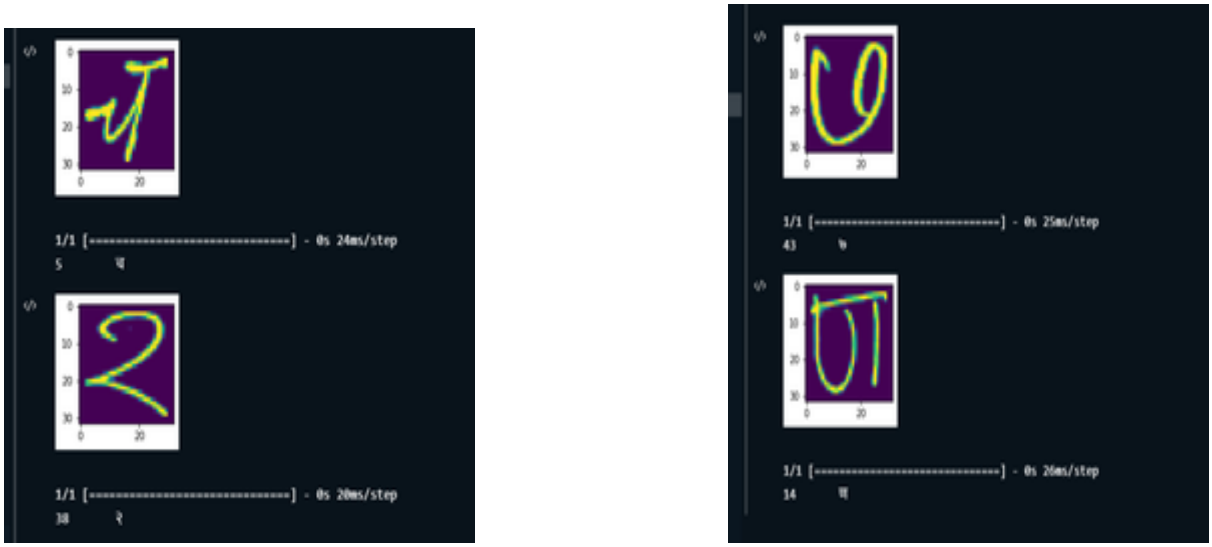


Figure 5- 27: Classification Results

6. RESULTS AND ANALYSIS

In our project up to this point, we are doing our work on segmentation and CNN model training.

6.1 Result of Segmentation

The segmentation process is one of the phases done in our project. The aim of this process is to separate the words from a sentence and then separate the letters from those words. This process is done so that the separated letters can then be passed through or given as input to the CNN model which then predicts the letter again reassembled to form the input word and the input sentence. Up to this point in our project, we have not worked on segmenting the sentence as a whole instead we have worked on segmenting a word that contains lower modifiers, upper modifiers, half characters and some special characters. Very complex words are also not being worked on as we have not worked on all the type of character's typefaces of the Devanagari script. The work on the upper and lower modifiers has been done currently and we get the better result on segmentation of them. Also, we the better segmentation for half form characters. There are many unique and different typeface characters whose dataset has not been created and the work on them is not completed so there won't be results to be shown on them.

We have got the results of segmentation of the uncomplicated or elementary words. There were different steps that we had to go through even while segmentation of the unadorned word. The different steps were input of the image, border definition, actual segmentation of the word and combination of recognized characters.

6.1.1 Input of the Image

In this process, the input is the image that is taken from a particular folder which is then grey scaled and shown in the output window so that we can see the input image. For example, one of the images shown in the output window is shown in the figure below:-

```
img = cv2.imread("C:/Users/NEPAL IS GREAT/Desktop/test.jpg", 0)

def show(img, figsize=(5, 5)):
    fig = plt.figure(figsize=figsize)
    plt.imshow(img, cmap=plt.cm.get_cmap('gray'))
    plt.show()
show(img)
```

[1] ✓ 10.4s

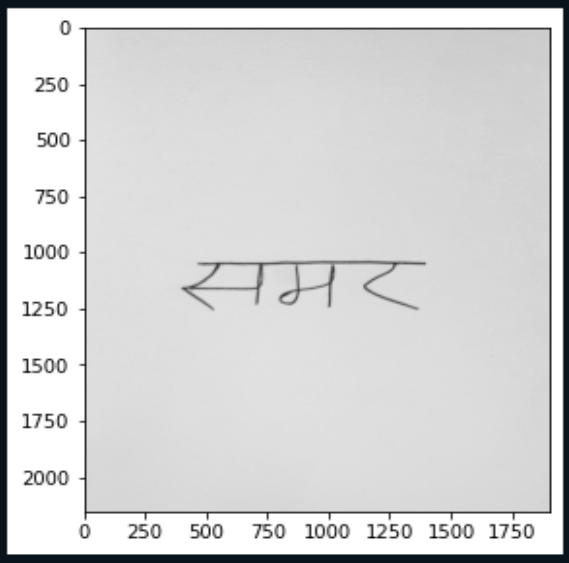


Figure 6- 1: Input image without modifier shown in output window

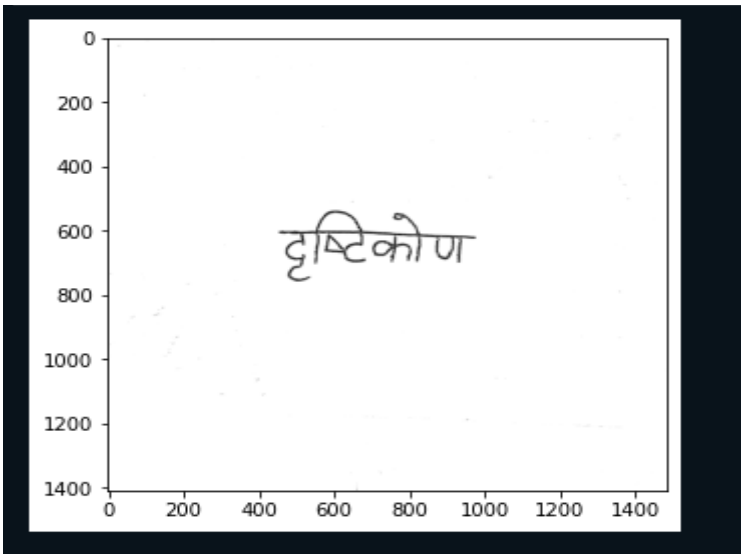


Figure 6- 2: Input image with lower modifier, upper modifier and half character

6.1.2 Cropping or Border Definition

In this step, the image is cropped to find out only the required portion of the image where the text lies. The noises or the unwanted garbage scribbles in the input image are also removed in this process. In this process, the image is scanned from top to bottom and left to right for texture in the image above a certain threshold, and the position of the text is defined. The textures in the image which are less than the threshold is considered as noise. The result of this step is displayed below:-

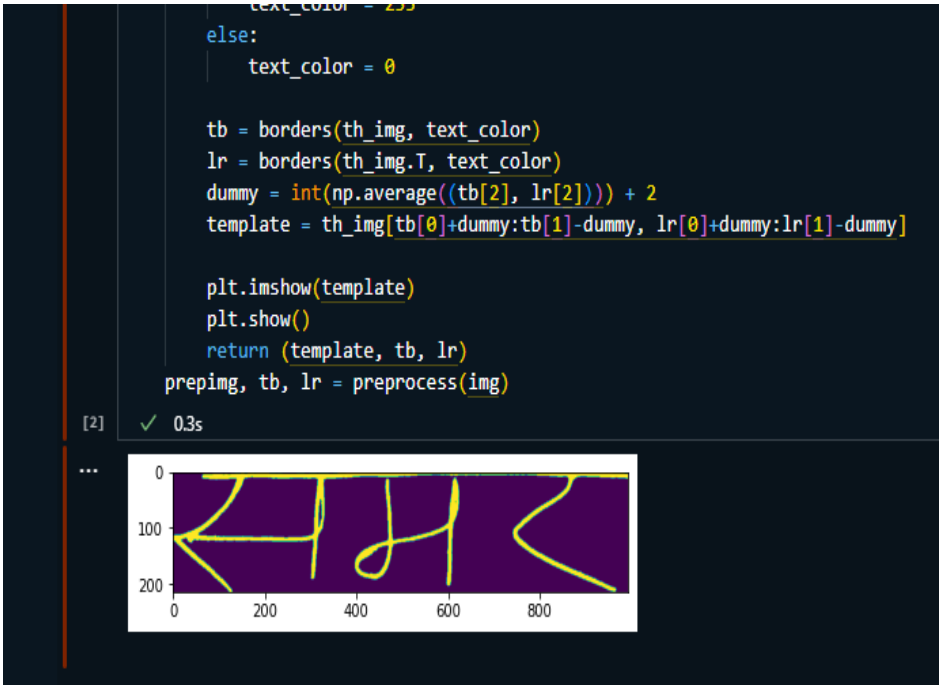


Figure 6- 3: Border definition of word not having modifiers

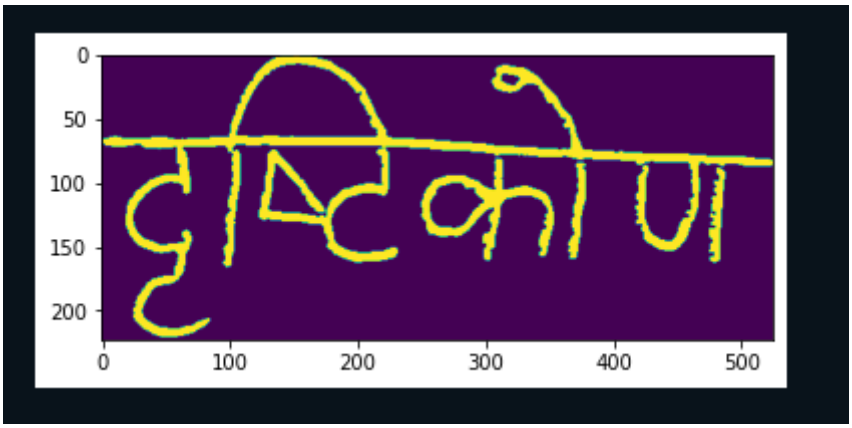


Figure 6- 4: Border definition of image having lower modifier, upper modifier and half character

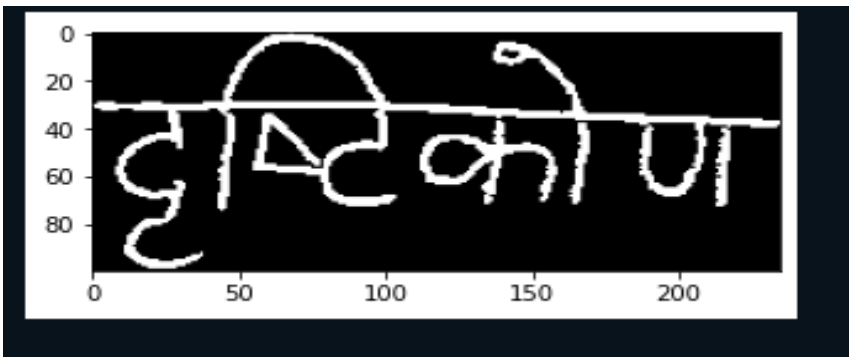
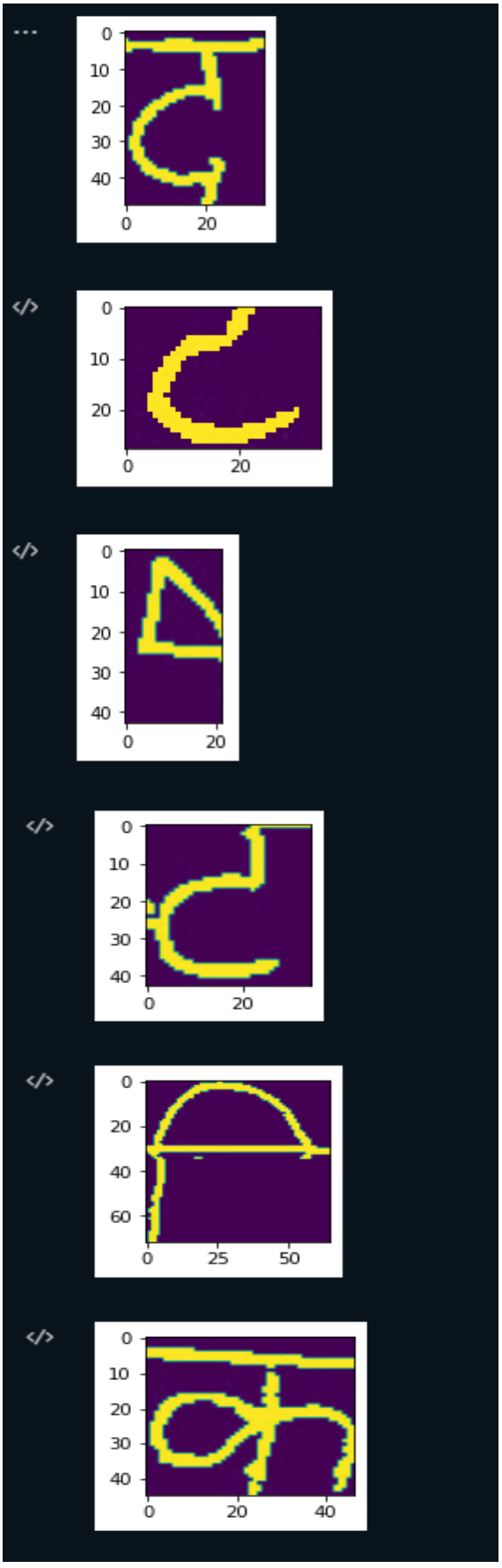


Figure 6- 5: Resized image

6.1.2 Division or Segmentation

In this step, the actual segmentation of the image is done where the image of the word is partitioned into the image of different words. This process is done by scanning the image but before that, the image is tilted, and then the image is scanned from to bottom and the image is segmented according to the color of the background color. The result of this step is displayed below:-

Figure 6- 6: Segmentation of word not having modifiers



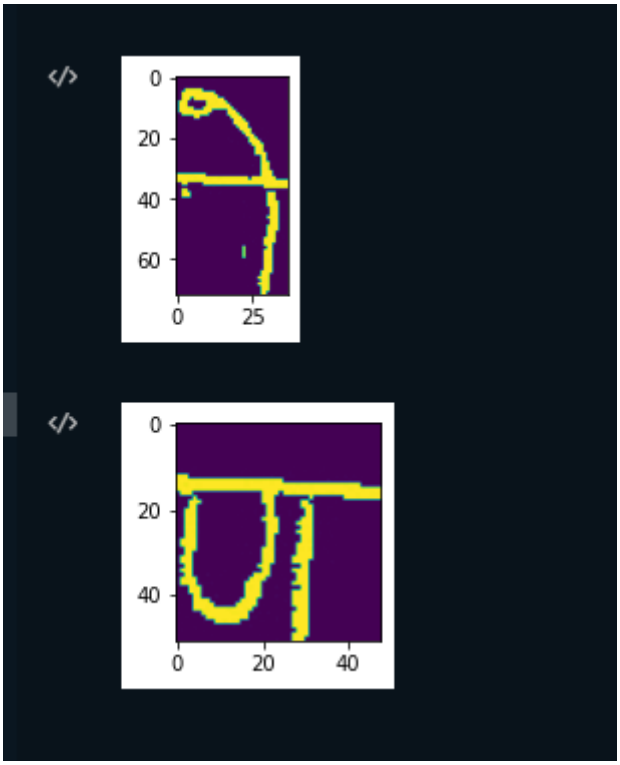
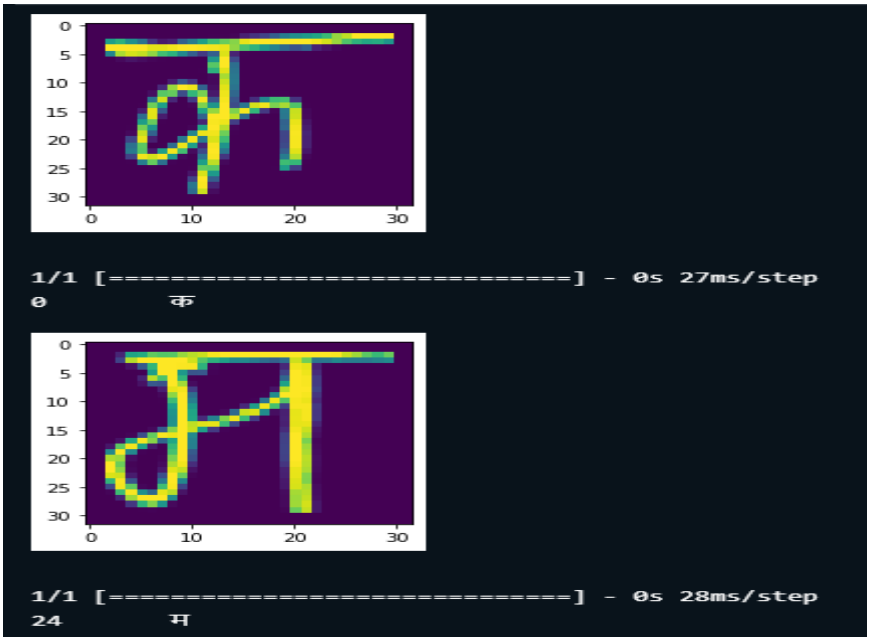
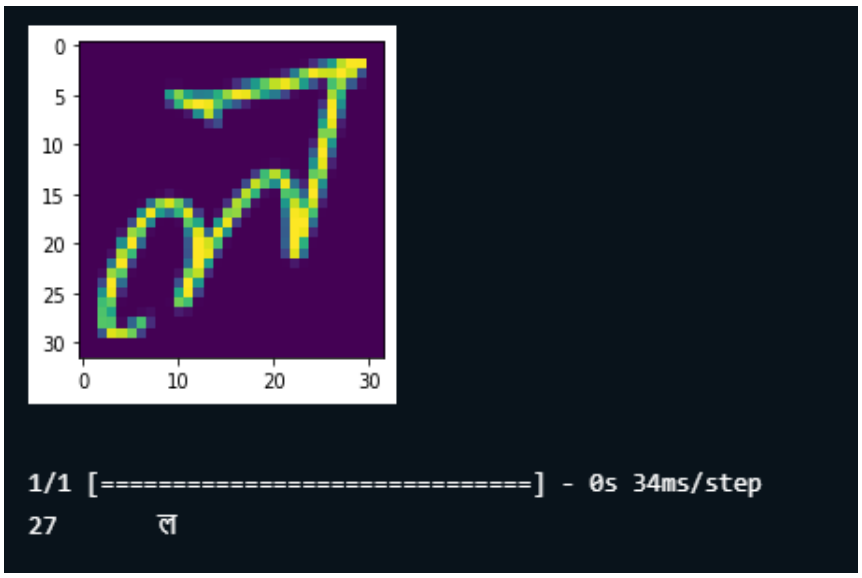


Figure 6- 7: Segmentation of word having lower modifier, upper modifier and half character

6.1.3 Combination of recognized Characters

A list of segmented characters is defined and label them. From the model , we get the array with the recognized character as 1 and other as 0. Lastly, the characters are appended to form a word. The result of this step is displayed below:





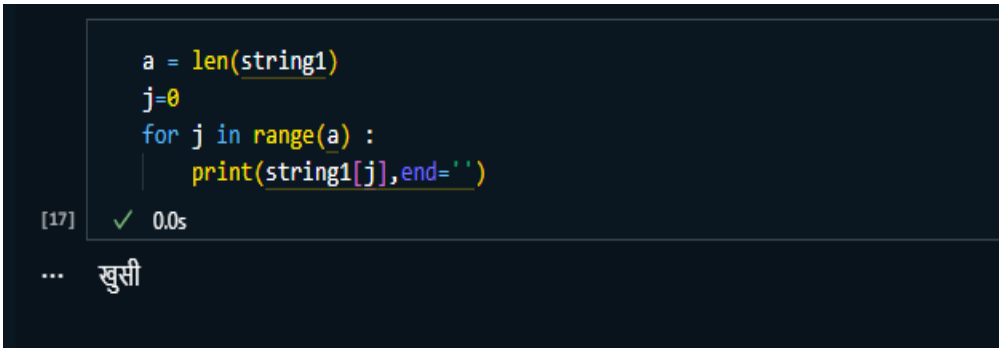


Figure 6- 8: Appended Characters from list of segmented characters

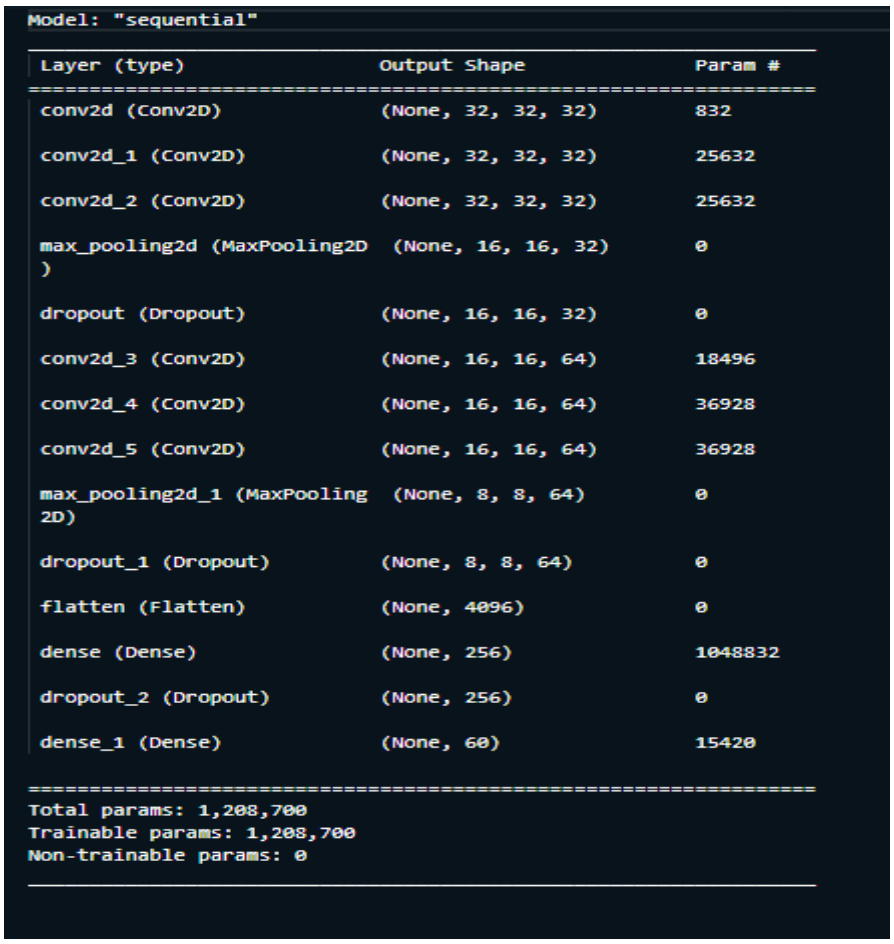
6.2 CNN Model

We are using the convolutional neural network model as our letter classifier or digit predictor. The convolutional neural network is best suited for the prediction of the images and especially

for those complex images that have pixel dependencies throughout. It uses relevant and different filters to find the different patterns in the image and to find the interconnection of those patterns in those images. The role of the convolution layer is to reduce the number of pixels that have to be processed by the dense layer without losing the important characteristics features of the image which may be essential for good prediction. Then, the Convolved Feature's spatial size is condensed by the Pooling layer. Through dimensionality reduction, the amount of computing power needed to process the data will be reduced. Furthermore, it aids in properly training the model by allowing the extraction of dominating characteristics that are rotational and positional invariant. The result of multiple of these convolutions (multiple filters) layers and then, the pooling layer is flattened and passed through a dense layer which may be multiple layers as well to classify the images.

There are four model of our CNN architecture named as main character model, half character model, lower modifier model and upper modifier where each model is divided into two model. Also, the model is created for sentence ending symbols. The summary of our architecture for main character model is given below and similar summary for others model.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 128)       1280

 max_pooling2d (MaxPooling2D  (None, 15, 15, 128)      0
 )

 dropout (Dropout)           (None, 15, 15, 128)       0

 conv2d_1 (Conv2D)           (None, 13, 13, 128)       147584

 max_pooling2d_1 (MaxPooling  (None, 7, 7, 128)        0
 2D)

 dropout_1 (Dropout)         (None, 7, 7, 128)         0

 conv2d_2 (Conv2D)           (None, 5, 5, 128)         147584

 max_pooling2d_2 (MaxPooling  (None, 3, 3, 128)        0
 2D)

 dropout_2 (Dropout)         (None, 3, 3, 128)         0

 flatten (Flatten)           (None, 1152)              0

 dense (Dense)               (None, 256)               295168

 dropout_3 (Dropout)         (None, 256)               0

 dense_1 (Dense)             (None, 128)               32896

 dropout_4 (Dropout)         (None, 128)               0

 dense_2 (Dense)             (None, 60)                7740

=================================================================
Total params: 632,252
Trainable params: 632,252
Non-trainable params: 0
_________________________________________________________________
```



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 32)        832

 conv2d_1 (Conv2D)           (None, 32, 32, 32)        25632

 conv2d_2 (Conv2D)           (None, 32, 32, 32)        25632

 max_pooling2d (MaxPooling2D  (None, 16, 16, 32)       0
 )

 dropout (Dropout)           (None, 16, 16, 32)        0

 conv2d_3 (Conv2D)           (None, 16, 16, 64)        18496

 conv2d_4 (Conv2D)           (None, 16, 16, 64)        36928

 conv2d_5 (Conv2D)           (None, 16, 16, 64)        36928

 max_pooling2d_1 (MaxPooling  (None, 8, 8, 64)         0
 2D)

 dropout_1 (Dropout)         (None, 8, 8, 64)          0

 flatten (Flatten)           (None, 4096)              0

 dense (Dense)               (None, 256)               1048832

 dropout_2 (Dropout)         (None, 256)               0

 dense_1 (Dense)             (None, 60)                15420

=================================================================
Total params: 1,208,700
Trainable params: 1,208,700
Non-trainable params: 0
_________________________________________________________________
```

Figure 6- 9: CNN model summary for main character model -1 and model -2

The model has and is being trained using the dataset of the different letters, sixty to be exact different types of letters. Each letter has about two thousand images for the model to train. Some of the letters were predicted using the model above after some training.

6.2.1 Training progress

We have created a four model in our project named as main character model, half character model, lower modifier model and upper modifier model. For each model, we made a two model and choose the model which has the better result. The model was trained for different epochs. It was trained using Adam optimizer and categorical cross entropy as loss. 20% of training data was split for validation. The batch size was 32.

Confusion Matrix

A confusion matrix is a N x N grid used to assess the effectiveness of a model for classification, where N is the amount of target classes. This matrix analyzes the actual values of the target to the predictions of the machine learning model. This gives us an in-depth understanding of how effectively our classification system is performing and the types of mistakes it generates. Precision is a measure of how many predictions are true positive out of all the total positive predicted. Mathematically,

$$Precision = \frac{TP}{\text{TP+FP}} \text{ .....(6.1)}$$

Recall is a measure of how many of the positive cases the classifier correctly predicted over aal the positive cases in the data. Mathematically,

$$Recall = \frac{TP}{\text{TP+FN}} \text{ .....(6.2)}$$

F1 - Score is a measure combining both precision and recall. It is harmonic mean of the two. Mathematically,

$$F1 - Score = \frac{2*\text{precision}*\text{recall}}{\text{precision+recall}} \text{ .....(6.3)}$$

The confusion matrix for each class was constructed where there were characters that were identified correctly while there was also cause of false identification. For each model, we have created a two model and constructed a confusion matrix. Lastly, we compare the confusion matrix and use the confusion matrix having better result.

A PR (Precision Recall) curve is essentially a graph with Precision on the y-axis and Recall on the x-axis. The precision-recall curve is used to assess the effectiveness of binary classification algorithms. It is frequently used in instances where the classes are severely uneven. The precision-recall curve is created by calculating and showing the precision vs recall for a single classifier at various thresholds.

The receiver operating characteristic (ROC) curve is often used to assess the effectiveness of binary classification techniques. It provides a graphical explanation of a classifier's effectiveness instead of a single value, unlike most other metrics. The ROC curve is created by computing and displaying the rate of true positives vs. the rate of false positives for a single classifier at different thresholds. The AUC is used to illustrate the ROC curve of a classifier. AUC is an abbreviation for the area under the (ROC) curve. The greater the AUC value, in general, the better a classifier performs for the given job.

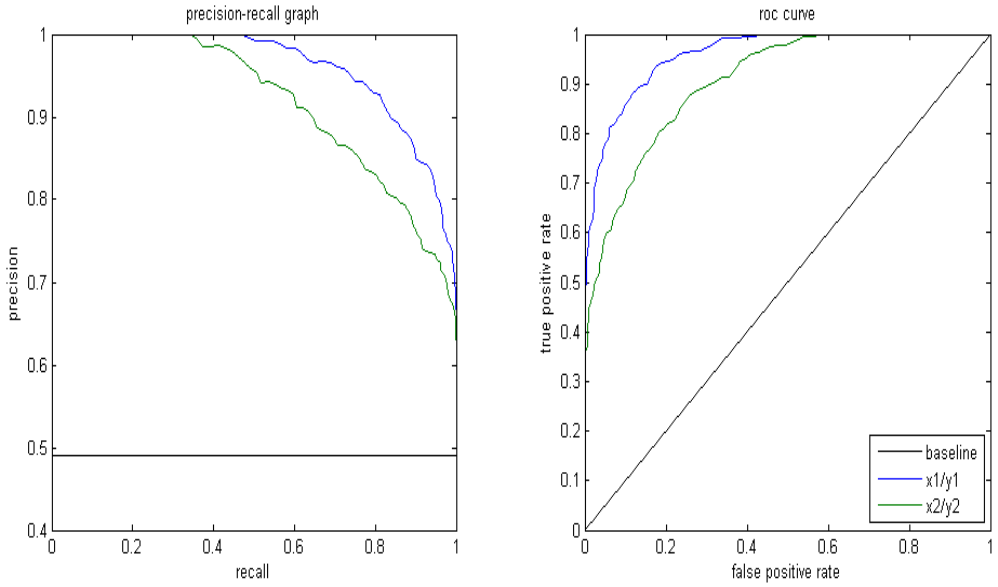


Figure 6- 10: PR and ROC Curves

Main character model -1

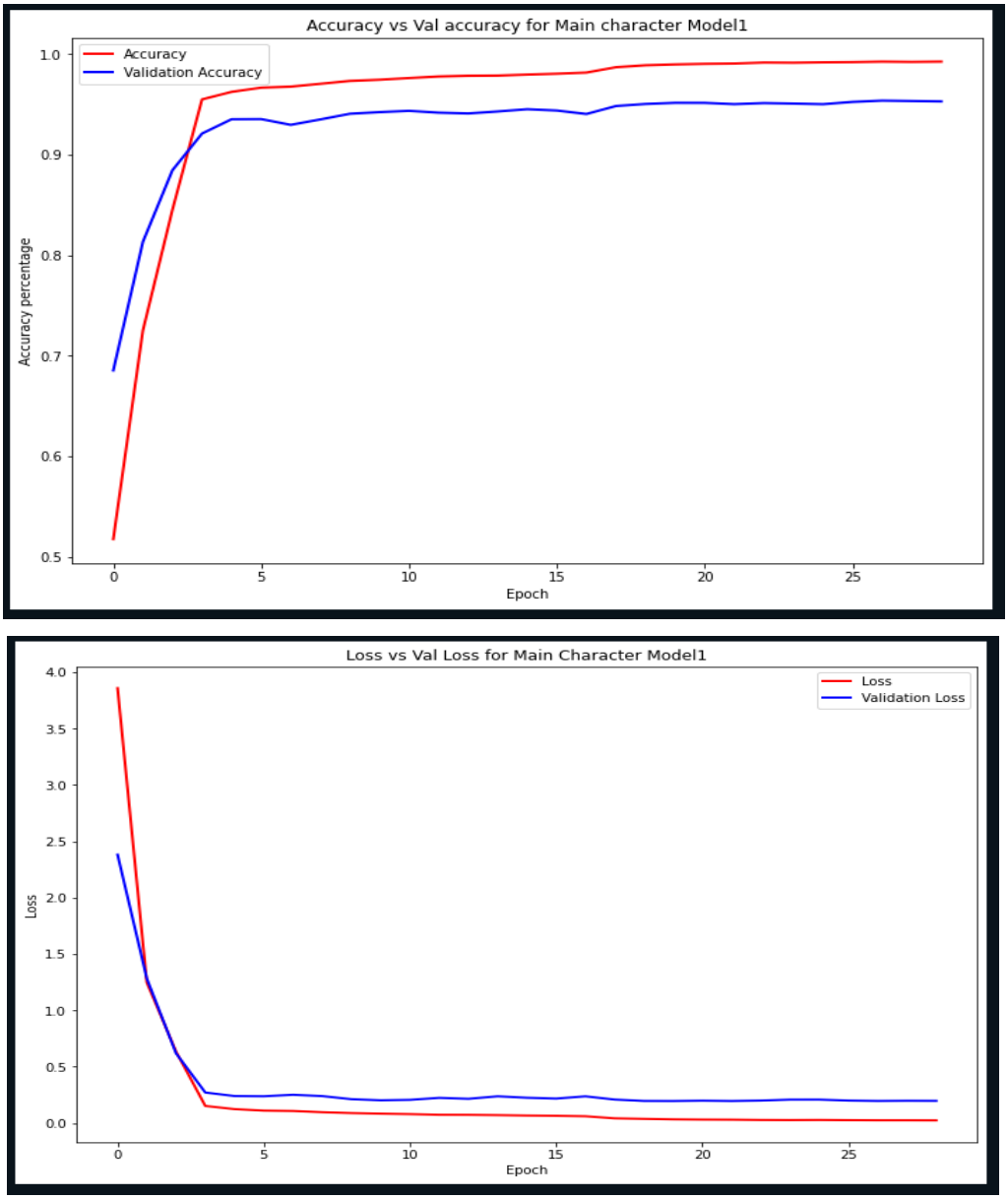


Figure 6- 11: Training accuracy vs validation accuracy and Training loss vs validation loss for main character model -1

Table 6- 1: Normalized confusion matrix for main character model -1

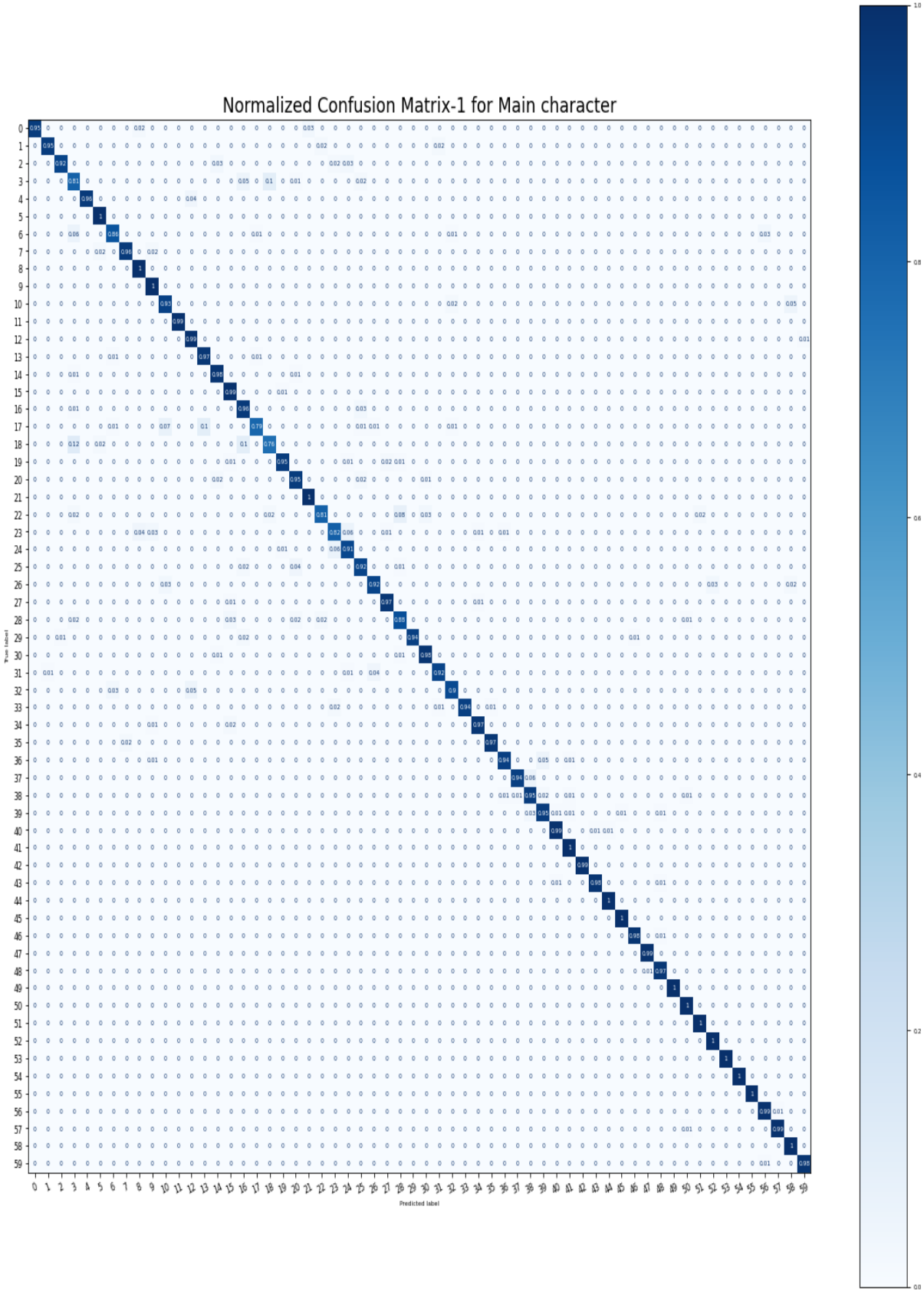


Table 6- 2: Evaluation metrices for main character model-1

	precision	recall	f1-score	support
0	0.99	0.95	0.97	400
1	0.98	0.95	0.97	400
2	0.97	0.92	0.94	400
3	0.76	0.81	0.79	400
4	1.00	0.96	0.98	400
5	0.95	1.00	0.98	400
6	0.94	0.86	0.90	400
7	0.96	0.96	0.96	400
8	0.94	0.99	0.97	400
9	0.94	1.00	0.97	400
10	0.89	0.93	0.91	400
11	0.99	0.99	0.99	400
12	0.92	0.99	0.95	400
13	0.90	0.97	0.94	400
14	0.95	0.98	0.97	400
15	0.92	0.99	0.95	400
16	0.83	0.95	0.89	400
17	0.97	0.79	0.87	400
18	0.86	0.76	0.81	400
19	0.97	0.95	0.96	400
20	0.92	0.95	0.93	400
21	0.97	1.00	0.98	400
22	0.94	0.81	0.87	400
23	0.90	0.81	0.85	400
24	0.89	0.91	0.90	400
25	0.91	0.92	0.92	400
26	0.95	0.92	0.93	400
27	0.97	0.97	0.97	400
28	0.88	0.89	0.88	400
29	0.99	0.94	0.97	400
30	0.95	0.97	0.96	400
31	0.96	0.92	0.94	400
32	0.94	0.90	0.92	400
33	0.99	0.94	0.97	400
34	0.98	0.97	0.97	400
35	0.99	0.97	0.98	400
36	0.97	0.94	0.95	194
37	0.98	0.94	0.96	194
38	0.90	0.95	0.93	196
39	0.93	0.95	0.94	198
40	0.97	0.99	0.98	188
41	0.99	1.00	0.99	202
42	1.00	0.99	0.99	400
43	0.99	0.98	0.98	400
44	0.99	1.00	1.00	400
45	0.99	1.00	0.99	402
46	0.98	0.98	0.98	400
47	0.99	0.99	0.99	400
48	0.97	0.97	0.97	400
49	1.00	1.00	1.00	309
50	0.97	1.00	0.99	400
51	0.97	1.00	0.99	400
52	0.96	0.99	0.98	400
53	0.99	1.00	1.00	400
54	0.99	0.99	0.99	400
55	1.00	1.00	1.00	400
56	0.96	0.99	0.97	400
57	0.99	0.99	0.99	400
58	0.93	0.99	0.96	400
59	0.99	0.98	0.99	400
accuracy			0.95	22683
macro avg	0.95	0.95	0.95	22683
weighted avg	0.95	0.95	0.95	22683

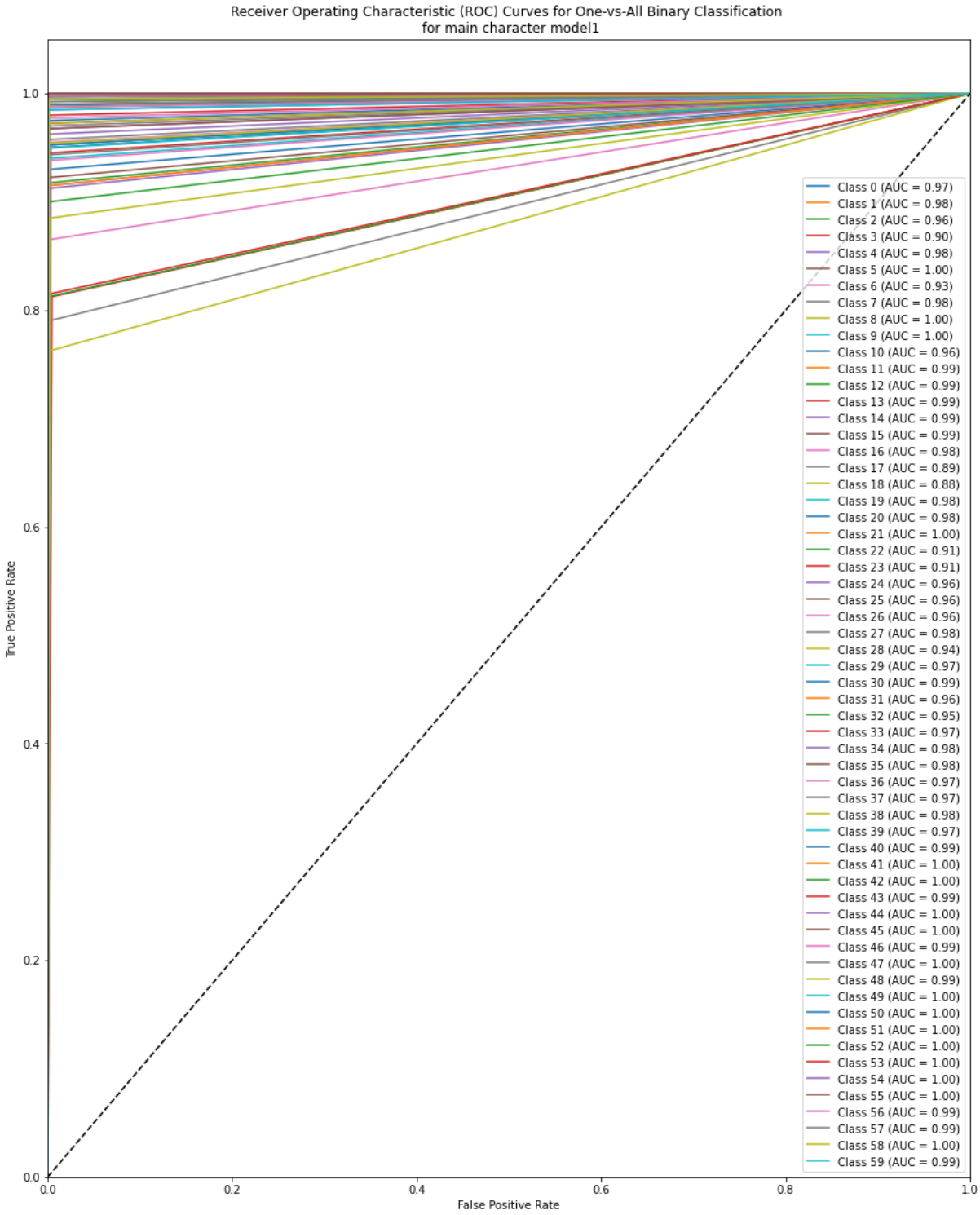


Figure 6- 12: ROC curve for main character model -1

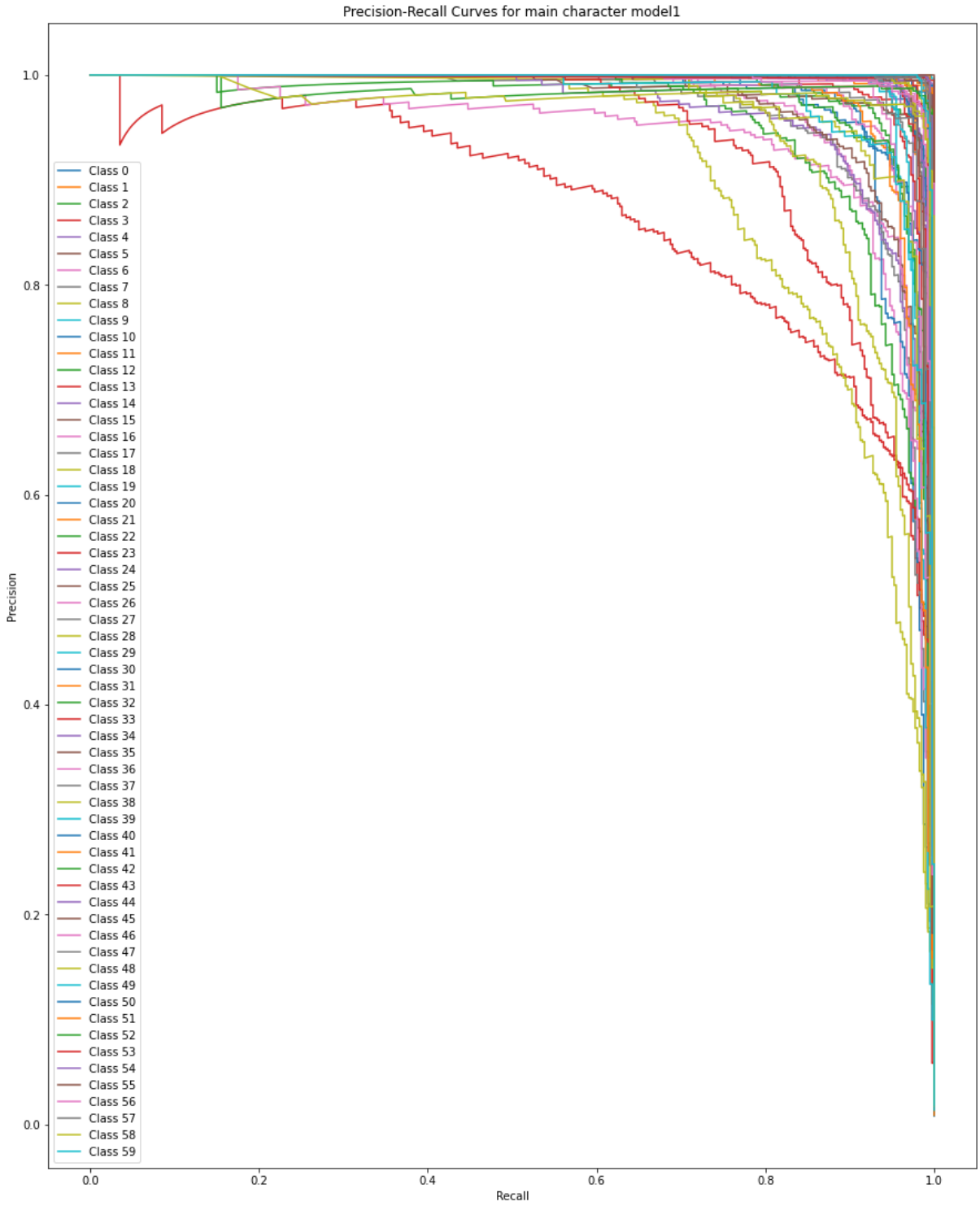
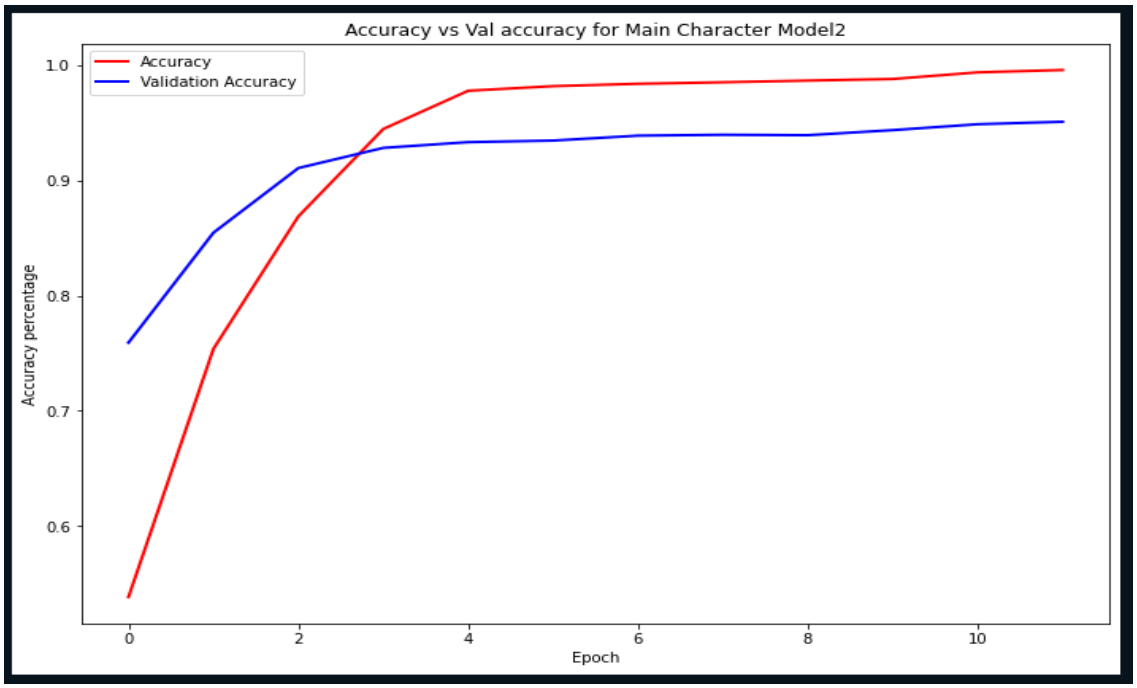


Figure 6- 13: PR curve for main character model -1

Main character model -2



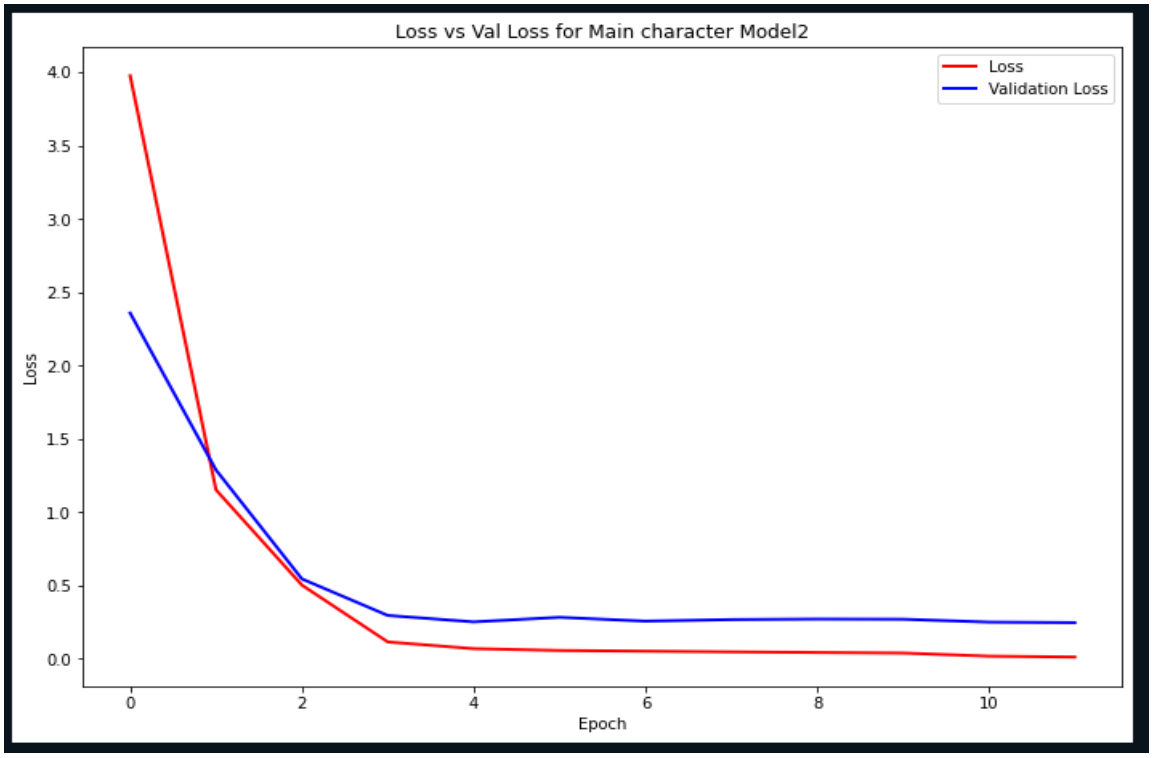


Figure 6- 14: Training accuracy vs validation accuracy and Training loss vs validation loss for main character model -2

Table 6- 3: Normalized confusion matrix for main character model -2

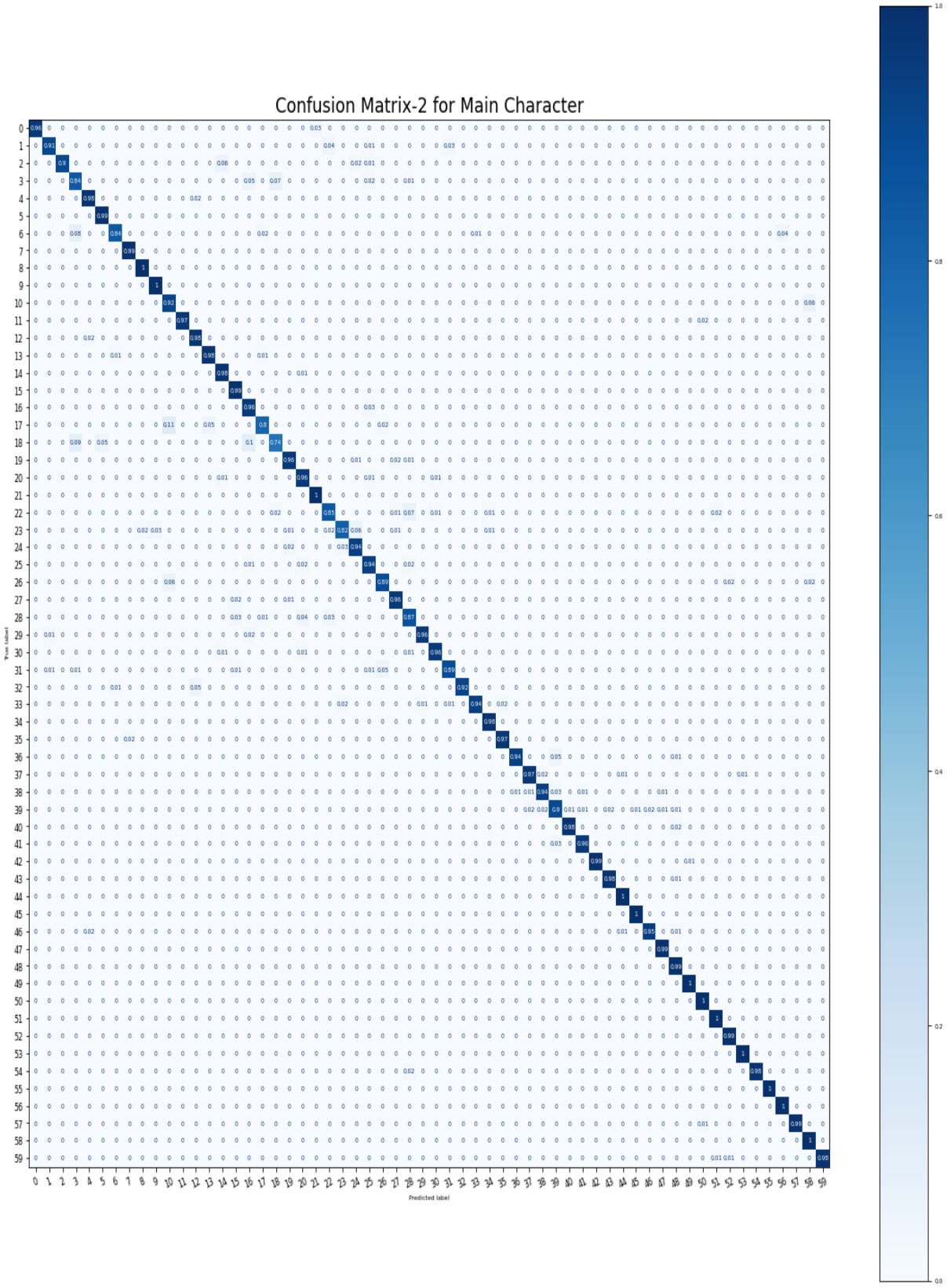


Table 6- 4: Evaluation metrices for main character model-2

	precision	recall	f1-score	support
0	1.00	0.96	0.98	400
1	0.98	0.91	0.94	400
2	0.98	0.90	0.94	400
3	0.81	0.84	0.83	400
4	0.96	0.98	0.97	400
5	0.94	0.99	0.96	400
6	0.97	0.83	0.90	400
7	0.98	0.99	0.99	400
8	0.98	1.00	0.99	400
9	0.96	1.00	0.98	400
10	0.84	0.92	0.88	400
11	0.99	0.97	0.98	400
12	0.94	0.98	0.96	400
13	0.93	0.98	0.96	400
14	0.92	0.98	0.95	400
15	0.93	0.99	0.96	400
16	0.84	0.96	0.90	400
17	0.94	0.80	0.86	400
18	0.89	0.74	0.81	400
19	0.96	0.96	0.96	400
20	0.91	0.95	0.93	400
21	0.97	0.99	0.98	400
22	0.88	0.85	0.86	400
23	0.93	0.82	0.87	400
24	0.91	0.94	0.92	400
25	0.90	0.94	0.92	400
26	0.92	0.89	0.90	400
27	0.95	0.96	0.96	400
28	0.85	0.87	0.86	400
29	0.98	0.96	0.97	400
30	0.97	0.96	0.97	400
31	0.94	0.89	0.92	400
32	0.98	0.92	0.95	400
33	0.98	0.94	0.96	400
34	0.97	0.98	0.98	400
35	0.97	0.97	0.97	400
36	0.98	0.94	0.96	194
37	0.96	0.97	0.97	194
38	0.97	0.94	0.95	196
39	0.90	0.90	0.90	198
40	0.99	0.98	0.99	188
41	0.98	0.96	0.97	202
42	1.00	0.99	0.99	400
43	0.98	0.97	0.98	400
44	0.99	1.00	0.99	400
45	0.98	1.00	0.99	402
46	0.99	0.95	0.97	400
47	0.99	0.99	0.99	400
48	0.96	0.99	0.98	400
49	0.99	1.00	1.00	309
50	0.97	1.00	0.99	400
51	0.96	1.00	0.98	400
52	0.97	0.99	0.98	400
53	0.98	1.00	0.99	400
54	0.98	0.97	0.98	400
55	1.00	1.00	1.00	400
56	0.96	0.99	0.98	400
57	0.99	0.99	0.99	400
58	0.92	1.00	0.96	400
59	0.99	0.98	0.99	400
accuracy			0.95	22683
macro avg	0.95	0.95	0.95	22683
weighted avg	0.95	0.95	0.95	22683

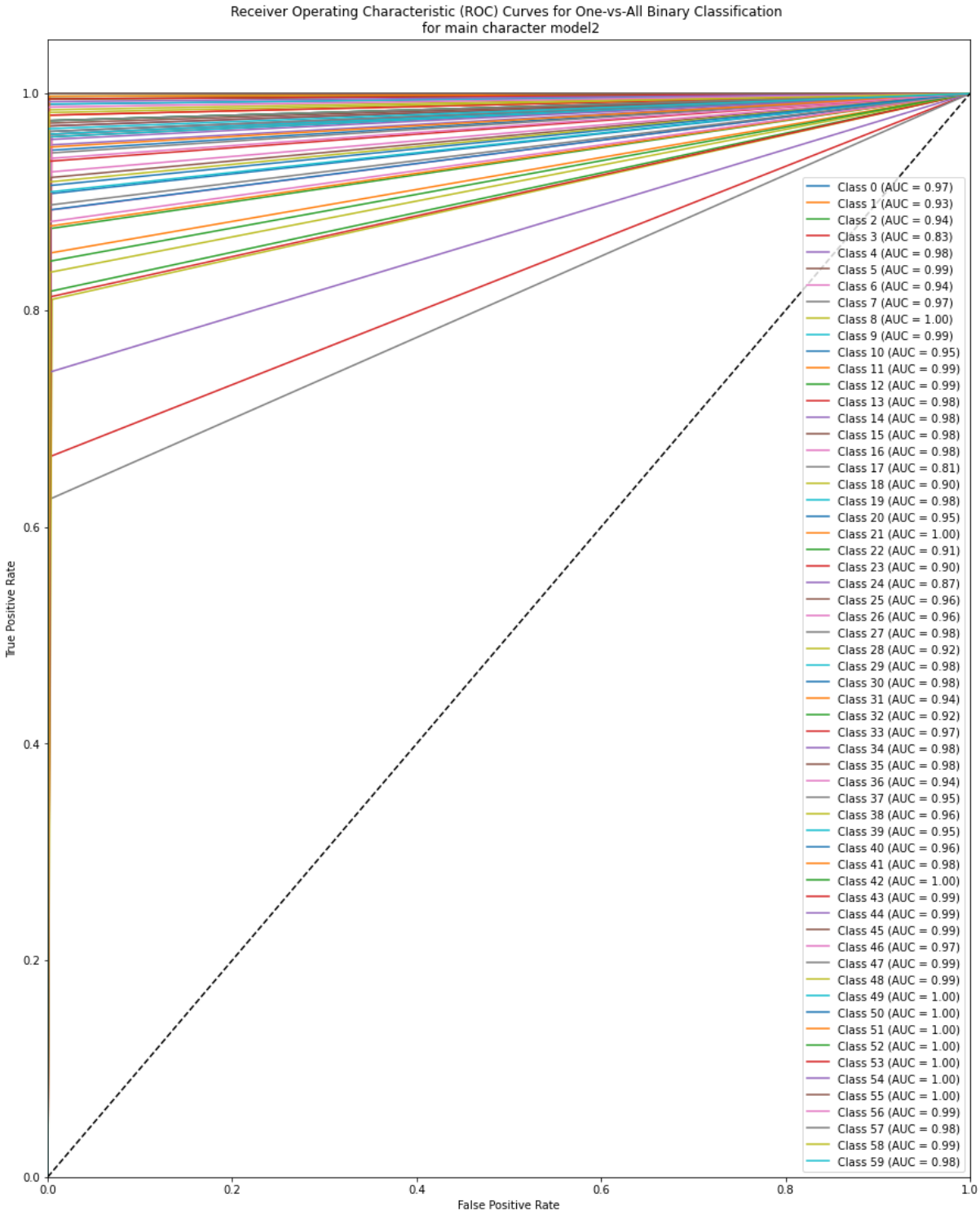


Figure 6- 15: ROC curve for main character model -2

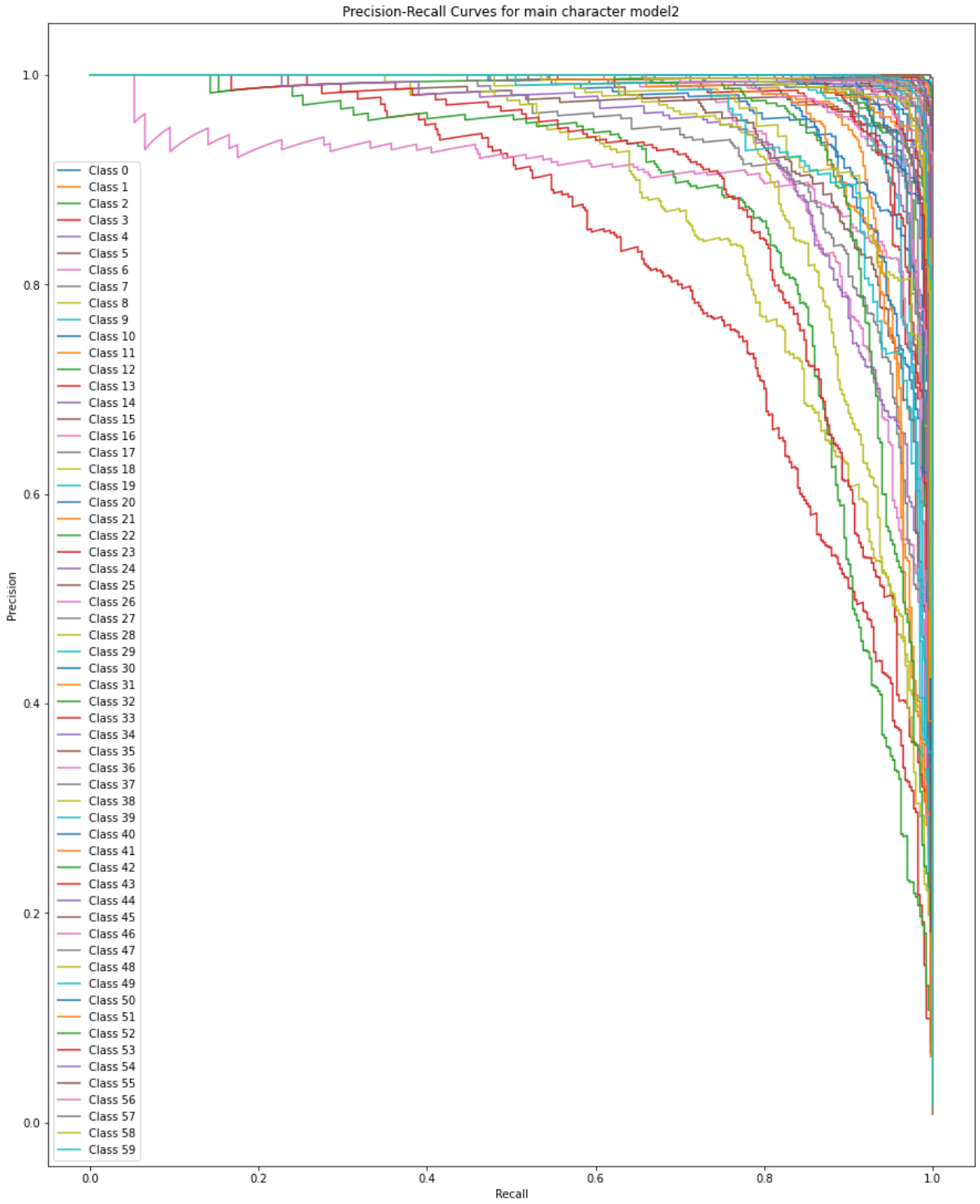
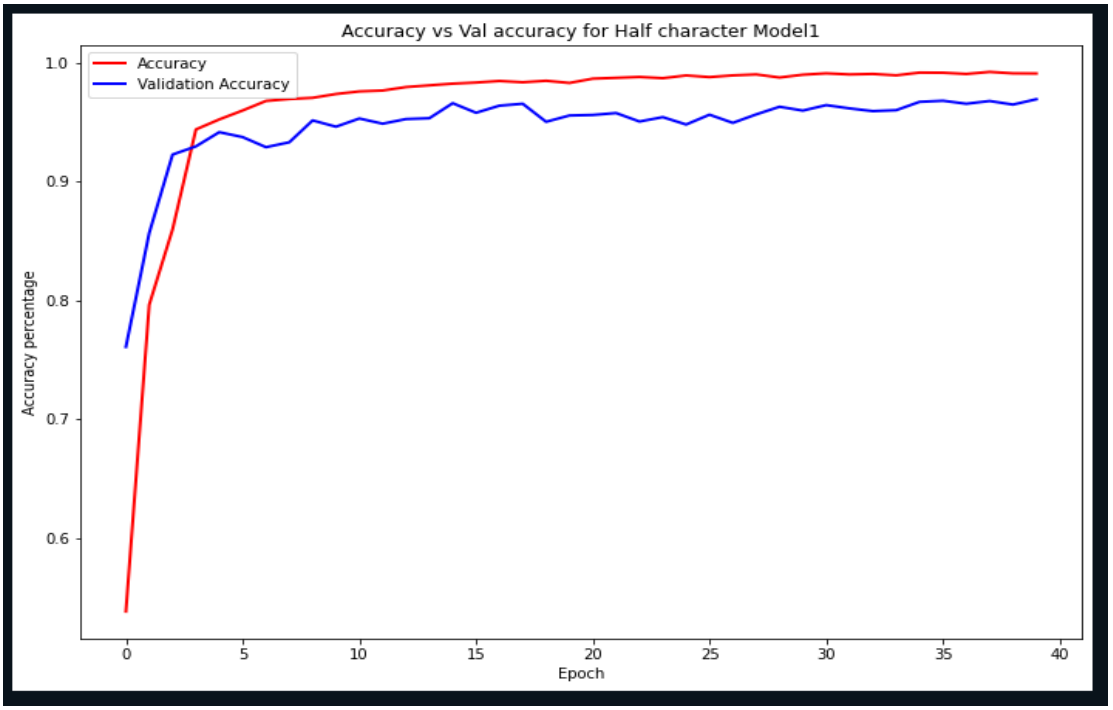


Figure 6- 16: PR curve for main character model -2

Half character model -1



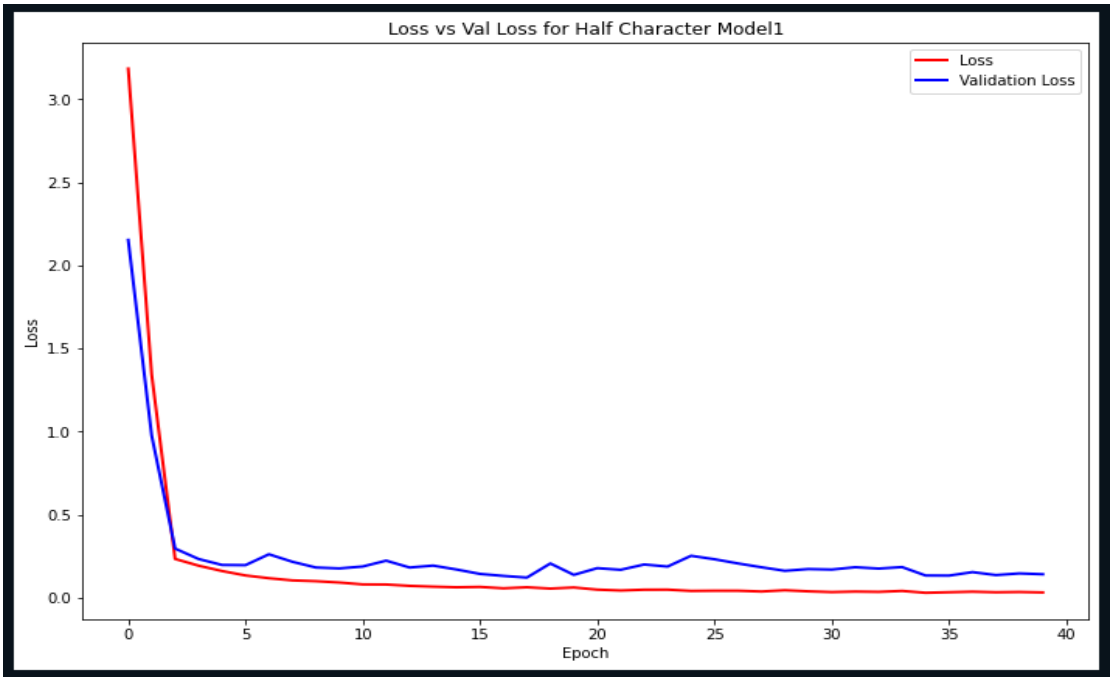


Figure 6- 17: Training accuracy vs validation accuracy and Training loss vs validation loss for half character model -1

Table 6- 5: Normalized confusion matrix for half character model -1

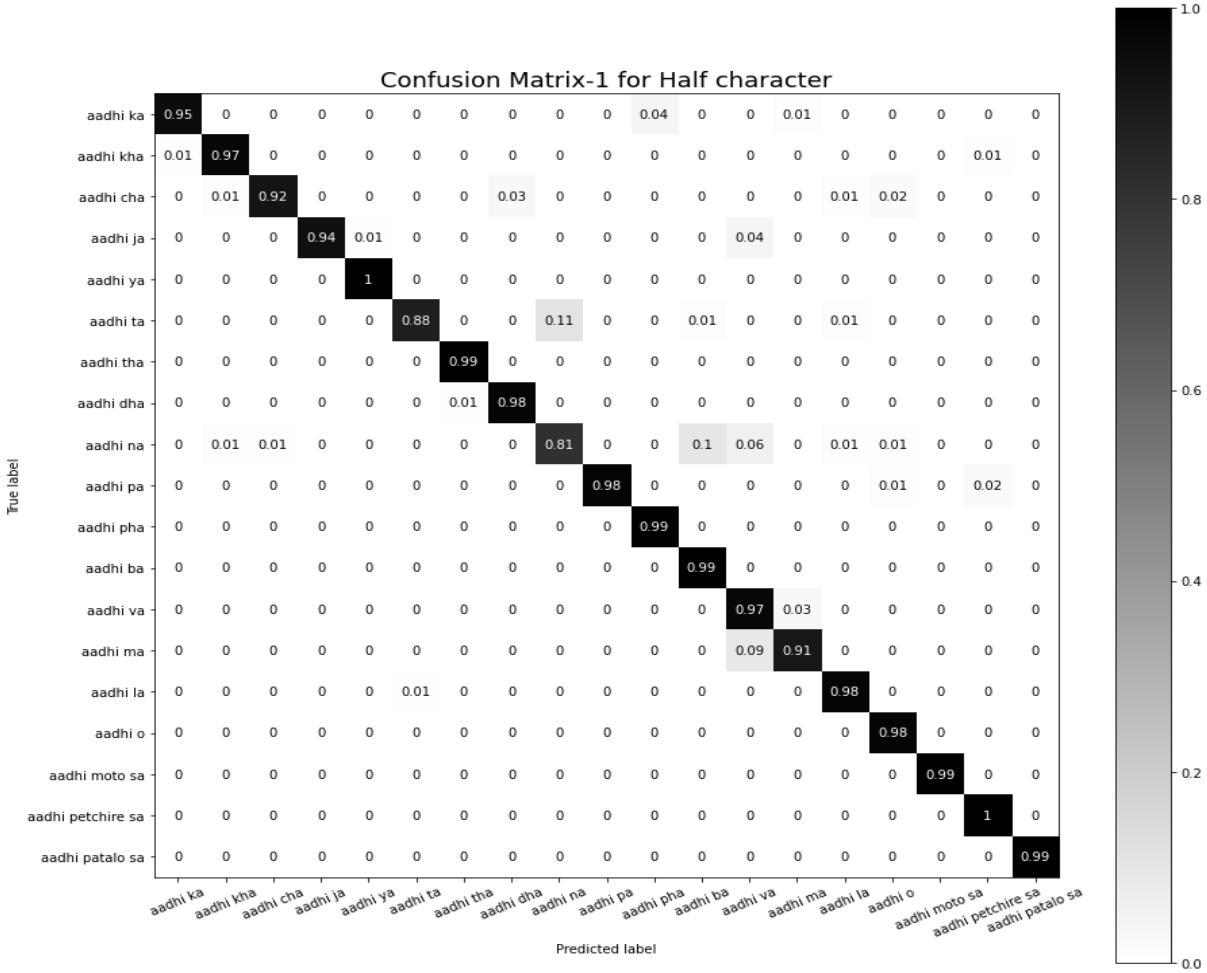


Table 6- 6: Evaluation metrices for half character model -1

	precision	recall	f1-score	support
0	0.98	0.95	0.96	199
1	0.98	0.97	0.98	208
2	0.98	0.92	0.95	156
3	1.00	0.94	0.97	201
4	0.99	1.00	0.99	199
5	0.97	0.88	0.93	171
6	0.98	0.99	0.98	210
7	0.97	0.98	0.97	227
8	0.85	0.81	0.83	145
9	0.98	0.98	0.98	190
10	0.97	0.99	0.98	255
11	0.92	0.99	0.95	239
12	0.89	0.97	0.93	311
13	0.95	0.91	0.93	237
14	0.97	0.98	0.98	262
15	0.98	0.98	0.98	329
16	1.00	0.99	0.99	271
17	0.98	1.00	0.99	296
18	1.00	0.99	1.00	244
accuracy			0.97	4350
macro avg	0.97	0.96	0.96	4350
weighted avg	0.97	0.97	0.97	4350

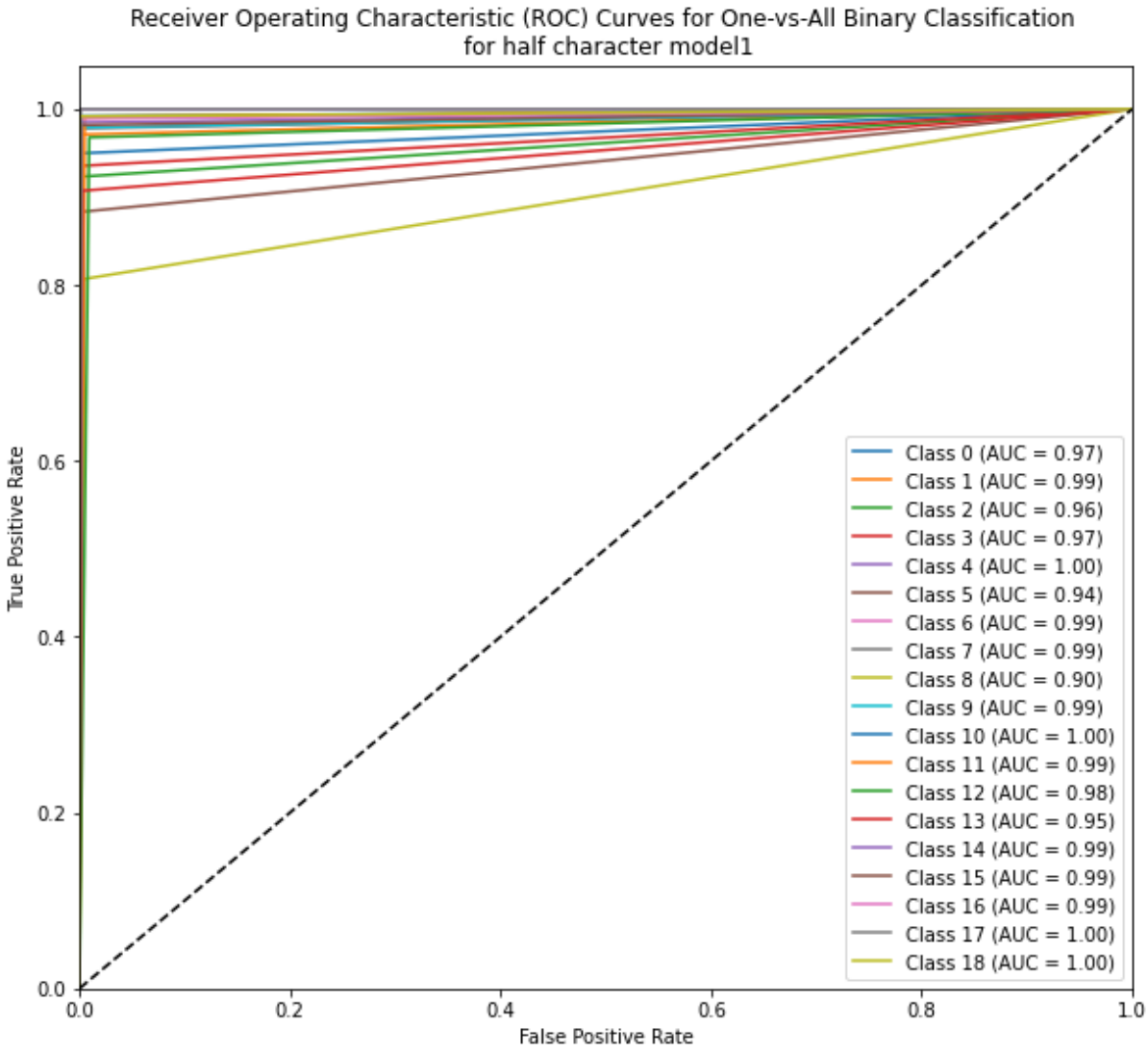


Figure 6- 18: ROC curve for half character model -1

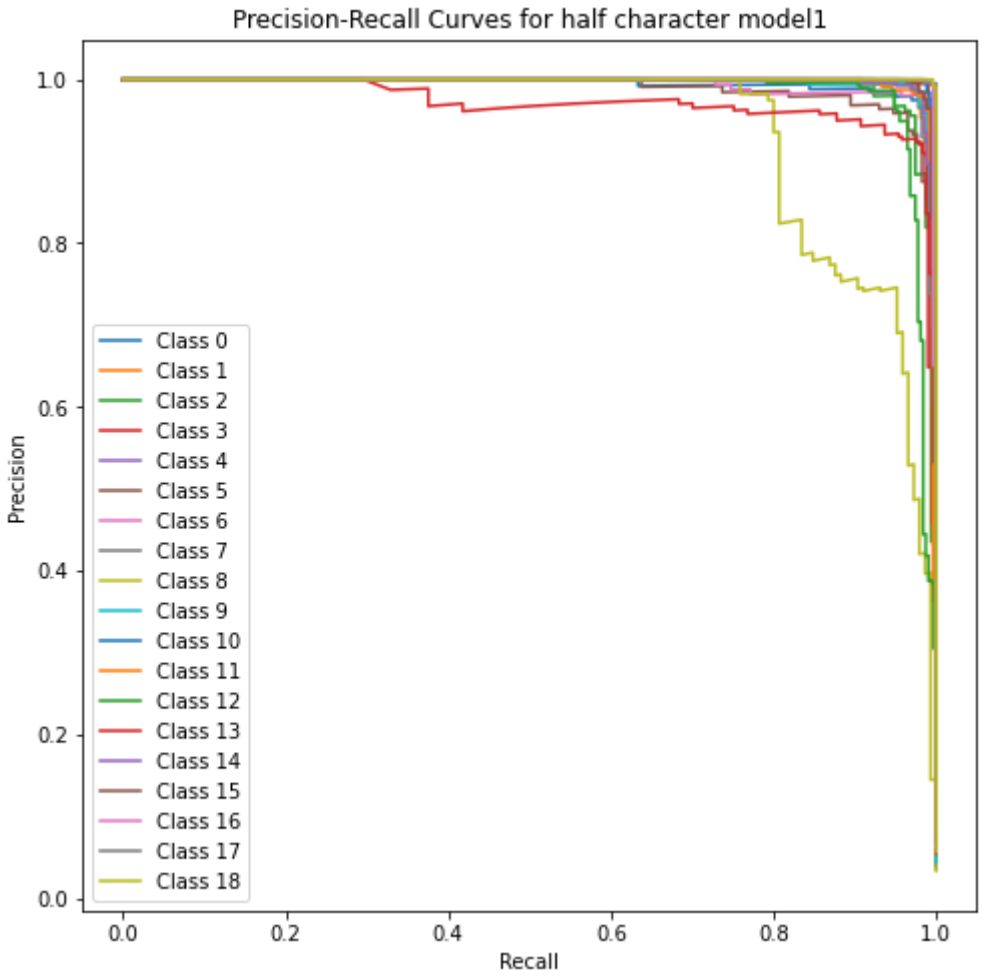
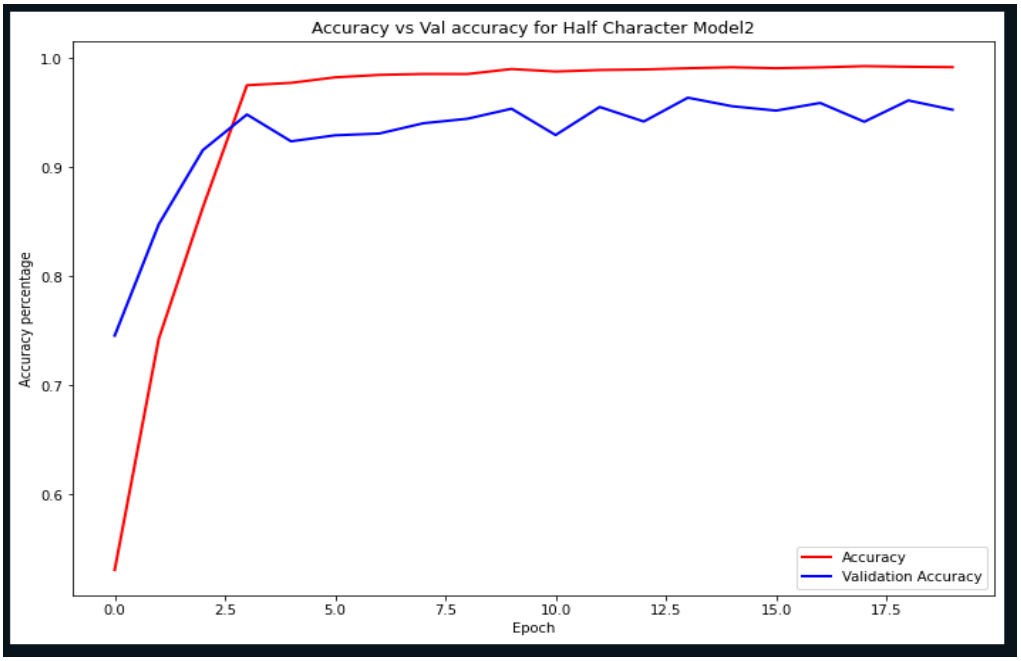


Figure 6- 19: PR curve for half character model -1

Half character model-2



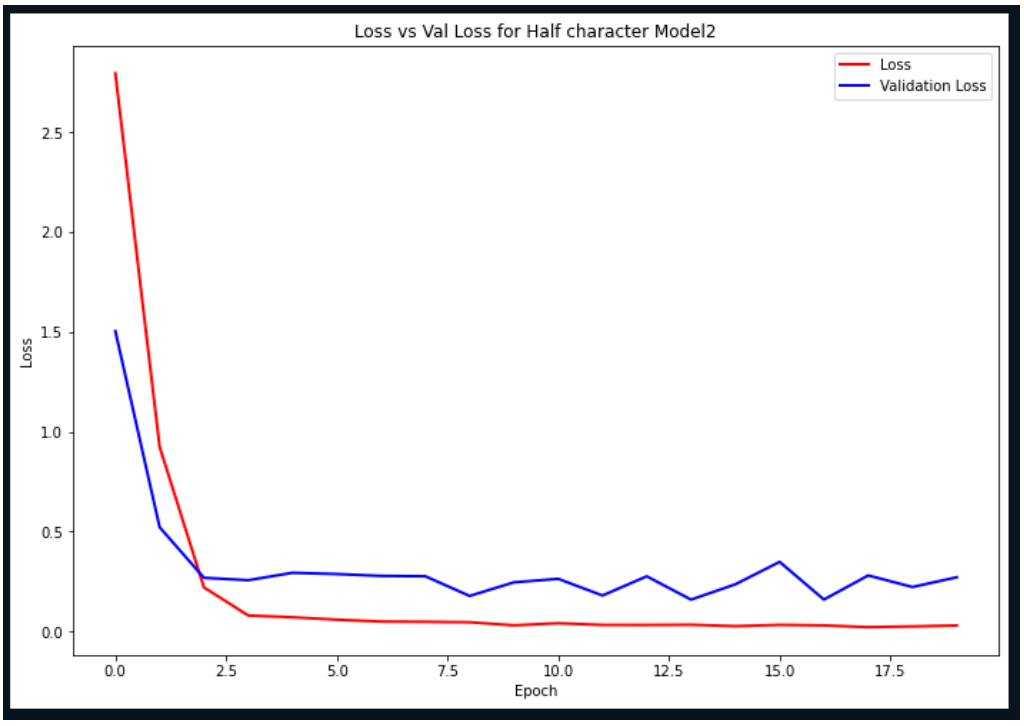


Figure 6- 20: Training accuracy vs validation accuracy and Training loss vs validation loss for half character model -2

Table 6- 7: Normalized confusion matrix for half character model -2

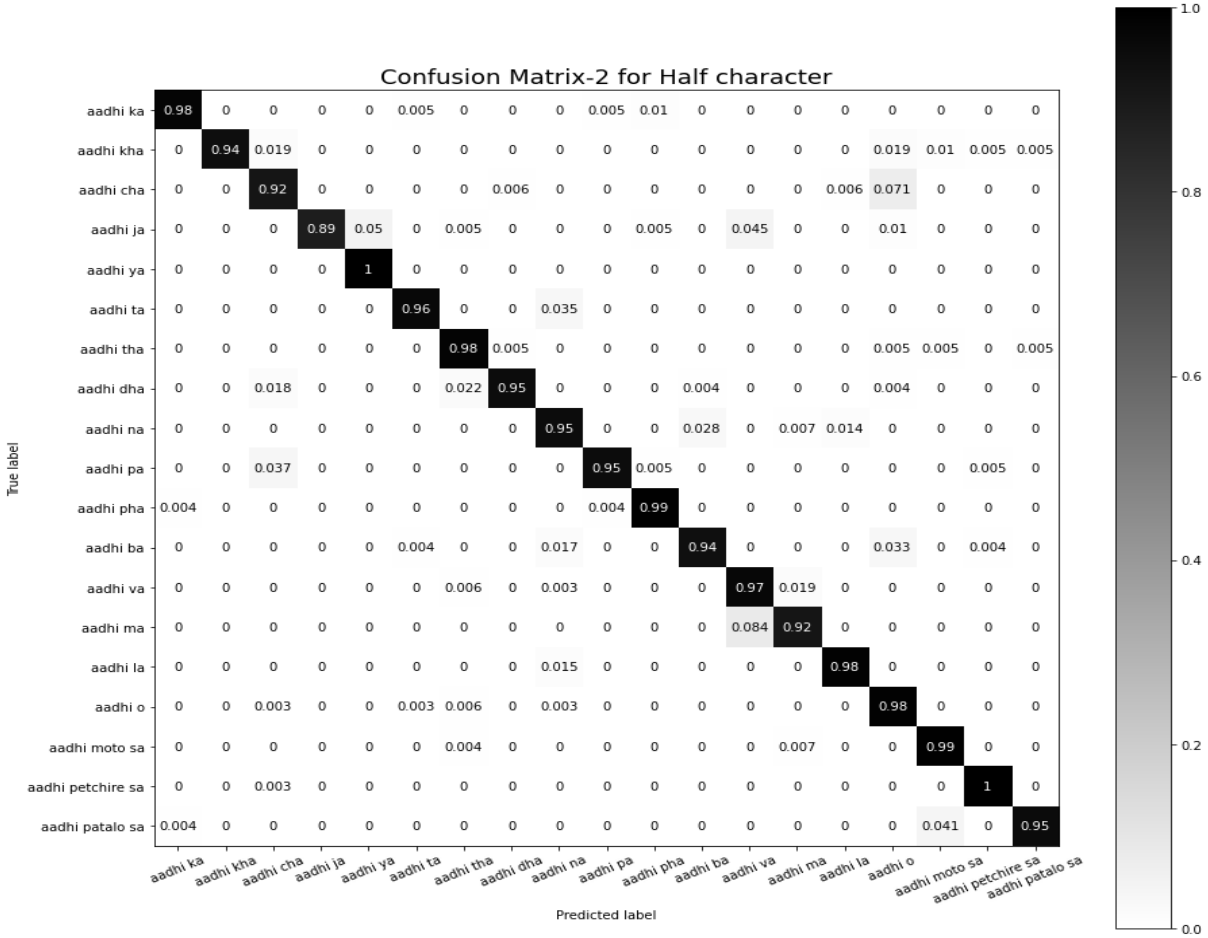


Table 6- 8: Evaluation metrices for half character model -2

	precision	recall	f1-score	support
0	0.99	0.98	0.98	199
1	1.00	0.94	0.97	208
2	0.89	0.92	0.91	156
3	1.00	0.89	0.94	201
4	0.95	1.00	0.98	199
5	0.98	0.96	0.97	171
6	0.95	0.98	0.96	210
7	0.99	0.95	0.97	227
8	0.90	0.95	0.92	145
9	0.99	0.95	0.97	190
10	0.98	0.99	0.99	255
11	0.98	0.94	0.96	239
12	0.91	0.97	0.94	311
13	0.96	0.92	0.94	237
14	0.99	0.98	0.99	262
15	0.92	0.98	0.95	329
16	0.95	0.99	0.97	271
17	0.99	1.00	0.99	296
18	0.99	0.95	0.97	244
accuracy			0.96	4350
macro avg	0.96	0.96	0.96	4350
weighted avg	0.96	0.96	0.96	4350

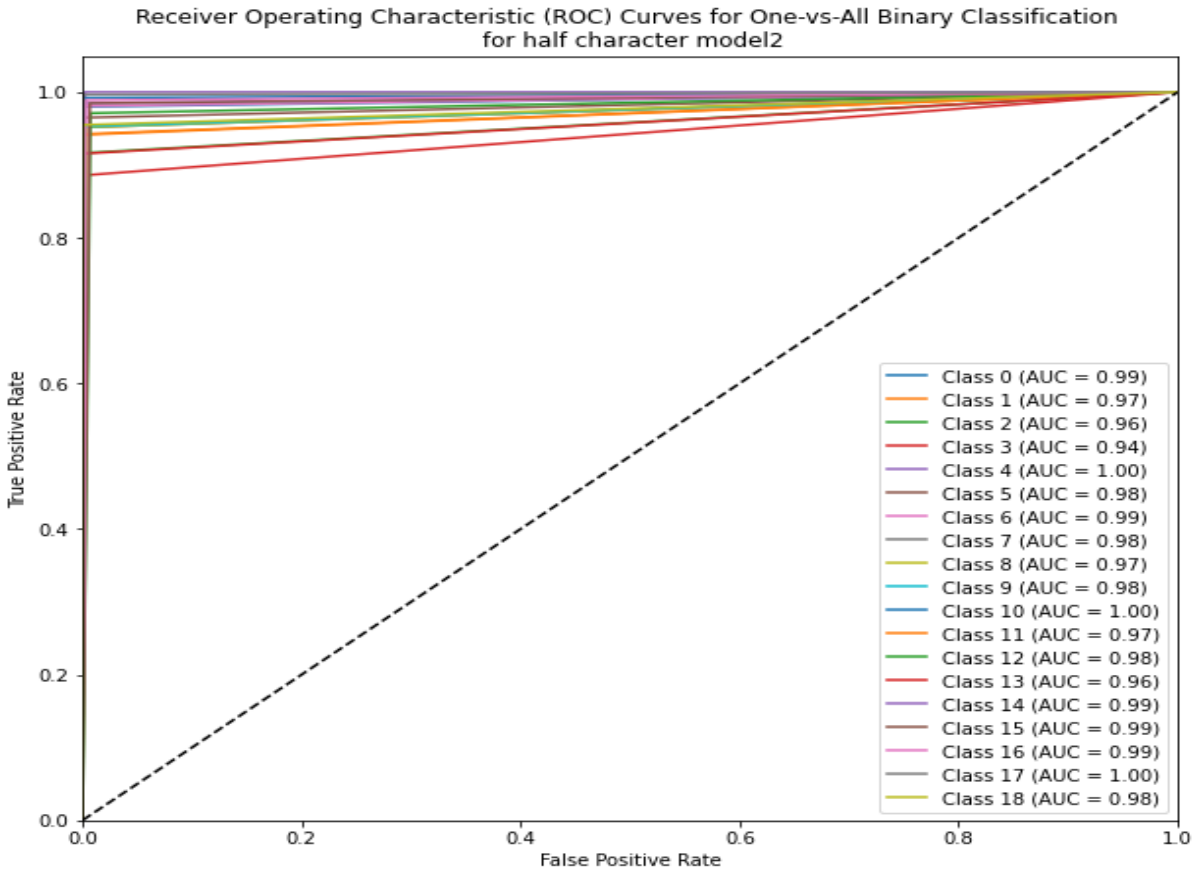


Figure 6- 21: ROC curve for half character model -2

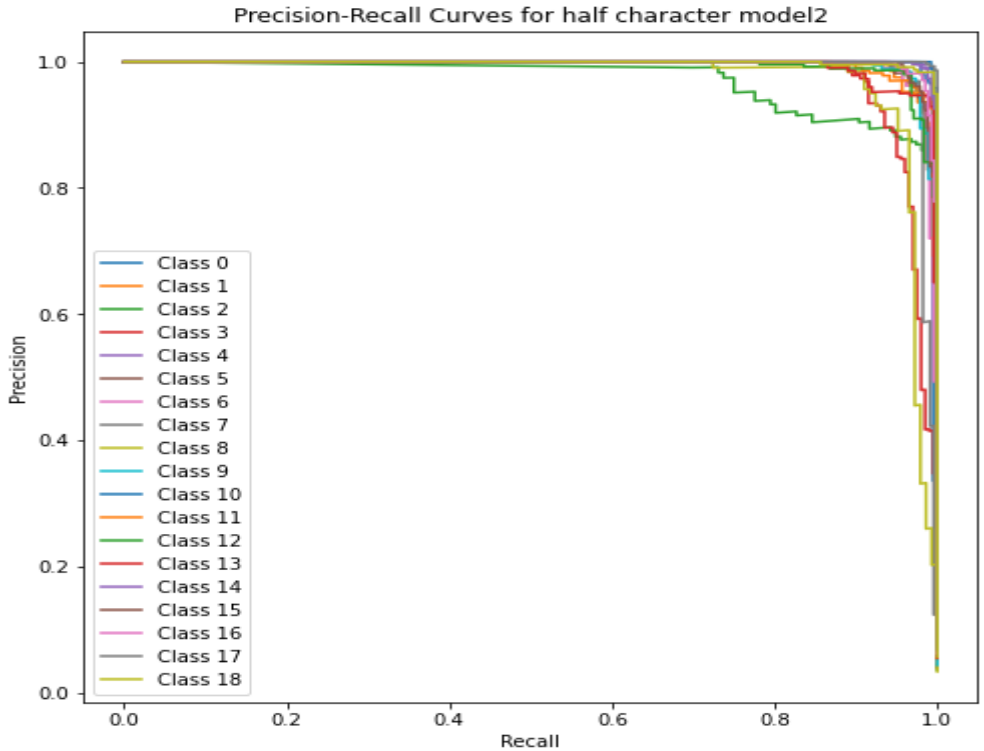
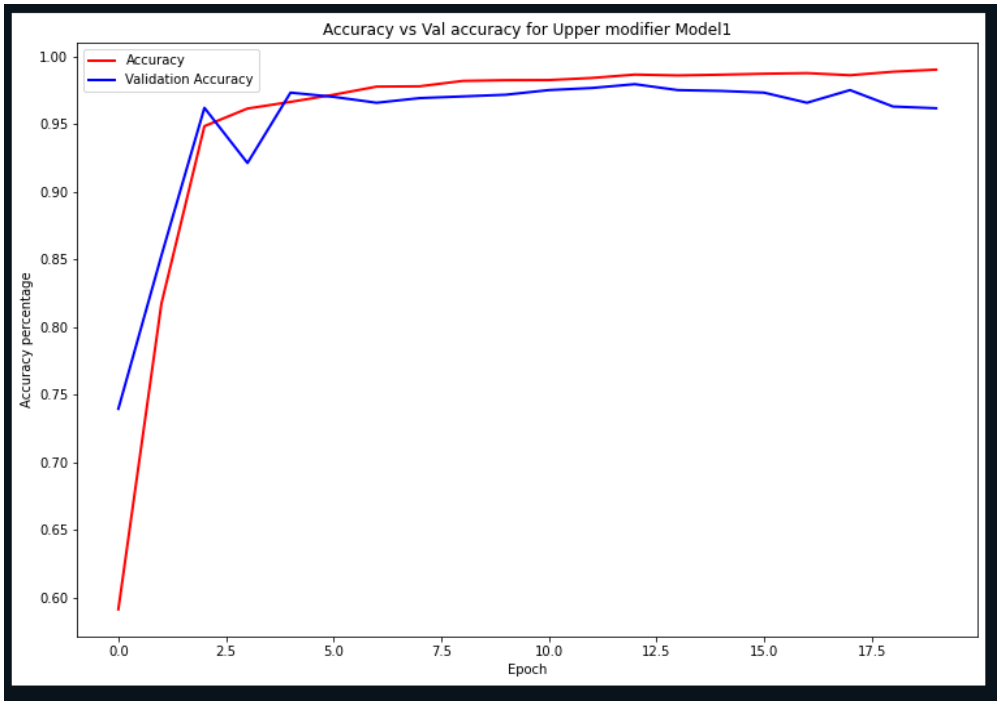


Figure 6- 22: PR curve for half character model -2

Upper modifier model -1



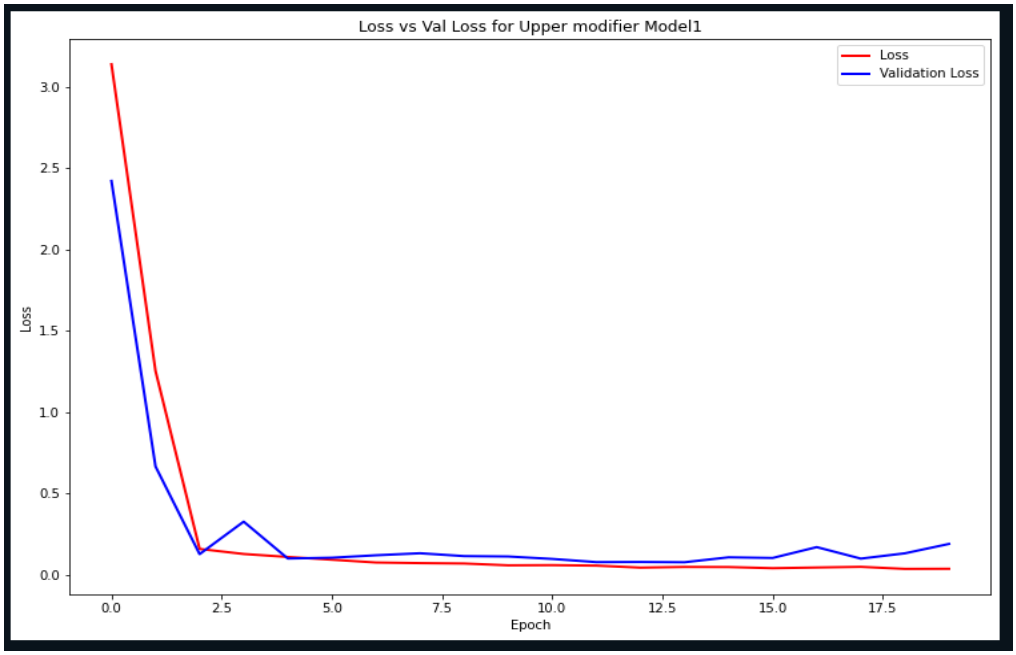


Figure 6- 23: Training accuracy vs validation accuracy and Training loss vs validation loss for upper modifier model -1

Table 6- 9: Confusion matrix for upper modifier model -1

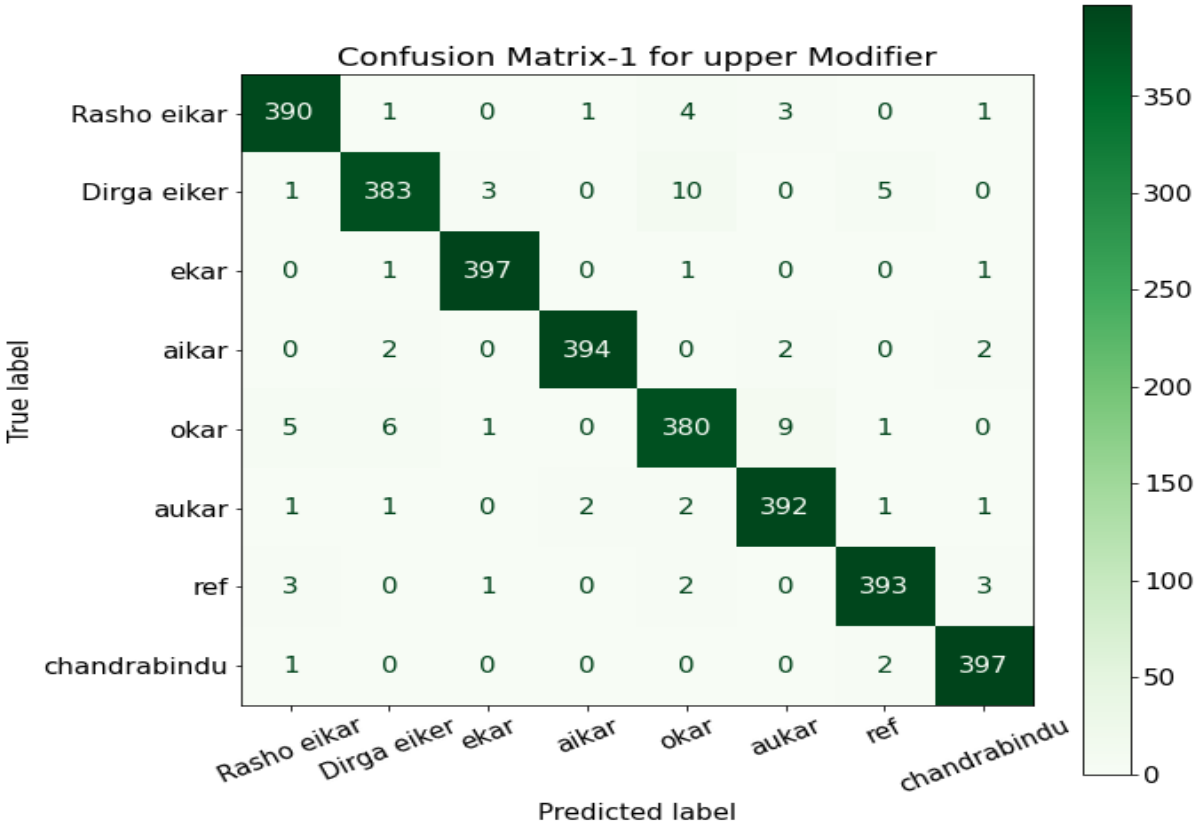


Table 6- 10: Normalized confusion matrix for upper modifier model -1

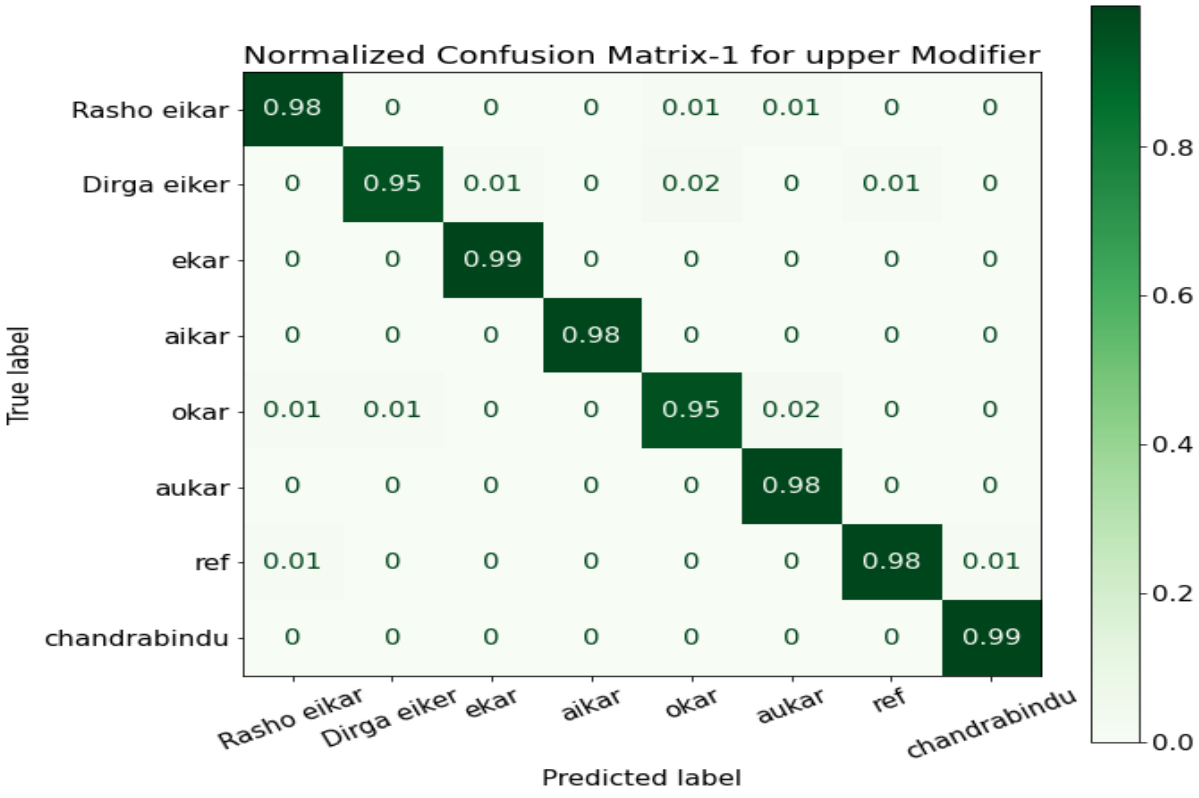


Table 6- 11: Evaluation metrices for upper modifier model -1

	precision	recall	f1-score	support
0	0.97	0.97	0.97	400
1	0.97	0.95	0.96	402
2	0.99	0.99	0.99	400
3	0.99	0.98	0.99	400
4	0.95	0.95	0.95	402
5	0.97	0.98	0.97	400
6	0.98	0.98	0.98	402
7	0.98	0.99	0.99	400
accuracy			0.98	3206
macro avg	0.98	0.98	0.98	3206
weighted avg	0.98	0.98	0.98	3206

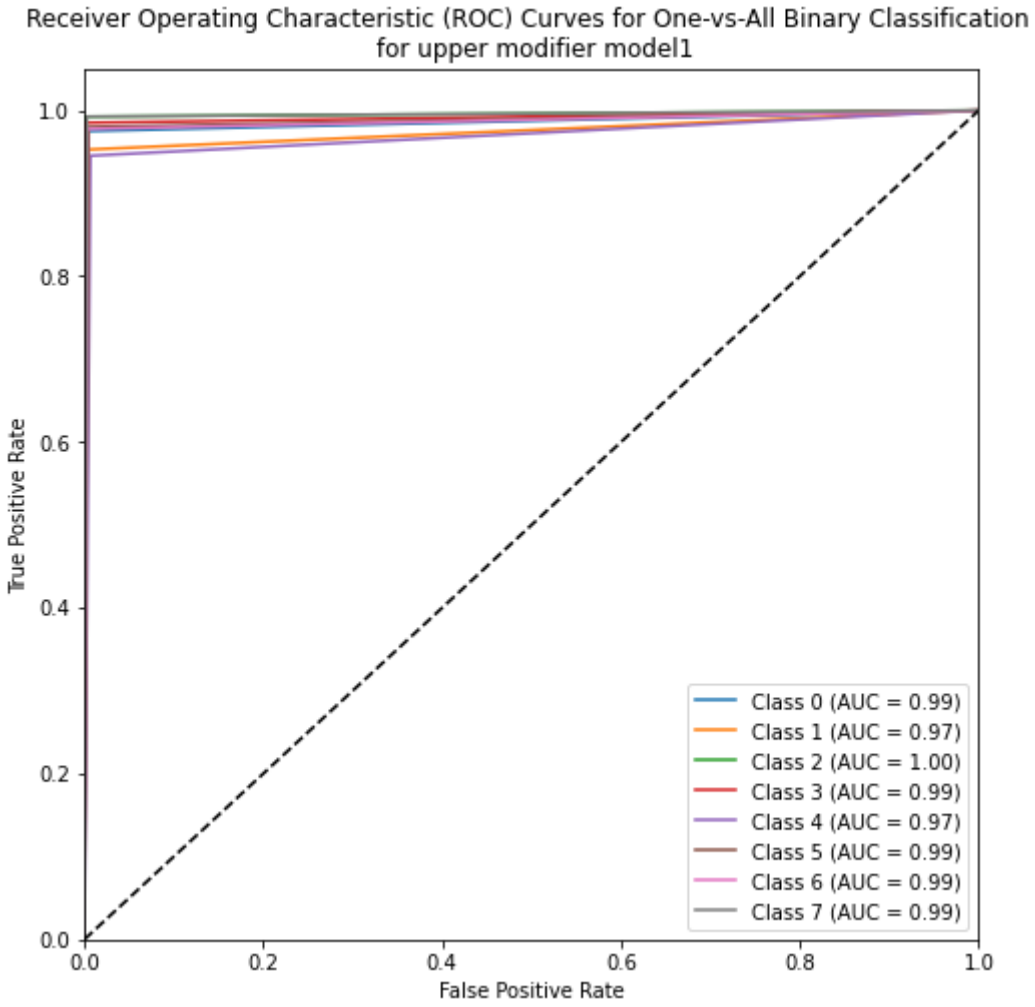


Figure 6- 24: ROC curve for upper modifier model -1

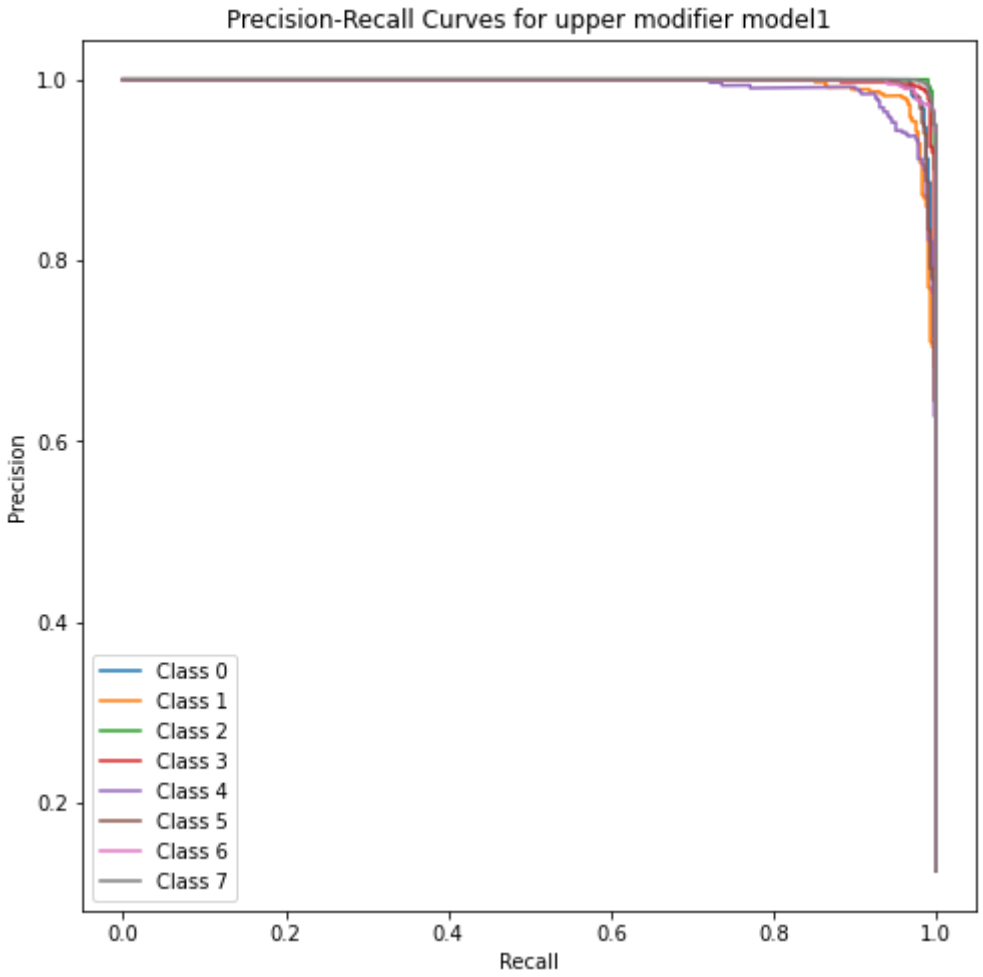
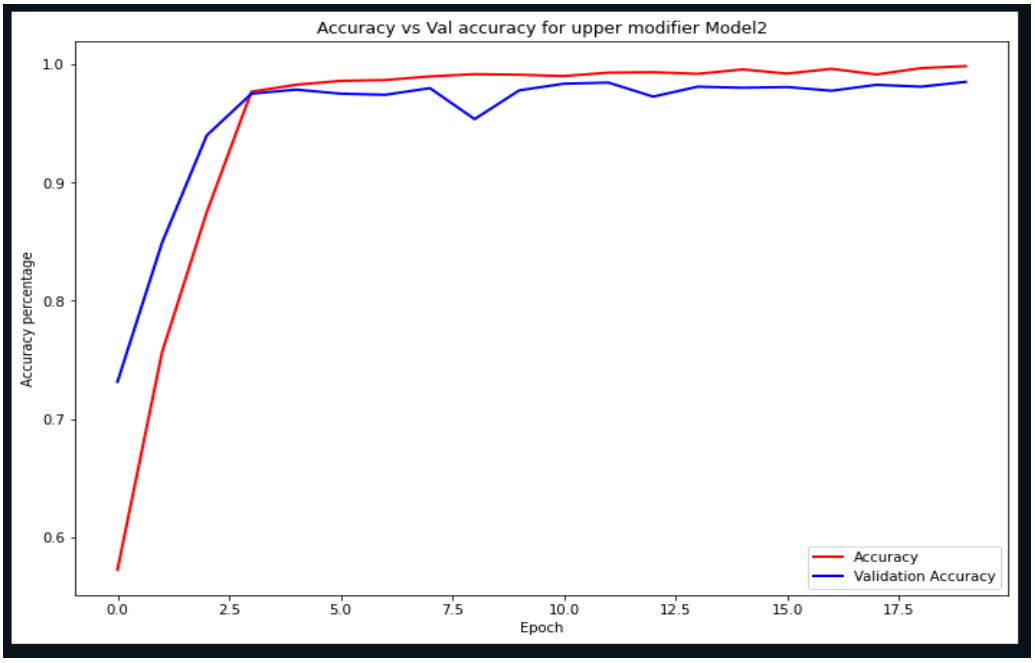


Figure 6- 25: PR curve for upper modifier model -1

Upper modifier model -2



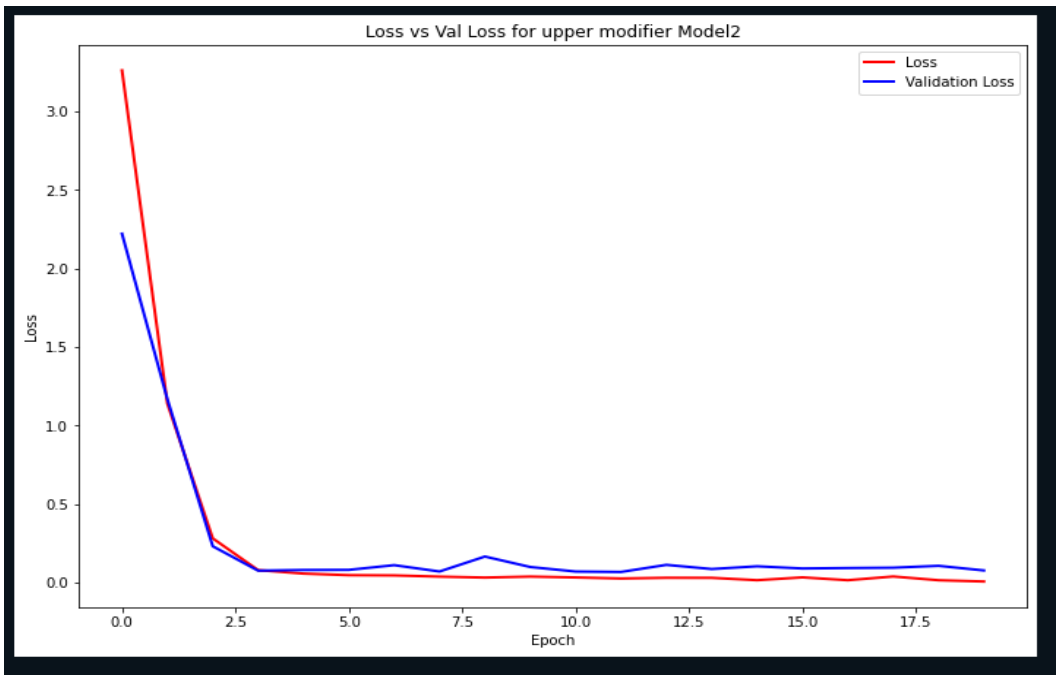


Figure 6- 26: Training accuracy vs validation accuracy and Training loss vs validation loss for upper modifier model -2

Table 6- 12:Confusion matrix for upper modifier model -2

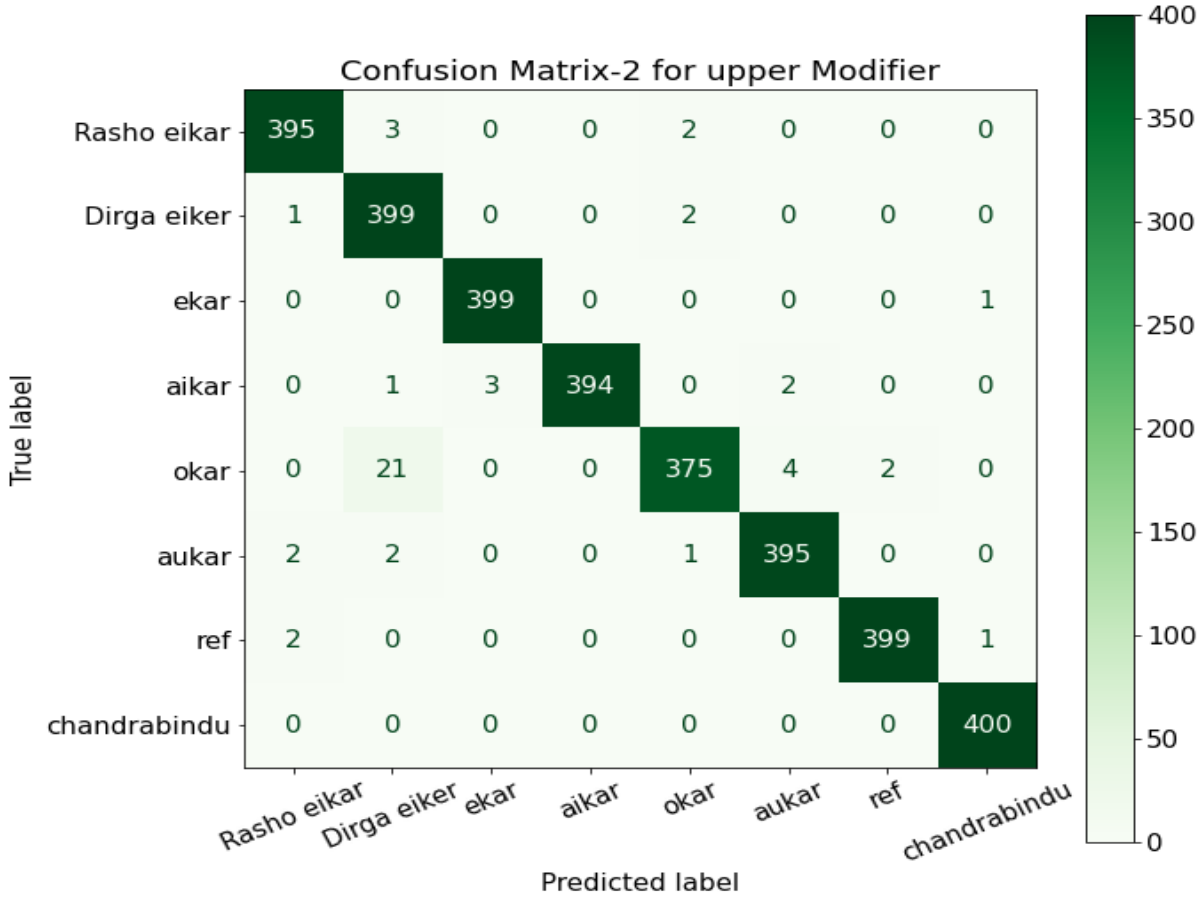


Table 6- 13: Normalized confusion matrix for upper modifier model -2

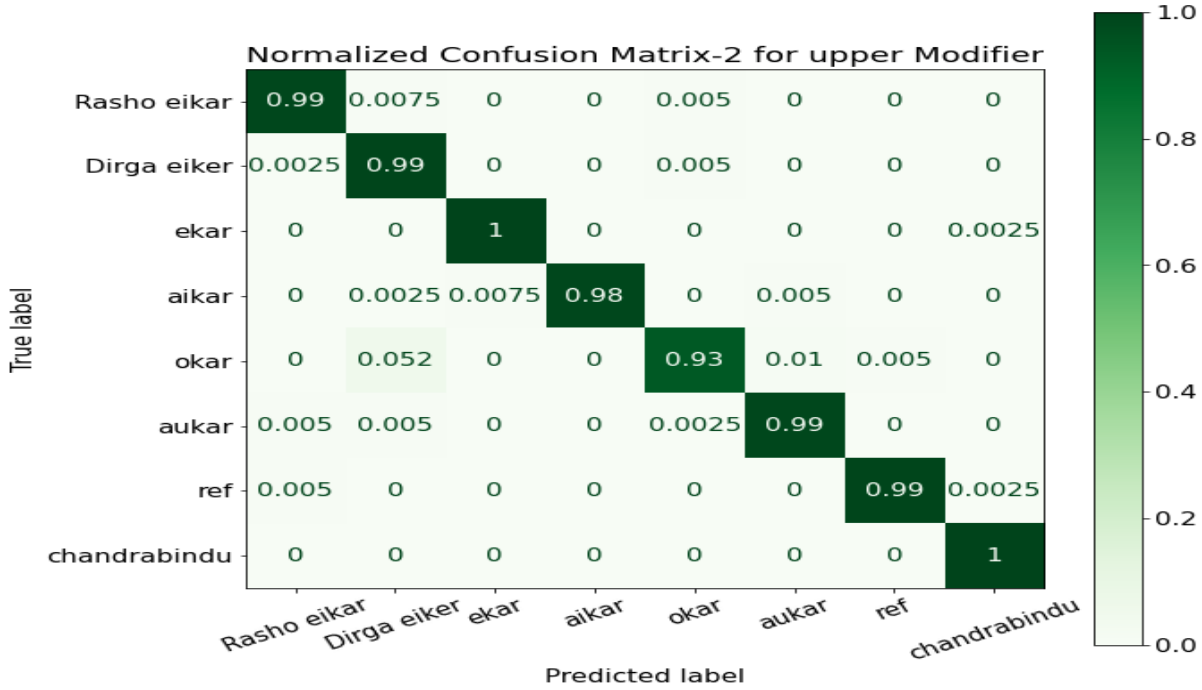


Table 6- 14: Evaluation metrices for upper modifier model -2

	precision	recall	f1-score	support
0	0.99	0.99	0.99	400
1	0.94	0.99	0.96	402
2	0.99	1.00	1.00	400
3	1.00	0.98	0.99	400
4	0.99	0.93	0.96	402
5	0.99	0.99	0.99	400
6	1.00	0.99	0.99	402
7	1.00	1.00	1.00	400
accuracy			0.98	3206
macro avg	0.98	0.98	0.98	3206
weighted avg	0.98	0.98	0.98	3206

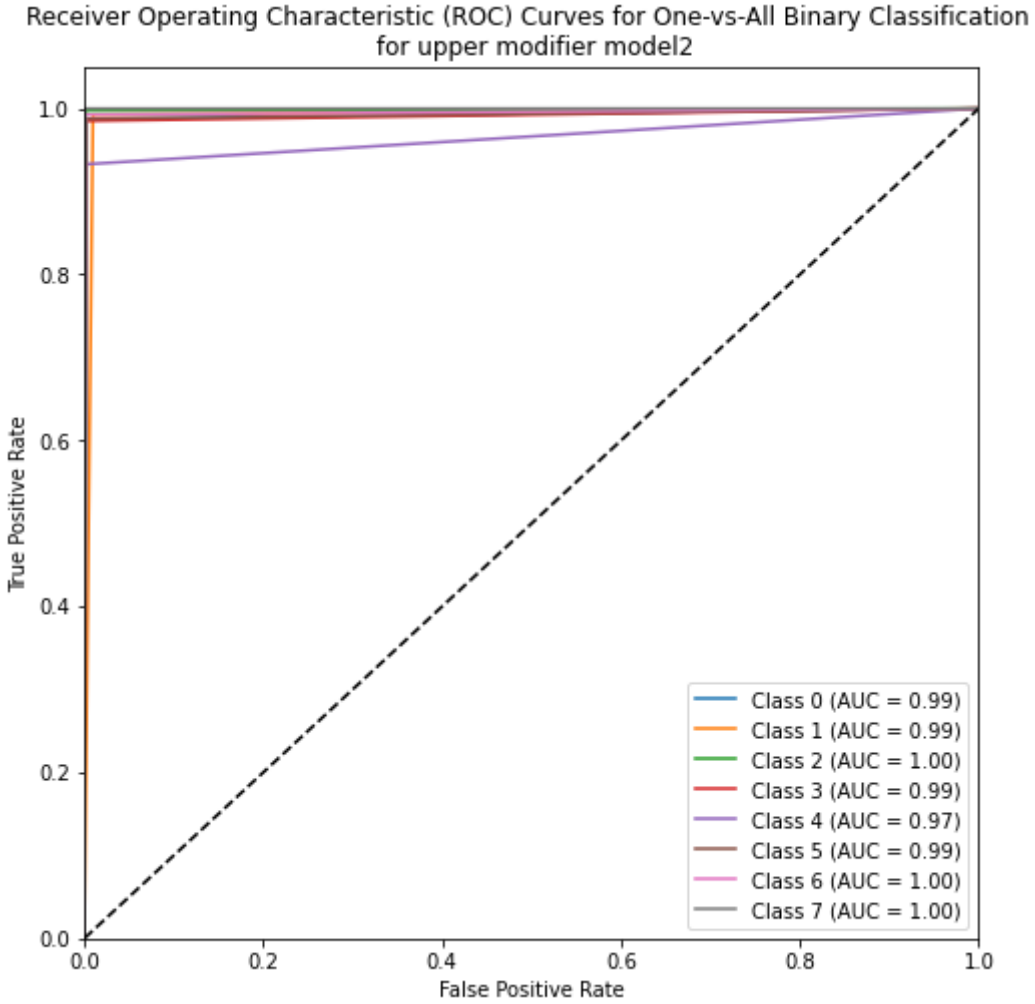


Figure 6- 27: ROC curve for upper modifier model -2

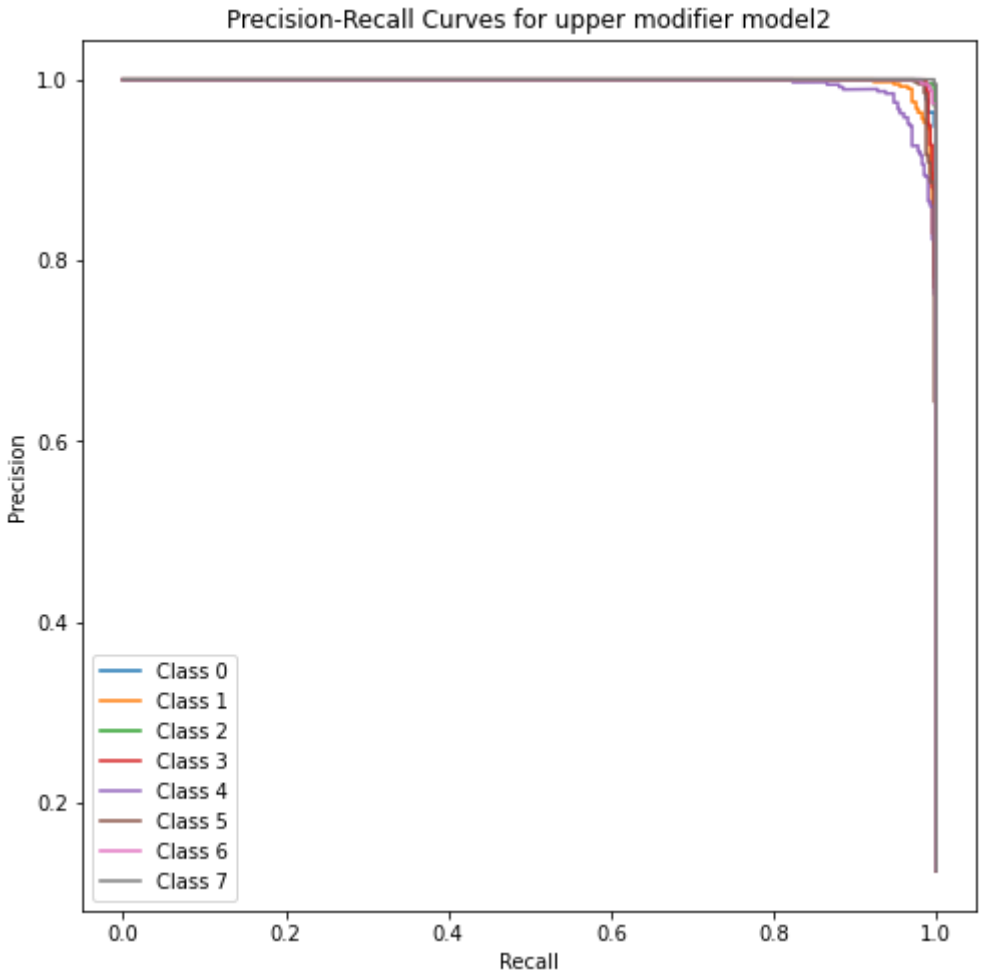
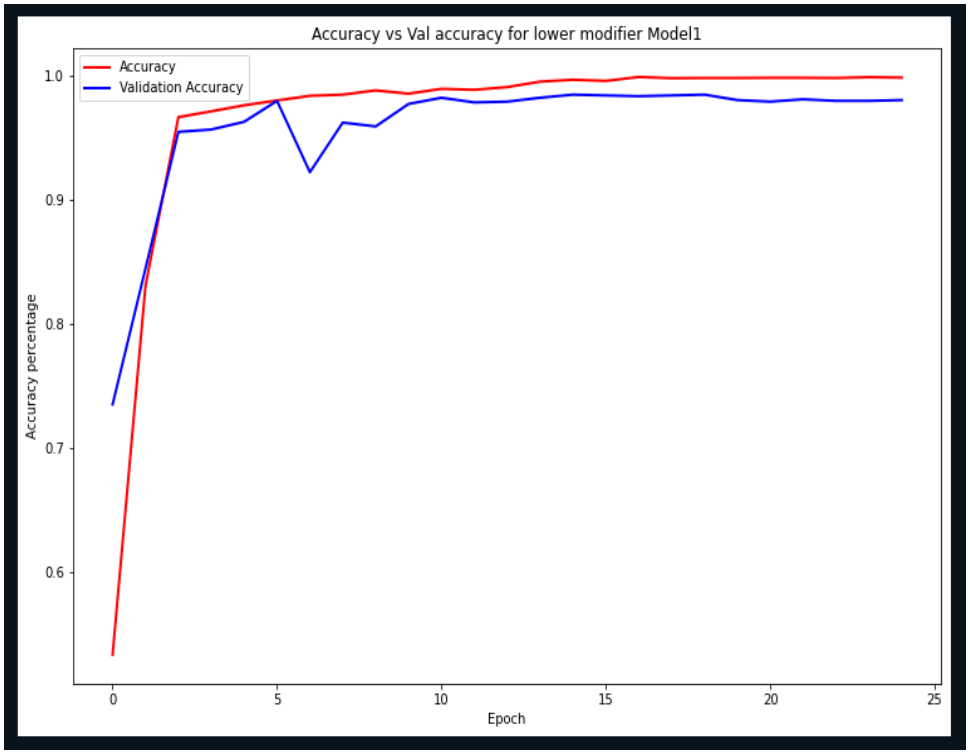


Figure 6- 28: PR curve for upper modifier model -2

Lower modifier model -1



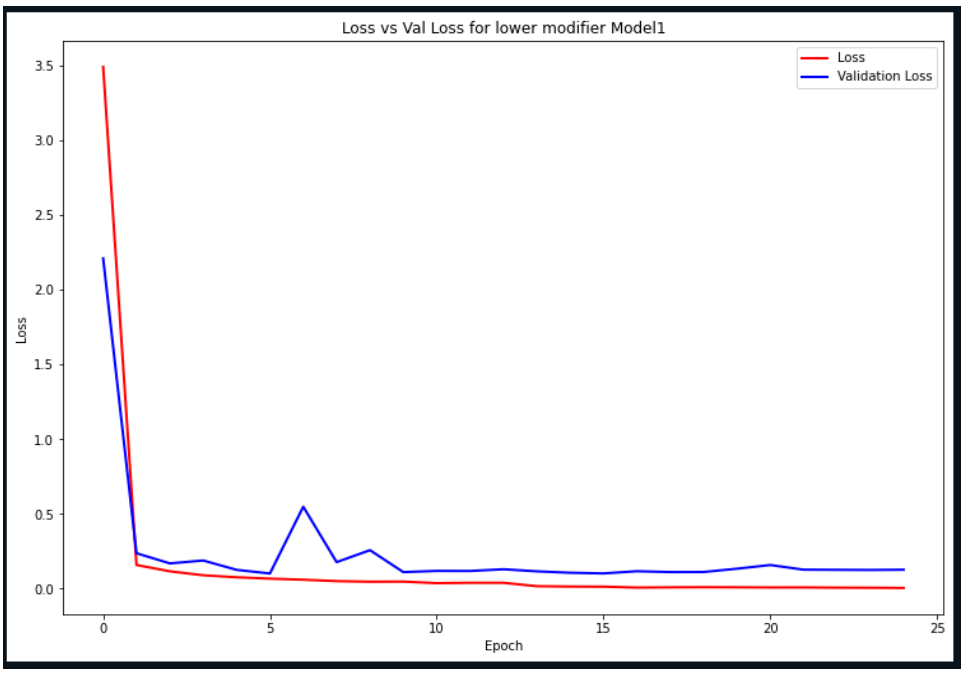


Figure 6- 29: Training accuracy vs validation accuracy and Training loss vs validation loss for lower modifier model -1

Table 6- 15: Confusion matrix for lower modifier model -1

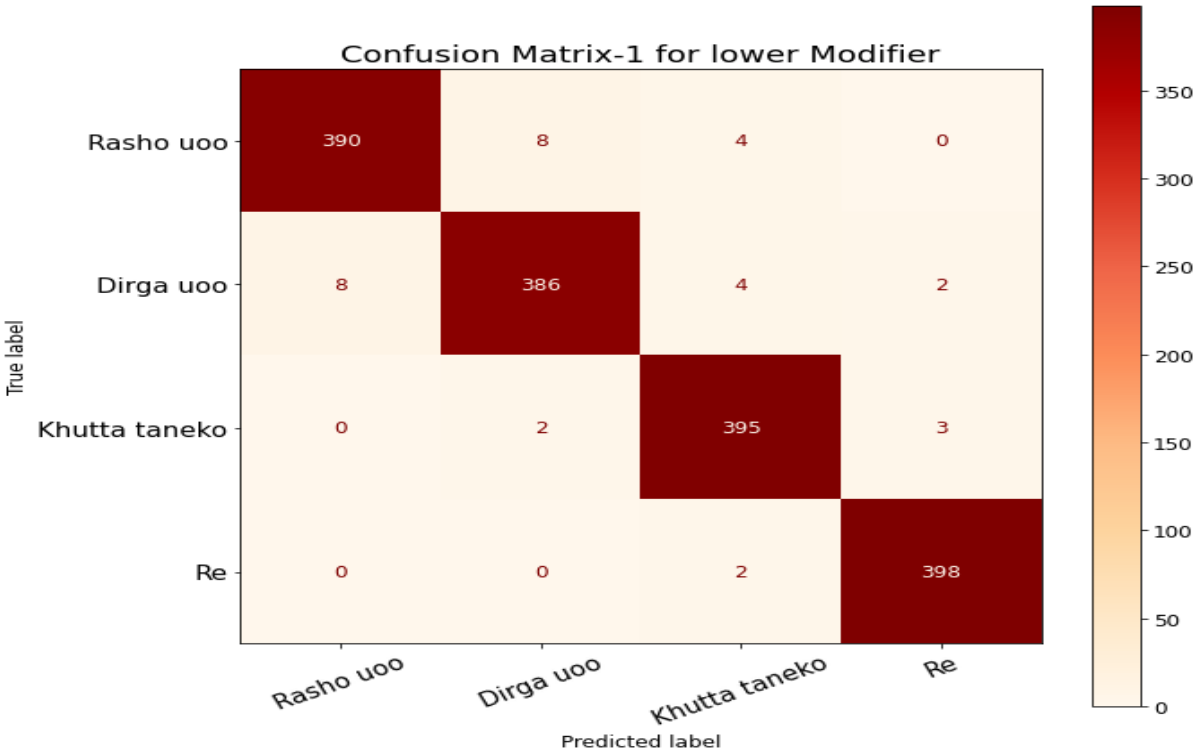


Table 6- 16: Normalized confusion matrix for lower modifier model -1

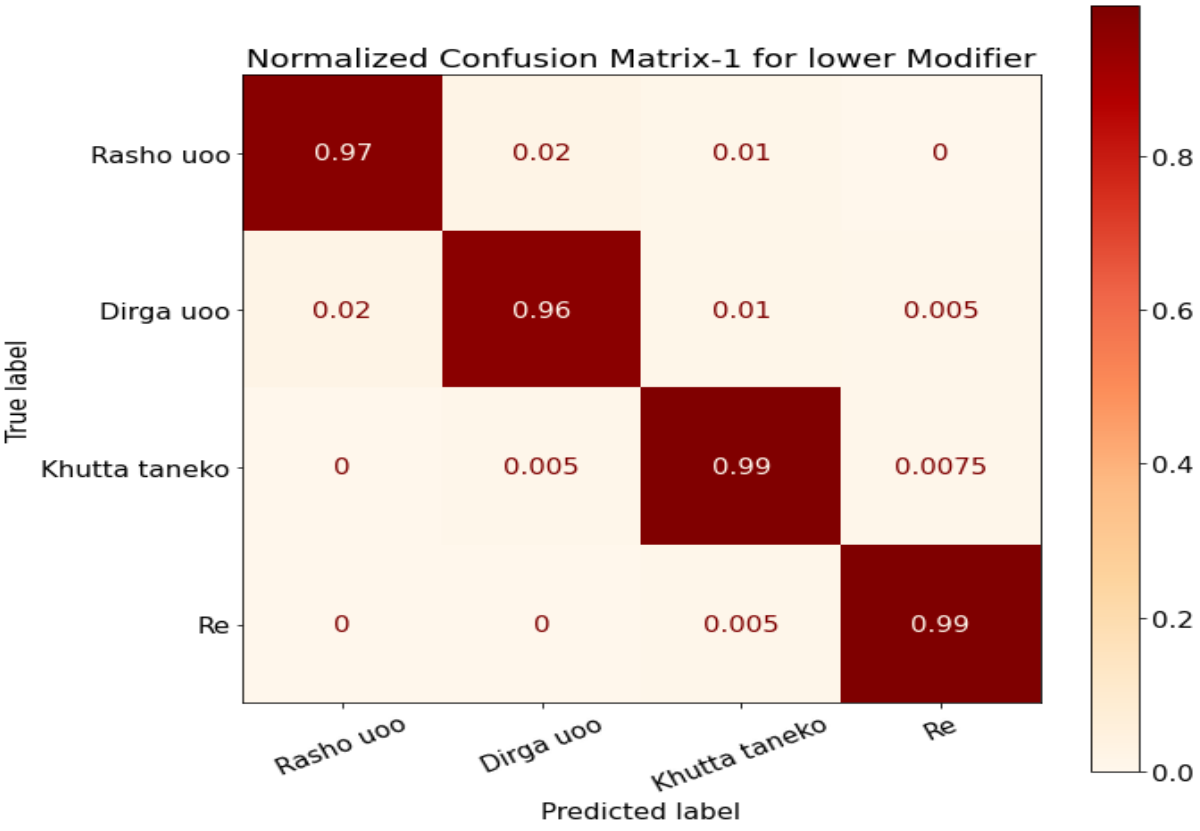


Table 6- 17: Evaluation metrices for lower modifier model -1

	precision	recall	f1-score	support
0	0.98	0.97	0.97	402
1	0.97	0.96	0.97	400
2	0.98	0.99	0.98	400
3	0.99	0.99	0.99	400
accuracy			0.98	1602
macro avg	0.98	0.98	0.98	1602
weighted avg	0.98	0.98	0.98	1602

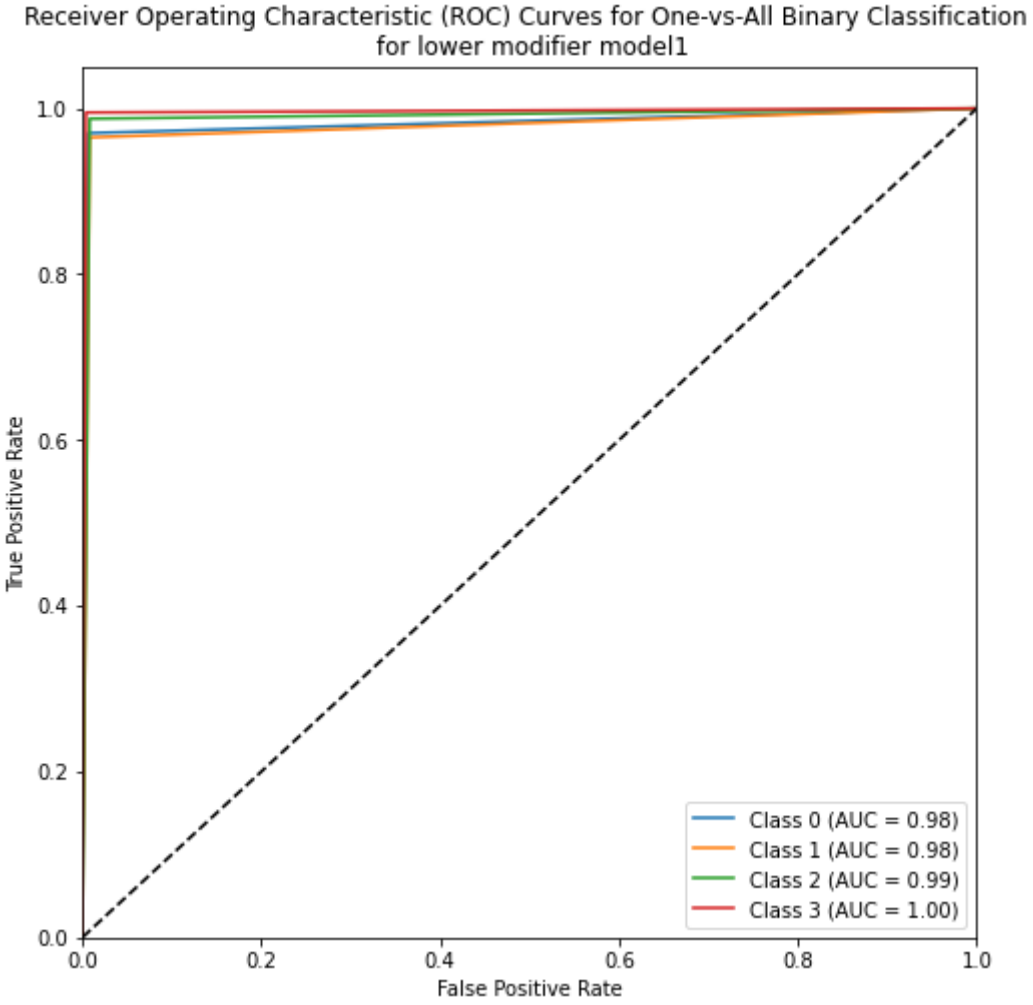


Figure 6- 30: ROC curve for lower modifier model -1

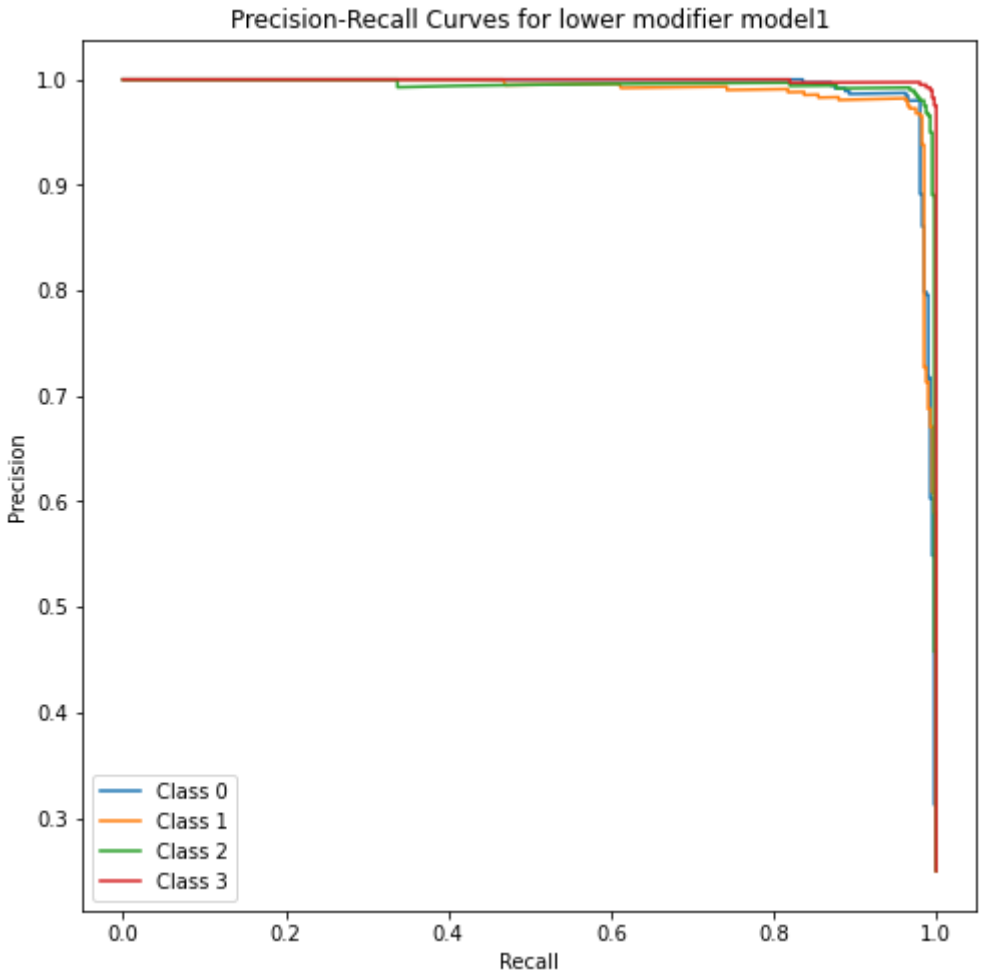
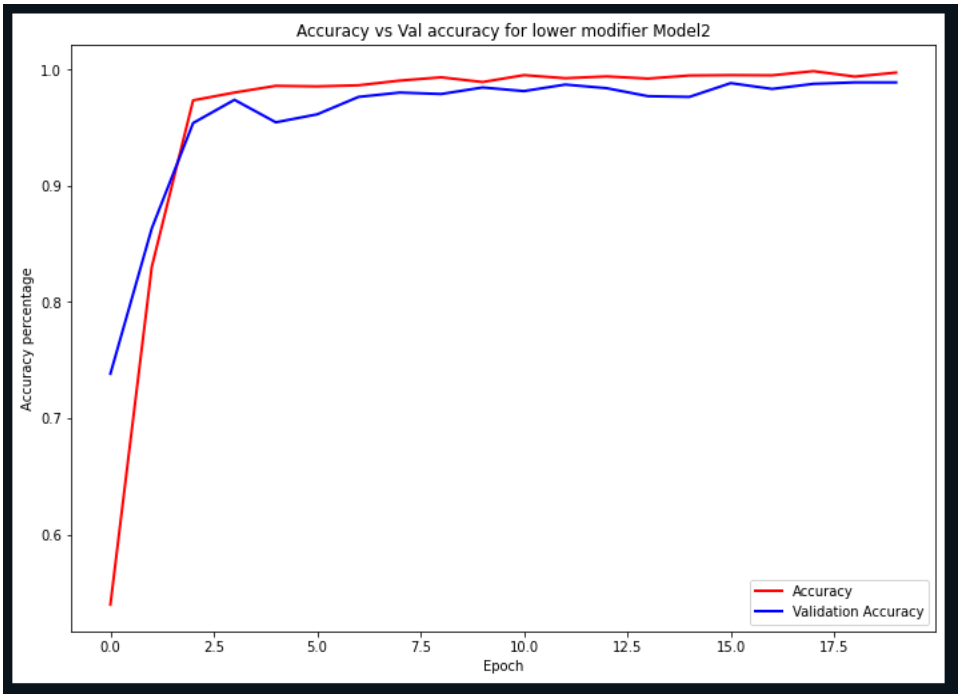


Figure 6- 31: PR curve for lower modifier model -1

Lower modifier model -2



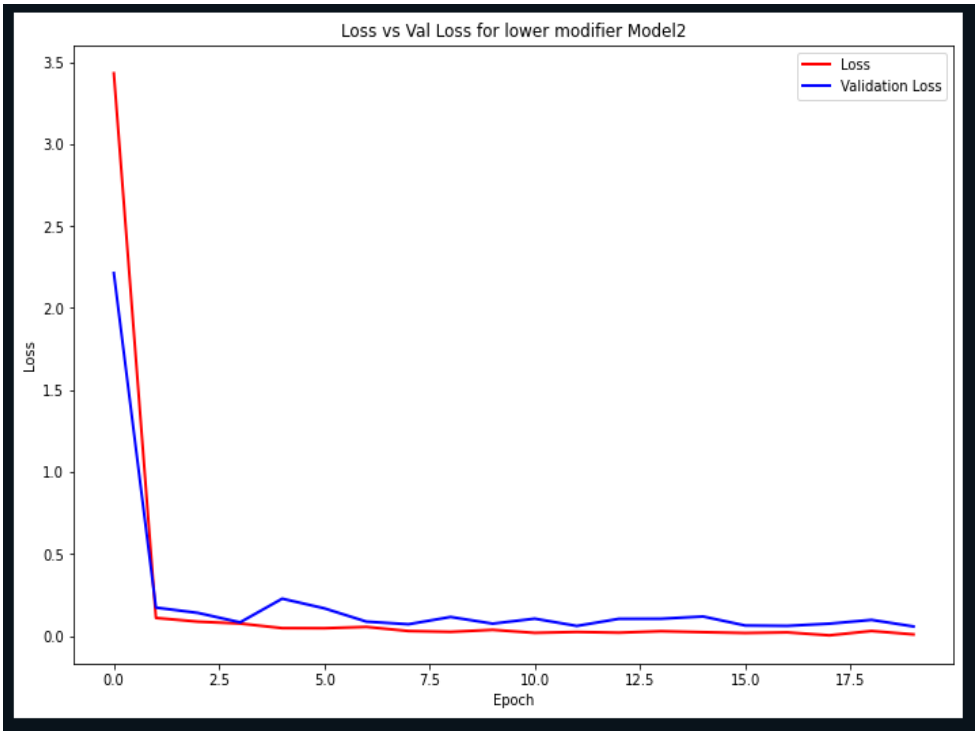


Figure 6- 32: Training accuracy vs validation accuracy and Training loss vs validation loss for lower modifier model -2

Table 6- 18: Confusion matrix for lower modifier model -2

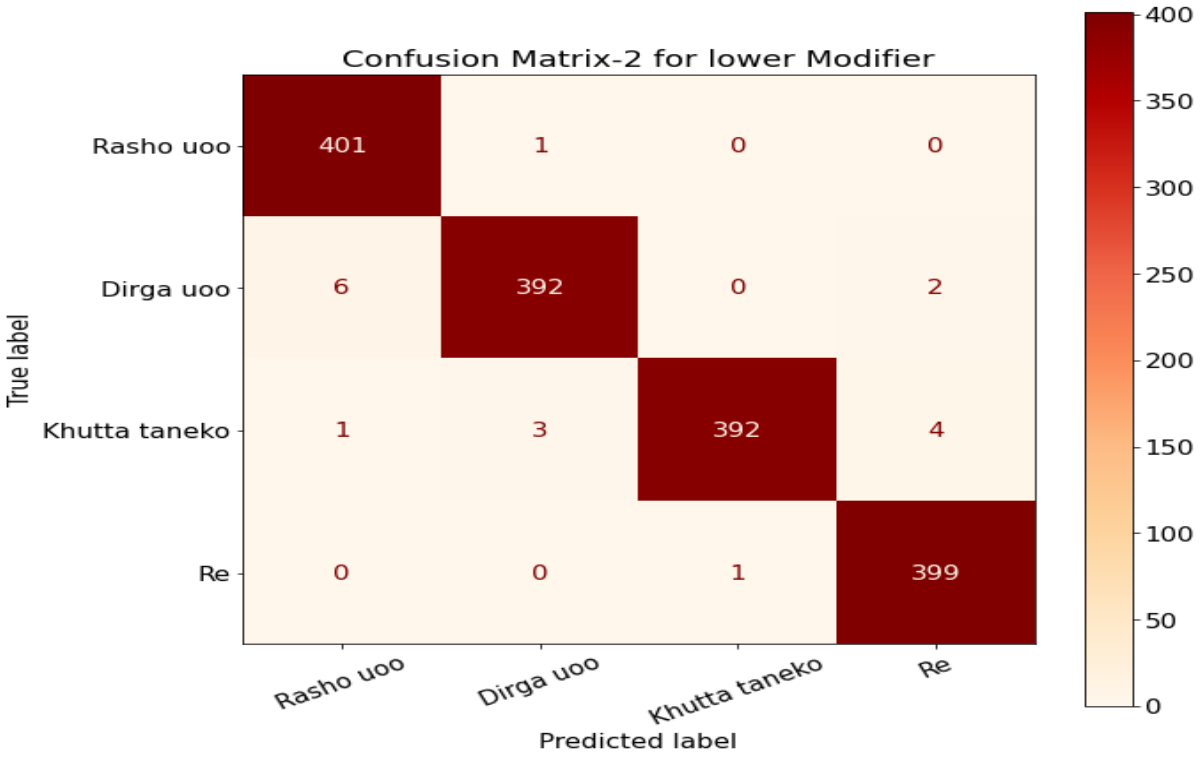


Table 6- 19: Normalized confusion matrix for lower modifier model -2

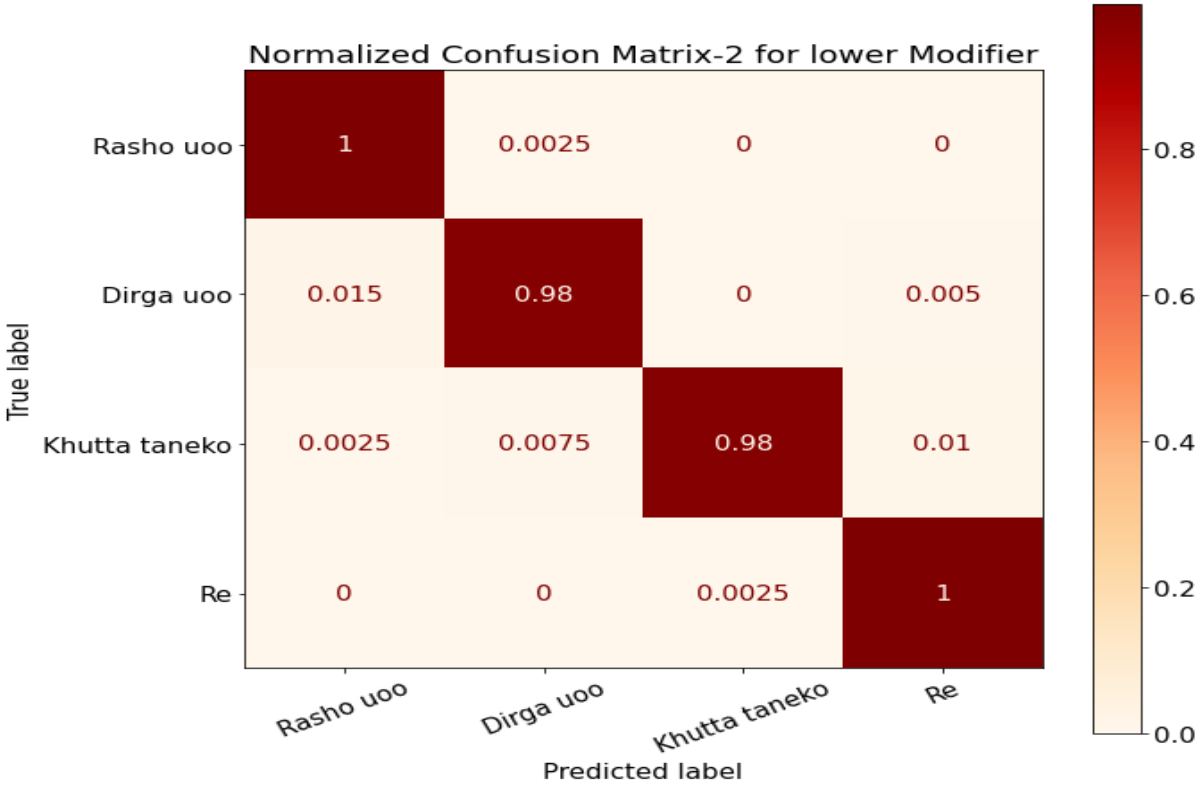


Table 6- 20: Evaluation metrices for lower modifier model -2

	precision	recall	f1-score	support
0	0.98	1.00	0.99	402
1	0.99	0.98	0.98	400
2	1.00	0.98	0.99	400
3	0.99	1.00	0.99	400
accuracy			0.99	1602
macro avg	0.99	0.99	0.99	1602
weighted avg	0.99	0.99	0.99	1602

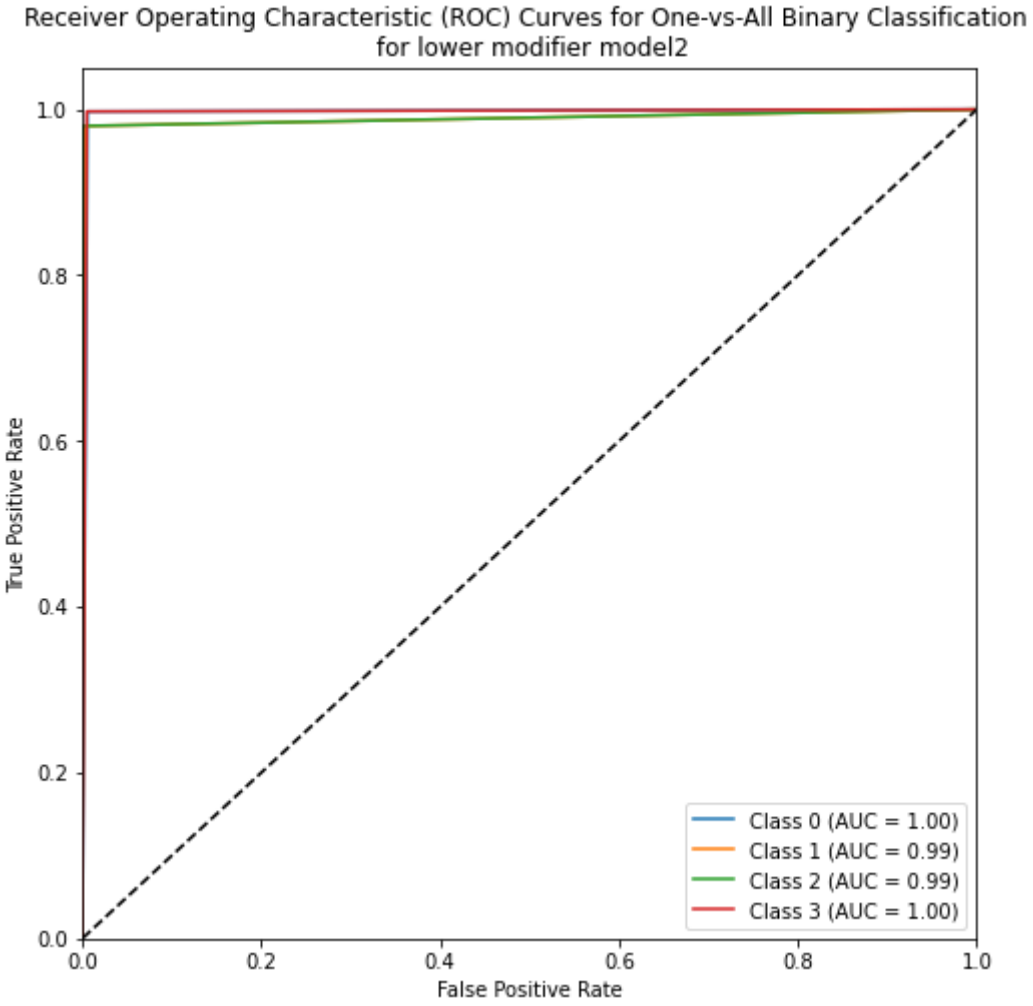


Figure 6- 33: ROC curve for lower modifier model -2

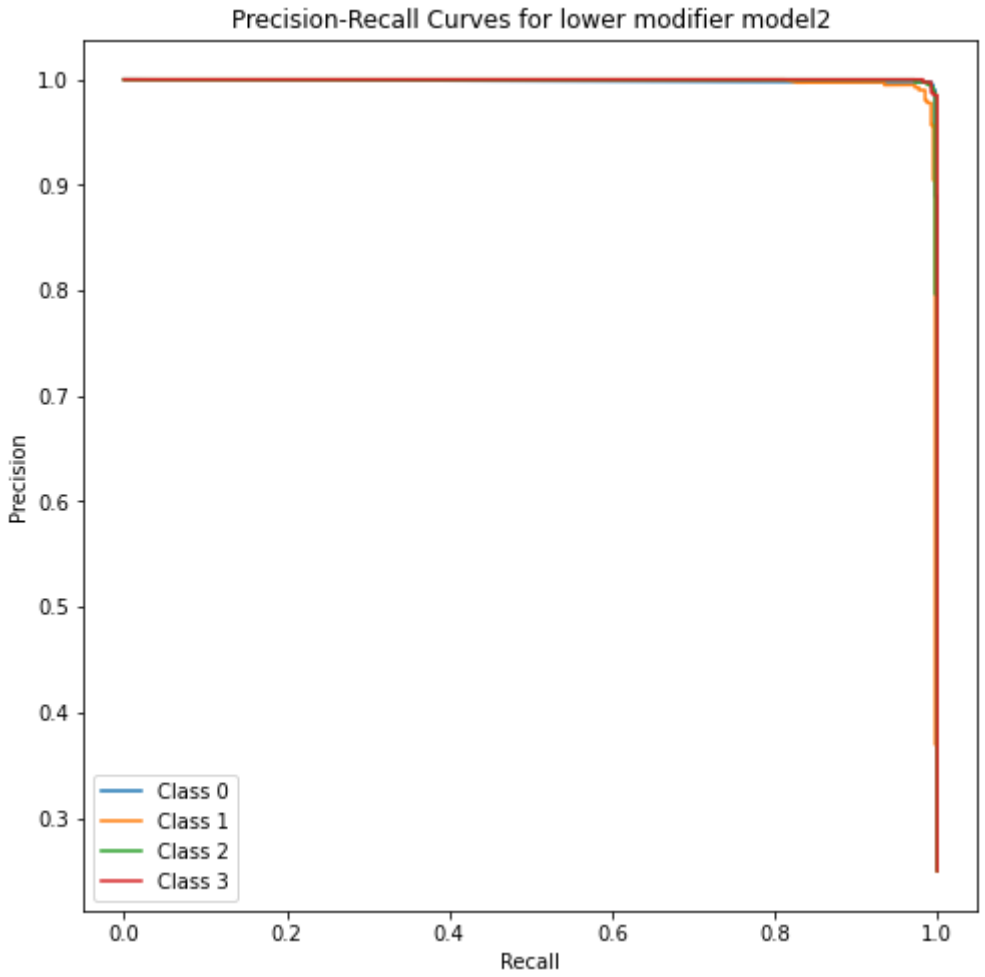


Figure 6- 34: PR curve for lower modifier model -2

While comparing the graph of training accuracy vs validation accuracy, training loss vs validation loss, confusion matrix, evaluation metrices, ROC curve and Precision Recall (PR) curve for each model, we use the model which has better result. The result is shown below:

1. For main character model - main character model -1
2. For half character model – half character model -2
3. For lower modifier model – lower modifier model -2
4. For lower modifier model – upper modifier model -2

A confusion matrix that has been scaled to display the proportion of accurate and inaccurate predictions for each class is called a normalized confusion matrix. As a result, comparing the effectiveness of various classes and determining which classes the model is having the most trouble are made simpler.

In our main model we have 60 different classes of Devanagari words where each row and column of the matrix's 60x60 grid represents a certain category of Devanagari words. The percentage of accurate and inaccurate predictions produced by the model for each class is represented by the values in the matrix.

The diagonal of the matrix, which reflects the proportion of accurate predictions for each class, must be examined while interpreting it. Values above the diagonal show the proportion of times a sample from one class was incorrectly identified as belonging to a different class, while values below the diagonal show the proportion of times a sample from one class was properly identified as belonging to a different class.

Which classes the model is having the greatest trouble with may be determined by analyzing the matrix. It could be feasible to determine which courses are most frequently misclassified as another class or which classes have the lowest percentage of accurate predictions. By concentrating on increasing the accuracy of the model for those specific classes, this knowledge may be leveraged to enhance the model's performance.

6.3 User Interface

Our project's primary objective is to digitize the Devanagari text. So, for this we have made a simple web-app using Streamlit which takes images as input and utilizes the trained model to classify the uploaded image and digitize. The system enables the user to upload image and after clicking "Digitize the text" button , the uploaded image is cropped and preprocessed and finally digitized text is displayed.

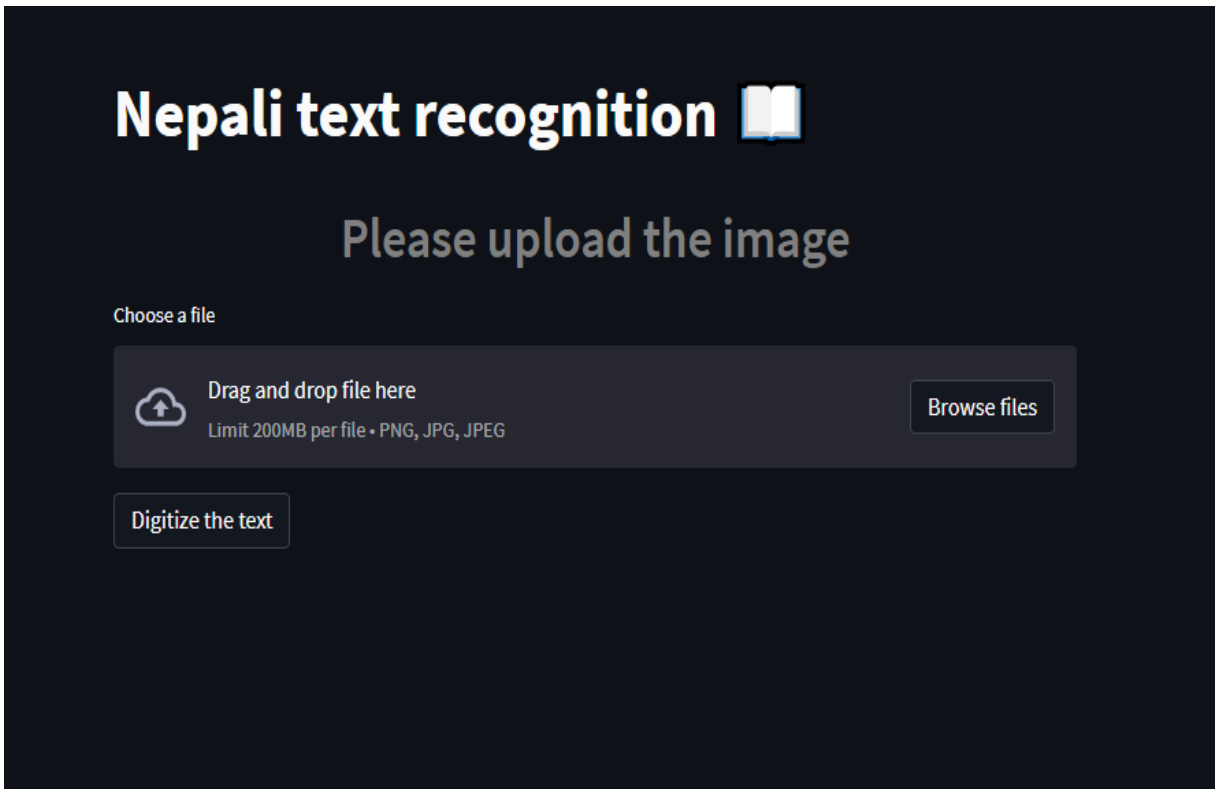


Figure 6- 35: Home screen

Figure 6- 36: Image uploaded

Figure 6- 37: Digitize result

6.4 Error Analysis

Sanskrit, Marathi, Hindi, and other Indian languages are all written in the Devanagari script. Due to the intricate character of the script, Convolutional Neural Networks (CNN) have proven difficult to use for the recognition of Devanagari text. A critical stage in assessing the effectiveness of a CNN-based Devanagari text recognition system is error analysis. The CNN model used in this study was trained on a dataset of Devanagari characters .The dataset comprised 2000 images of each individual Devanagari characters .The model was trained using a batch size of 32 and a learning rate of 0.001. The model was evaluated using precision, recall, and F1-score metrics.

The model struggled with segmenting and recognizing Devanagari text with ligatures which resulted in a lower accuracy. This error was primarily due to the lack of training data on Devanagari text with ligatures. The error analysis revealed that the model frequently misclassified certain Devanagari characters, such as ज, ञ, and ड, due to their visual similarity. Additionally, the model struggled with recognizing certain word structures, such as words with multiple ligatures and words with small spaces between characters.

For individual character identification and word recognition challenges, the CNN-based Devanagari text recognition system showed good accuracy. Unfortunately, the model frequently misclassified visually identical letters and had trouble identifying Devanagari text with ligatures. The model needed more training data for Devanagari text with ligatures and improvements in detecting visually similar letters, according to the error analysis. Overall, the findings demonstrated the promise of CNN-based Devanagari text recognition systems and the necessity of more study in this area.

7. FUTURE ENHANCEMENT

There are various difficulties and opportunities that may be addressed in the future to increase performance even further. Some proposed enhancements are outlined below:

1. Improved accuracy:

The recognition system can be further optimized to improve its accuracy in recognizing handwritten characters, especially in the case of slanted handwriting or characters written in other styles together.

2. Better variant management:

The system can be trained to better handle variations in handwriting, such as variations in character size, spacing, and thickness.

3. Contextual Identification:

The system can be enhanced to take into account the context of recognized characters, such as before and after characters, to improve its accuracy.

4. Identify multiple scripts:

The system can be trained to recognize scripts other than Devanagari, such as Latin or other Indian scripts, to make it more flexible.

5. Integration with other technologies:

The system can be integrated with other technologies, such as Natural Language Processing (NLP) to provide a more comprehensive solution for recognition and processing. document.

6. Improved user interface:

The recognition system's user interface can be improved to make it more user-friendly and accessible with features such as autocorrect suggestions or the ability to add new characters to the system's training data system.

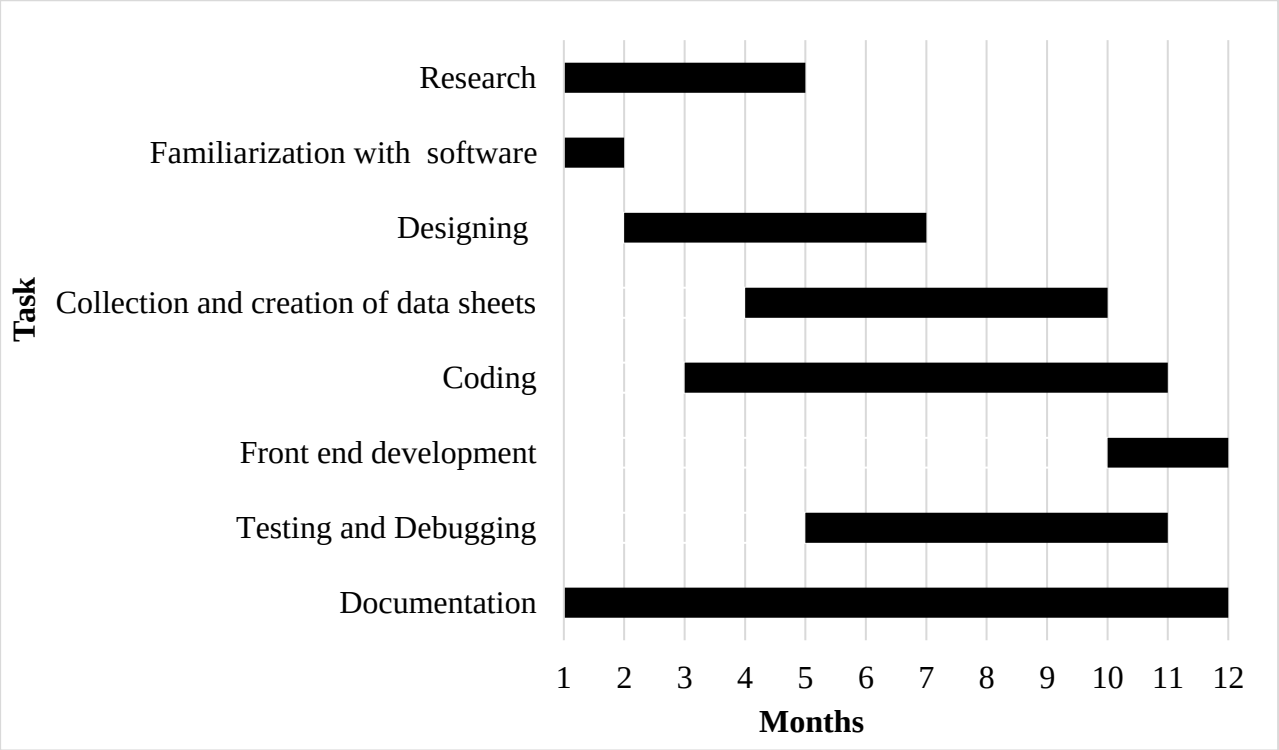
8. CONCLUSION

In conclusion, Nepali handwritten text recognition is a challenging task due to the complex and varied nature of handwritten text. However, with advances in machine learning and computer vision, there have been significant advances in the development of systems capable of accurately recognizing Devanagari handwriting. These systems have the potential to significantly improve access to information and knowledge, especially in areas where traditional printed text is not readily available. Future improvements in accuracy, variant handling, contextual recognition, multi-script recognition, integration with other technologies, and improved user interface will likely help systems This is more efficient and more accessible. However, the Devanagari handwriting recognition field is constantly evolving and further research and development is needed to continue to improve the performance and capabilities of these systems.

9. APPENDICES

Appendix A: Gantt Chart

Table 9- 1: Project Schedule



Appendix B: Project Budget

Table 9- 2: Project Budget

S.N.	TOPIC	COST (Rs.)
1.	Documentation	4,000
2.	Miscellaneous	1,000
Total		5,000

Appendix C: Code Snippets

```
def word_preprocess(bgr_img):#gray image
    blur = cv2.GaussianBlur(bgr_img,(5,5),0)
    th_img = cv2.threshold(blur,0,255,cv2.THRESH_BINARY_INV+cv2.THRESH_OTSU)[1]

    rows, cols = th_img.shape
    bg_test = np.array([th_img[i][i] for i in range(5)])
    if bg_test.all() == 0:
        text_color = 255
    else:
        text_color = 0

    tb = word_borders(th_img, text_color)
    lr = word_borders(th_img.T, text_color)
    dummy = int(np.average((tb[2], lr[2]))) + 2
    template = th_img[tb[0]:tb[1]+int(dummy/2), lr[0]:lr[1]]

    st.image(template, caption='Cropped and Preprocessed image')
    return (template, tb, lr)
```

Figure 9- 1: Preprocessing Code Snippet

```
def siro_rekha_finder(resized):
    photo = resized
    shape=photo.shape
    number = []

    for row in range(1, int(0.6*shape[0])):
        count = 0
        for column in range(1, shape[1]):
            if (photo[row][column]==255):
                count +=1
            else:
                count = count
        number.append(count)

    siro_rekha=np.argmax(number)+2
    if (number[siro_rekha-2]<int(0.4*shape[1])):
        return 0
    else:
        return siro_rekha
```

Figure 9- 2: Siro Rekha detection Code Snippet

```python
import Augmentor

# Define the source folder containing the original images
source_folder = "F:/adhi moto sa/adhi moto sa inverted"

# Create a pipeline object
p = Augmentor.Pipeline(source_directory=source_folder, output_directory="F:/half form characters/aadhi moto sa")

# Add operations to the pipeline to perform image augmentation
p.rotate(probability=0.2, max_left_rotation=1.5, max_right_rotation=1.5)
#p.zoom(probability=0.2, min_factor=1.1, max_factor=1.25)
p.skew(probability=0.2)
p.shear(probability=0.2, max_shear_left=1, max_shear_right=1)
#p.crop_random(probability=0.2, percentage_area=0.3)
p.resize(probability=1.0, width=32, height=32)
#p.random_distortion(probability=0.2, grid_width=4, grid_height=4, magnitude=5)
#p.zoom_random(probability=0.5, percentage_area=0.8)
p.random_contrast(probability=0.7, min_factor=0.75, max_factor=1.25)
p.random_brightness(probability=0.7, min_factor=0.75, max_factor=1.25)
```

Figure 9- 3: Data Augmentation Code Snippet

```python
def segmentation(bordered, siro_rekha, width):
    shape = bordered.shape
    img = bordered
    if siro_rekha>15:
        img[siro_rekha-8:siro_rekha+10,0:width] = 0
    else:
        img[0:siro_rekha+13,0:width] = 0
    image = img.T
    shape = image.shape

    bg = np.repeat(0, shape[1])
    array = [0]
    for row in range(1, shape[0]):
        if  (np.equal(bg, image[row]).all()):
            array.append(row)

    l1 = len(array)
```

Figure 9- 4: Segmentation Code Snippet

```
a = len(string1)
j=0
for j in range(a) :
    print(string1[j],end='')
```

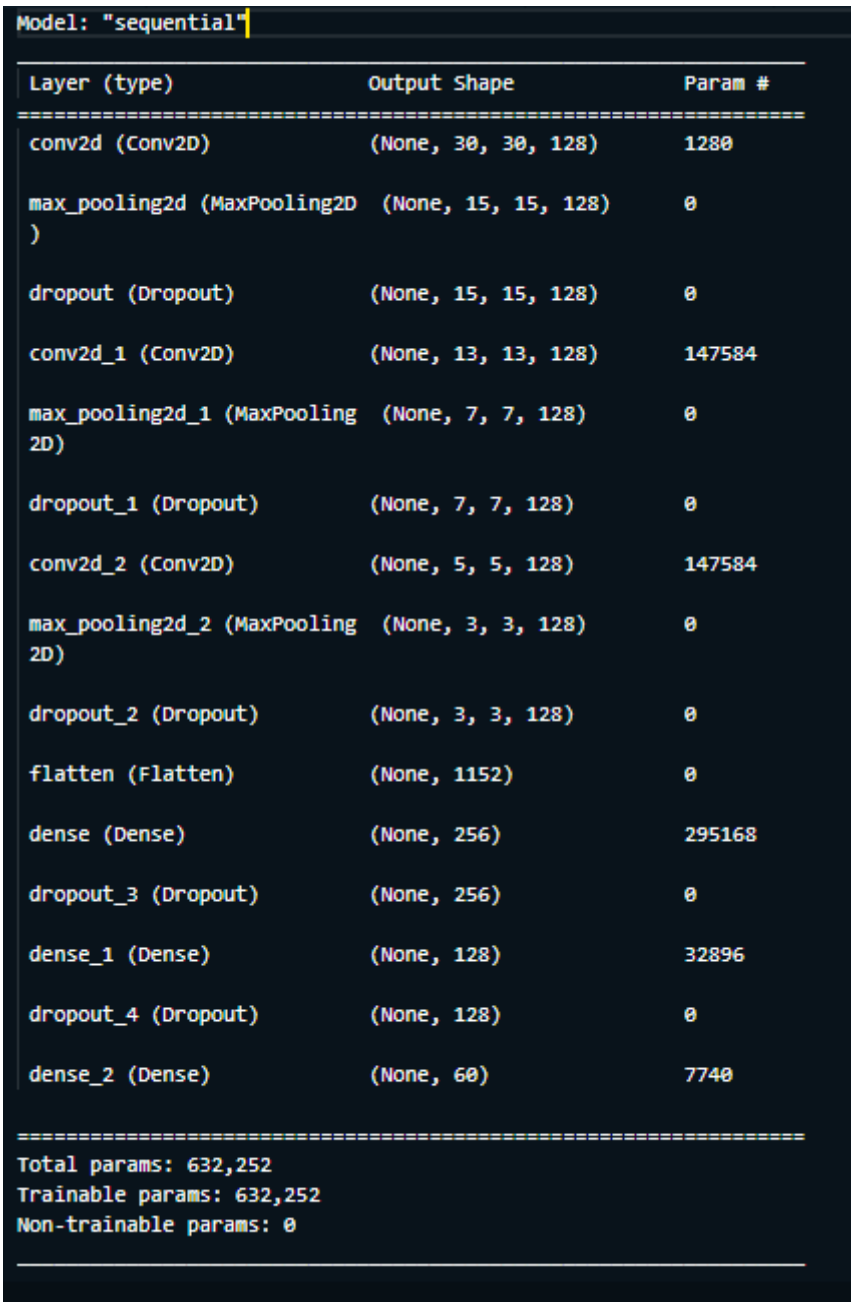
[17] ✓ 0.0s

... खुसी

Figure 9- 5: Combination of recognized character

```
image_size = (32,32)
batch_size = 46

#Data generators
train_datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)

train_batches = train_datagen.flow_from_directory(
    dataset_path,
    target_size=image_size,
    batch_size=batch_size,
    color_mode = 'grayscale',
    class_mode='sparse',
    subset='training'
)

validation_batches = train_datagen.flow_from_directory(
    dataset_path,
    target_size=image_size,
    batch_size=batch_size,
    color_mode = 'grayscale',
    class_mode='sparse',
    subset='validation'
)

test_batches = train_datagen.flow_from_directory(
    dataset_path,
    target_size=image_size,
    batch_size=batch_size,
    color_mode = 'grayscale',
    class_mode='sparse',
    shuffle=False,
    subset='validation'
)
```

Figure 9- 6: Data Generator Code Snippet



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 30, 30, 128)       1280

max_pooling2d (MaxPooling2D  (None, 15, 15, 128)      0
)

dropout (Dropout)            (None, 15, 15, 128)       0

conv2d_1 (Conv2D)            (None, 13, 13, 128)       147584

max_pooling2d_1 (MaxPooling  (None, 7, 7, 128)        0
2D)

dropout_1 (Dropout)          (None, 7, 7, 128)         0

conv2d_2 (Conv2D)            (None, 5, 5, 128)         147584

max_pooling2d_2 (MaxPooling  (None, 3, 3, 128)        0
2D)

dropout_2 (Dropout)          (None, 3, 3, 128)         0

flatten (Flatten)            (None, 1152)              0

dense (Dense)                (None, 256)               295168

dropout_3 (Dropout)          (None, 256)               0

dense_1 (Dense)              (None, 128)               32896

dropout_4 (Dropout)          (None, 128)               0

dense_2 (Dense)              (None, 60)                7740

=================================================================
Total params: 632,252
Trainable params: 632,252
Non-trainable params: 0
_________________________________________________________________
```

Figure 9- 7: Code Snippet for main character model

Appendix D: Plagiarism Summary

Digitization of Devanagari Handwritten Text

ORIGINALITY REPORT

19%
SIMILARITY INDEX

PRIMARY SOURCES

1	mafiadoc.com Internet	321 words — 2%
2	www.researchgate.net Internet	294 words — 2%
3	techscience.com Internet	204 words — 1%
4	www.coursehero.com Internet	167 words — 1%
5	M. HANMANDLU, POOJA AGRAWAL, BREJESH LALL. "Segmentation of Handwritten Hindi Text: A Structural Approach", International Journal of Computer Processing of Languages, 2009 Crossref	166 words — 1%
6	Lecture Notes in Computer Science, 2015. Crossref	130 words — 1%
7	www.jetir.org Internet	115 words — 1%
8	huggingface.co Internet	81 words — < 1%

9	Dhanashree Joshi, Sarika Pansare. "Combination of Multiple Image Features along with KNN Classifier for Classification of Marathi Barakhadi", 2015 International Conference on Computing Communication Control and Automation, 2015 Crossref	73 words — < 1%
10	Guna Suryo Aji, Teuku Salman Farizi, Muhammad Farhan, Riri Fitri Sari. "Machine Learning Based SPAM Message Classification System using Blockchain Technology", 2021 17th International Conference on Quality in Research (QIR): International Symposium on Electrical and Computer Engineering, 2021 Crossref	71 words — < 1%
11	moam.info Internet	71 words — < 1%
12	kalaharijournals.com Internet	70 words — < 1%
13	dokumen.pub Internet	61 words — < 1%
14	Shailesh Acharya, Ashok Kumar Pant, Prashnna Kumar Gyawali. "Deep learning based large scale handwritten Devanagari character recognition", 2015 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA), 2015 Crossref	59 words — < 1%
15	Veshraj Joshi, Sanjeeb Prasad Panday. "Character Component Segmentation and Categorization of Machine Printed Text in Devanagari (Nepali) Script in Digital Image Processing", 2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS), 2018 Crossref	55 words — < 1%

16	www.ijeit.com Internet	53 words — < 1%
17	link.springer.com Internet	48 words — < 1%
18	www.irjmets.com Internet	48 words — < 1%
19	Khorshed Alam, Nishargo Nigar, Heidy Erler, Anonnya Banerjee. "Speech Emotion Recognition from Audio Files Using Feedforward Neural Network", 2023 International Conference on Electrical, Computer and Communication Engineering (ECCE), 2023 Crossref	45 words — < 1%
20	N. Shahid, M. Ali Shah. "ANOMALY DETECTION IN SYSTEM LOGS IN THE SPHERE OF DIGITAL ECONOMY", Competitive Advantage in the Digital Economy (CADE 2021), 2021 Crossref	40 words — < 1%
21	Sandeep Dwarkanath Pande, Pramod Pandurang Jadhav, Rahul Joshi, Amol Dattatray Sawant et al. "Digitization of handwritten Devanagari text using CNN transfer learning – A better customer service support", Neuroscience Informatics, 2022 Crossref	40 words — < 1%
22	cdn.iiit.ac.in Internet	34 words — < 1%
23	medium.com Internet	32 words — < 1%

24	"Proceedings of the Future Technologies Conference (FTC) 2020, Volume 1", Springer Science and Business Media LLC, 2021 Crossref	29 words — < 1%
25	monkeylearn.com Internet	27 words — < 1%
26	www.baeldung.com Internet	27 words — < 1%
27	Arna Fariza, Wiratmoko Yuwono, Reesa Akbar, Rengga Asmara, I Gede Kresna Putra Aryawan. "Mobile Based Mosquito Larvae Recognition from Photo Image Using Convolutional Neural Network", Trans Tech Publications, Ltd., 2023 Crossref	26 words — < 1%
28	journal.deerwalk.edu.np Internet	25 words — < 1%
29	"Third International Conference on Image Processing and Capsule Networks", Springer Science and Business Media LLC, 2022 Crossref	24 words — < 1%
30	D. Dinesh Ram, U. Muthukumaran, N. Sabiyath Fatima. "Chapter 36 Enhanced Human Action Recognition with Ensembled DTW Loss Function in CNN LSTM Architecture", Springer Science and Business Media LLC, 2023 Crossref	22 words — < 1%
31	Sudan Prajapati, Shashidhar Ram Joshi, Aman Maharjan, Bikash Balami. "Evaluating Performance of Nepali Script OCR using Tesseract and Artificial Neural Network", 2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS), 2018	22 words — < 1%

Crossref

32	Nirajan Pant, Bal Krishna Bal. "Improving Nepali OCR performance by using hybrid recognition approaches", 2016 7th International Conference on Information, Intelligence, Systems & Applications (IISA), 2016 Crossref	21 words — < 1%
33	tnsroindia.org.in Internet	20 words — < 1%
34	www.victoriaparkat12.com Internet	19 words — < 1%
35	Feifei Cui, Shuang Li, Zilong Zhang, Miaomiao Sui, Chen Cao, Abd El-Latif Hesham, Quan Zou. "DeepMC-iNABP: Deep learning for multiclass identification and classification of nucleic acid-binding proteins", Computational and Structural Biotechnology Journal, 2022 Crossref	18 words — < 1%
36	Pellakuri Vidyullatha, Bui Thanh Hung, Prasun Chakrabarti. "An Adaptive Deep Convolution Neural Network for High Pixel Image Segmentation and Classification", 2023 International Conference on Innovative Data Communication Technologies and Application (ICIDCA), 2023 Crossref	18 words — < 1%
37	Belle Fille Murorunkwere, Origene Tuyishimire, Dominique Haughton, Joseph Nzabanita. "Fraud Detection Using Neural Networks: A Case Study of Income Tax", Future Internet, 2022 Crossref	17 words — < 1%
38	Haipeng Gong, George D. Rose. "Does secondary structure determine tertiary structure in	17 words — < 1%

proteins?", Proteins: Structure, Function, and Bioinformatics, 2005
Crossref

39	dspace.bracu.ac.bd Internet	16 words — < 1%
40	www.iosrjournals.org Internet	16 words — < 1%
41	Kesler, Karen Keller. "The Direct and Indirect Effects of Site Suitability on Eastern Monarch Butterfly Migratory Populations.", The University of North Carolina at Greensboro ProQuest	15 words — < 1%
42	dione.lib.unipi.gr Internet	15 words — < 1%
43	hasty.ai Internet	14 words — < 1%
44	soar.wichita.edu Internet	14 words — < 1%
45	www.inf.uni-hamburg.de Internet	14 words — < 1%
46	www.mdpi.com Internet	14 words — < 1%
47	Abhijan Wasti, Saugat Tripathi, Sandeep Regmi, Sanjeev Yadav, Dinesh Baniya Kshatri. "Aerial View Based Guidance System", Journal of Innovations in Engineering Education, 2020 Crossref	13 words — < 1%

48	Liang Gao, Hui Liu, Minhang Yang, Long Chen, Yaling Wan, Yurong Qian, Zhengqing Xiao. "STransFuse: Fusing Swin Transformer and Convolutional Neural Network for Remote Sensing Image Semantic Segmentation", Institute of Electrical and Electronics Engineers (IEEE), 2021 Crossref Posted Content	13 words — < 1%
49	github.com Internet	13 words — < 1%
50	web.archive.org Internet	13 words — < 1%
51	Cao, Yang. "Framework for User Sentiment Analysis to Improve the Usability of Applications for Generation z and Young Adults", Auburn University, 2023 ProQuest	12 words — < 1%
52	Guide to OCR for Arabic Scripts, 2012. Crossref	12 words — < 1%
53	LATESH MALIK, P. S. DESHPANDE. "RECOGNITION OF HANDWRITTEN DEVANAGARI SCRIPT", International Journal of Pattern Recognition and Artificial Intelligence, 2011 Crossref	12 words — < 1%
54	Mamta Bisht, Richa Gupta. "Offline Handwritten Devanagari Word Recognition Using CNN-RNN-CTC", SN Computer Science, 2022 Crossref	12 words — < 1%
55	lib.dr.iastate.edu Internet	12 words — < 1%

56	"Computing Science, Communication and Security", Springer Science and Business Media LLC, 2020 Crossref	11 words — < 1%
57	"International Conference on Innovative Computing and Communications", Springer Science and Business Media LLC, 2022 Crossref	11 words — < 1%
58	Roshini Mohanan, Jisha Jacob, G. R. Gnana King. "Chapter 72 A CNN-Based Underage Driver Detection System", Springer Science and Business Media LLC, 2023 Crossref	11 words — < 1%
59	Veena Bansal, R.M.K. Sinha. "Segmentation of touching and fused Devanagari characters", Pattern Recognition, 2002 Crossref	11 words — < 1%
60	scholarworks.iupui.edu Internet	11 words — < 1%
61	1library.net Internet	10 words — < 1%
62	ee.iisc.ac.in Internet	10 words — < 1%
63	flipkarma.com Internet	10 words — < 1%
64	"Intelligent Computing and Applications", Springer Science and Business Media LLC, 2021 Crossref	9 words — < 1%

65	Knopf, Katelyn. "A Non-Contacting System for Rail Neutral Temperature and Stress Measurements", University of South Carolina, 2020 ProQuest	9 words — < 1%
66	P. Pavan Kumar, Chakravarthy Bhagvati, Arun Agarwal. "On performance analysis of end-to-end OCR systems of Indic scripts", Proceeding of the workshop on Document Analysis and Recognition, 2012 Crossref	9 words — < 1%
67	Yasarer, Hakan. "Decision making in engineering prediction systems.", Proquest, 2013. ProQuest	9 words — < 1%
68	citeseerx.ist.psu.edu Internet	9 words — < 1%
69	mobt3ath.com Internet	9 words — < 1%
70	worldcomp-proceedings.com Internet	9 words — < 1%
71	www.hindawi.com Internet	9 words — < 1%
72	www.sreedattha.ac.in Internet	9 words — < 1%
73	"Guide to OCR for Indic Scripts", Springer Science and Business Media LLC, 2010 Crossref	8 words — < 1%
74	"Intelligent System Design", Springer Science and Business Media LLC, 2021 Crossref	8 words — < 1%

75	Akter, Mousumy. "Crash Risk Analysis of Two Way Stop Control Intersections on Rural Two-Lane Highways Using Machine Learning Classification Algorithms", University of Louisiana at Lafayette, 2021 ProQuest	8 words — < 1%
76	Deepika Gupta, Soumen Bag. "An Efficient Character Segmentation Approach for Handwritten Hindi Text", 2018 5th International Conference on Signal Processing and Integrated Networks (SPIN), 2018 Crossref	8 words — < 1%
77	Divakar Yadav, Sonia Sanchez-Cuadrado, Jorge Morato. "Optical Character Recognition for Hindi Language Using a Neural-network Approach", Journal of Information Processing Systems, 2013 Crossref	8 words — < 1%
78	Qiushi Wang, Haolin Li, Haiqian Zhao, Xiaoxue Zhang, Müslüm Arıcı, Huaizhi Li. "Quantitative analysis on the oil content of oilfield wastewater based on a convolutional neural network model and ultraviolet transmission spectroscopy", Water Science and Technology, 2023 Crossref	8 words — < 1%
79	digitalcommons.mtu.edu Internet	8 words — < 1%
80	tudr.thapar.edu:8080 Internet	8 words — < 1%
81	www.aimlfront.com Internet	8 words — < 1%
82	www.ncbi.nlm.nih.gov Internet	8 words — < 1%

83	www.research-collection.ethz.ch Internet	8 words — < 1%
84	www.research.manchester.ac.uk Internet	8 words — < 1%
85	www.semanticscholar.org Internet	8 words — < 1%
86	Handbook of Document Image Processing and Recognition, 2014. Crossref	7 words — < 1%
87	"Computer Vision, Pattern Recognition, Image Processing, and Graphics", Springer Science and Business Media LLC, 2018 Crossref	6 words — < 1%
88	Sulove Bhattarai, Sudip Bhujel, Santosh Adhikari, Shanta Maharjan. "Design and Implementation of a Portable ECG Device", Journal of Innovations in Engineering Education, 2020 Crossref	6 words — < 1%
89	www.ijfrcsce.org Internet	6 words — < 1%

EXCLUDE QUOTES	ON	EXCLUDE SOURCES	OFF
EXCLUDE BIBLIOGRAPHY	ON	EXCLUDE MATCHES	OFF

References

[1] S. Kompalli, S. Setlur and V. Govindaraju, "Devanagari OCR using a recognition driven segmentation framework and stochastic language models," *International Journal on Document Analysis and Recognition (IJDAR)* , vol. 12, pp. 123-138, 2009.

[2] B. Shaw, S. K. Parui and M. Shridhar, "Offline Handwritten Devanagari Word Recognition: A Holistic Approach Based on Directional Chain Code Feature and HMM," *2008 International Conference on Information Technology*, pp. 203-208, 2008.

[3] N. V. Rao, D. A. Sastry, A. Chakravarthy and K. P, "OPTICAL CHARACTER RECOGNITION TECHNIQUE," *Journal of Theoretical and Applied Information Technology,* vol. 83, 2016.

[4] N. Pant, "Improving Nepali OCR performance by using hybrid recognition approaches," *7th International Conference on Information, Intelligence, Systems & Applications (IISA,* pp. 1-6, 2016.

[5] S. Puri and P. S. Singh, "An efficient Devanagari character classification in printed and handwritten documents using SVM," *Procedia Computer Science 152,* pp. 111-121, 2019.

[6] D. S. Pande, P. P. Jadhav, R. Joshi, A. D. Sawant, V. Muddebihalkar, S. Rathod, M. N. Gurav and S. Das, "Digitization of handwritten Devanagari text using CNN transfer learning- A better customer service support," *Neuroscience Informatics 2*, vol. 2, no. 3, 2022.

[7] S. Sayyad, A. Jadhav, M. Jadhav, S. Miraje, P. Bele and A. Pandhare, "Devnagiri Character Recognition Using Neural Networks," *International Journal of Engineering and Innovative Technology (IJEIT),* vol. 3, no. 1, pp. 476-480, 2013.

[8] C. Johny, L. Wolf-Sonkin, A. Gutkin and B. Roark, "Finite-state script normalization and processing utilities:The Nisaba Brahmic library," *The 16th Conference of the European Chapter of the Association for Computational Linguistics* , pp. 14-23, 2021.

[9] M. Sinha and V. Bansal, "A complete OCR for printed Hindi text in Devanagari script," *Proceedings of Sixth International Conference on Document Analysis and Recognition,* pp. 800-804, 2001.

[10] S. Shakya, S. Tuladhar, R. Pandey and B. K. Bal, "Interim Report on Nepali OCR," *Madan Puraskar Pustakalaya,* 2009.

[11] A. Ghimire, A. Chapagain, U. Bhattarai and A. Jaiswal, "Nepali Handwriting Recognition using Convolution Neural Network," *International Research Journal of Innovations in Engineering and Technology,* vol. 4, no. 5, pp. 5-9, 2020.

[12] B. R. Dawadi, R. C. Pandey, S. Sharma and A. Basnet, "Dictionary Based Nepali Word Recognition using Neural Network," *International Journal of Scientific & Engineering Research,* vol. 8, no. 5, pp. 473-479, 2017.

[13] S. Prajapati, A. Maharjan, S. R. Joshi and B. Balami, "Assessing and Analyzing Tesseract Based Nepali Script OCR," *Deerwalk Journal of Computer Science and Information Technology,,* vol. 1, pp. 37-46, 2019 .

[14] Ingroj, "nepalinlp," 14 12 2017. [Online]. Available: https://nepalinlp.com/processing-unicodedevnagari-in-python/. [Accessed 20 12 2022].

[15] S. Bag and A. Krishna, "Character Segmentation of Hindi Unconstrained Handwritten Words," *IWCIA 2015: Combinatorial Image Analysis* , vol. 9448, pp. 247-260, 2016.

[16] [Online]. Available: https://sourceforge.net/projects/tesseracthindi/. [Accessed 04 02 2023].

[17] "Convolutional Neural Networks (CNN): Summary," 18 August 2018. [Online]. Available: https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-summary?fbclid=IwAR1uOeNxti92XGE tp_uRiLKMuUo4W4SyNVRdfgXp_3ZxX4F2sOV4qSA8Xc. [Accessed 15 August 2022].