TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

Minor Project Report

On

Writer Prediction based on Handwritten Text

Submitted By:

Manoj Paudel (Exam Roll No: 31472)

Prince Poudel (Exam Roll No: 31483)

Ronish Shrestha (Exam Roll No: 31487)

Sonish Poudel (Exam Roll No: 31489)

Submitted To:

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

March, 2024

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

Minor Project Report

On

Writer Prediction based on Handwritten Text

Submitted By:

Manoj Paudel (THA077BCT025)

Prince Poudel (THA077BCT036)

Ronish Shrestha (THA077BCT040)

Sonish Poudel (THA077BCT042)

Submitted To:

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

In partial fulfillment for the award of the Bachelor's Degree in Computer Engineering.

Under the Supervision of

Er. Praches Acharya

March, 2024

DECLARATION

We hereby declare that the report of the project entitled **"Writer prediction based on 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 Computer 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, we are the only author of this complete work, and no sources other than the listed here have been used in this work.

Manoj Paudel	(THA077BCT025)	_____
Prince Poudel	(THA077BCT036)	_____
Ronish Shrestha	(THA077BCT040)	_____
Sonish Poudel	(THA077BCT042)	_____

Date: March, 2024

CERTIFICATE OF APPROVAL

The undersigned certify that they have read and recommended to the **Department of Electronics and Computer Engineering, IOE, Thapathali Campus**, a minor project work entitled **"Writer prediction based on Handwritten text"** submitted by **Manoj Paudel, Prince Poudel, Ronish Shrestha** and **Sonish Poudel** in partial fulfillment for the award of Bachelor's Degree in Computer 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 Computer Engineering.

_____

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

_____

External Examiner
Er. Birodh Rijal
Everest Engineering College, Sanepa, Lalitpur

_____

Project Co-ordinator
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, 2024

COPYRIGHT

ACKNOWLEDGEMENT

We would like to offer our genuine gratitude towards the Department of Electronics and Computer Engineering, Thapathali Campus for providing this opportunity to learn and carry our insight in the form of a minor venture as well as for enabling us to advance our professional development through it.

We are humbly grateful to our supervisor **Er. Praches Acharya** for his invaluable guidance and feedback for this project.

Lastly, we would like to thank all of our teachers, classmates and other direct and indirect contributors for their insightful advice and recommendations.

Manoj Paudel (THA077BCT025)

Prince Poudel (THA077BCT036)

Ronish Shrestha (THA077BCT040)

Sonish Poudel (THA077BCT042)

ABSTRACT

In order to identify the author on handwritten text, Variation in how people write plays a critical role. This project proposes a new technique using advanced computer models called “Transformer” to solve this problem. Mainly, Transformer are used for Natural language processing but after introduction of Vision Transformer, we can also use transformer for computer vision problem. The limitation of Vision transformer is that we have to train the model in large amount of dataset with millions of parameters. So, we use the Compact Convolutional Transformer (CCT) due to its advantage of working well even with limited data and uses fewer calculations comparatively than other transformer models. After introduction of Convolutional Tokenization and sequence pooling technique, we could down sample the number of parameters making less time and resource consuming during processing. We test our model’s accuracy on the popular IAM handwriting database, CVL database and custom dataset, ensuring its ability to generalize across diverse handwriting samples.

Keywords: CCT, CVL dataset, IAM dataset, Transformer ,Vision Transformer

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
API	Application Programming Interface
CCT	Compact Convolutional Transformer
CNN	Convolutional Neural Network
CV	Computer Vision
CVT	Compact Vision Transformer
GUI	Graphical User Interface
HDL	Hybrid Deep Language
ICDAR	International Conference on Document Analysis and Recognition
KNN	K-Nearest Neighbors
MHSA	Multi-Head Self Attention
MLP	Multi-level Perceptron
MSVM	Multiclass-Support Vector Machine
SVM	Support Vector Machine
ViT	Vision Transformer

1 INTRODUCTION

1.1 Background

In this project, we identify the writer primarily based on human handwritten text. As we know, Handwriting is a kind of behavioral biometrics. As human beings, there are certainly many variations within the manner a person writes. Writer is identified by means of capturing particular traits of the handwriting dependency of one writer, which differs from other authors. Writer identification has been applied in anti-crime and historical report evaluation fields and can be extended to many applications. As plenty of the arena has been making technological improvements we see the sector utilizing digital documents and on-line signatures. It is vital to distinguish off-line and virtual or on-line handwriting. On-line handwriting has the functionality of shooting specific functions together with the amount of strain implemented even as writing or the series of strokes, opposite to distinguishing photographs of off-line handwriting, which is limited in features, as it relies on just its visual properties.

Forensic scholars have analyzed handwriting by evaluating substances to determine who wrote the fragments. Some of the matters an expert can compare are the structure of the phrases, the curvature of every letter and the spacing. In handwriting evaluation, as generation improves, the argument is that human judgment is unreliable; computers are being used to distinguish clerks from forensic professionals. Vision transformer has been capable of outperform the traditional CNN architecture in photo type duties. But, it calls for more datasets than CNN. Compact transformers is the lightweight variation of imaginative and prescient transformer, which has established to get proper effects on small dataset. This motivates the application of compact transformers to apprehend the writer of handwritten photographs. In particular, we wish to predict the author of the given handwriting.

1.2 Motivation

The motivation for undertaking this project stems from our desire to engage in challenging tasks that solely depend on software components .The writer identification based on human text begins a widely recognized problem in the forensic world, and is an ideal candidate for our exploration. Instead of existing solutions that utilize human

scanning which contributes some human error, our project is motivated by resourcefulness and efficiency. We pursue a simple yet effective alternative to the existing method of author identification, showing problem-solving skills in computer and engineering disciplines.

1.3 Problem Definition

The problem in the project is to predict the writer based on the handwritten text.

1.4 Objectives

- To predict the writer of the handwritten document using Compact Convolutional Transformer (CCT).

1.5 Scope and Applications

1.5.1 Biometric Identification

The mission of the usage of handwritten textual content for human identity could fall beneath the umbrella of biometric identity. Biometric identifiers are particular bodily or behavioral traits used to differentiate people. Handwriting, with its individual quirks and styles, suits this definition.

1.5.2 Forensic Analysis

In Forensic investigations, analyzing handwritten text plays as a powerful tool for identifying individuals and revealing the truth. Moreover, attributing authorship to anonymous notes or identifying forgeries in crucial documents, the unique quirks and stylistic nuances of handwriting can provide valuable clues. In criminal investigations, reading handwritten evidence from crime scenes, confessions, or witness statements can link suspects to their actions or become aware of victims. Even in bloodless cases, reanalyzing old handwritten evidence with superior techniques can free up new leads or find previously unknown culprits.

1.5.3 Historical Document Analysis

This technology could unveil the authentic pen at the back of innovative proclamations, expose forgeries plaguing ancient facts, or maybe become aware of nameless

chroniclers who documented pivotal activities. By analyzing stylistic nuances and person traits in historical scripts, the mission can become a detective within the records, reconstructing narratives and revealing the hidden palms that fashioned history. However, the challenges of decoding diminished ink, managing archaic scripts, and accounting for stylistic evolution problems across centuries must no longer be underestimated.

1.5.4 Scalability and Adaptability

Overall, there are challenges to address; the project has promising potential for scalability and adaptability with careful consideration of the factors like Data Bias, Privacy Concerns and Generalizability. By employing efficient algorithms, robust data management solutions, and techniques for handling variability, the project can be scaled to real-world applications and adapted to different domains and languages.

2 LITERATURE REVIEW

Many researchers have already researched in the field of writer identification based on handwritten text using various dataset in many languages and using various types of model. Variation in writing styles from person to person makes this project challenging. Many machine-learning approaches have been applied to get the accurate result while predicting the writers. Researchers have also explored the techniques like graphology, textual features which can be use with deep learning to solve this problem.

2.1 Different methods used for this task

Xiang et al. have used multi stream structure for writer identification on their paper "DeepWriter: A multistream Deep CNN for Text-independent Writer Identification". They conducted their experiment on two languages that is English and Chinese. They trained their model in both language and found to obtain better performance. [1]

Accuracy obtained by their model for different inputs:

- 99.01% on 301 writers with one English sentence input.
- 97.03% accuracy on 657 writers with one English sentence input.
- 93.85% accuracy on 300 writers with one Chinese character input.

Oka et al. proposed a handwriting identification technique where instead of using single character or word to predict the writer, line of text can be more effective to get complete characteristics of handwriting. [2]

Major of the work in computer vision is done using convolutional neural network. Pilhyun et al. also designed the twin convolutional neural network to classify the writer using two images. In this architecture, two images are passed through the identical convolutional neural network and after merging both of the output feature vector we could obtain a similarity score. This architecture required large amount of data and data should be pre-processed to increase the accuracy of the system. It could predict the writer with accuracy of 85.5% on a test set where similar architecture is used to the training set. [3]

At the paper written by Yonma et al. proposed a model where they not only use the Convolutional Neural Network (CNN) for their classification, they add another layer, where they use Multiclass-Support Vector Machine (MSVM) which helps in increasing the accuracy. They experimented this method using English/Arabic handwriting samples. This method takes handwritten text images as input and passed to the CNN, which provide feature vector as output and then MSVM classifier is used to classify the writer based on the extracted feature vector. This method predicts the writer with the accuracy of 99.8% where they used two public databases Khatt as an Arabic database and IAM as an English database, which contain 206 writers. [4]

2.2 Previous work and Research on Compact Convolutional Transformer

Dosovitskiy et al. proposed the vision transformer, which can take image as an input and process it to classify the image at their paper "An Image is worth 16 x 16 words: Transformers for Image Recognition at scale". Transformer have gained popularity in Natural Language Processing (NLP) and to extend this concept in computer vision, they designed this transformer model. [5]

Vision transformer requires large amount of data, computation power to process the data and to classify the image. As a solution for this problem and to remove the myth that transformers are "data hungry". Ali Hassani et al. [6] proposed the Compact Convolutional Transformer which looks similar to the vision transformer but with some major changes to reduce the need of large amount of data and computational power. They replaced the simple patching method for obtaining token to convolutional tokenization system which provides more information of spatial orientation and added a sequence pooling layer at the end to obtained result by small dataset and limited resources. Only 0.28M parameters can provide the accuracy of 98.00% on CIFAR-10 dataset which is very small dataset for image classification.

M. Koepf et al. used the vision transformer method for writer identification using the two public dataset (CVL Database and database of the ICDAR 2013 Competition on Writer Identification) as well as a forensic dataset (Write dataset). This is the deviation from current state of the art approach, which is CNN. The patches were extracted using SIFT keypoints technique and the output of the ViT is mean aggregated to obtained the

prediction. Proposed system in this paper achieves a top-1 accuracy up to 99% in CVL and 97% in ICDAR 2013. [7]

3 REQUIREMENT ANALYSIS

3.1 Software Requirement

3.1.1 Python

Python is a high level, versatile and user-friendly general purpose, dynamic programming language where multiple programming paradigms, including object-oriented, imperative and functional programming are supported. It is an open source programming language, which let us work quickly and integrate system easily.

3.1.2 Numpy

Numpy stands for Numerical Python. Numpy is a package to perform numeric programming extension for python. It is the core library to compute scientific calculation in python. Numpy provide an array object which is up to 50x faster than traditional python lists.

3.1.3 Tensorflow

Tensorflow is basically defined as a open source software library, developed by google for various numerical computation using data flow. Tensorflow is used for building and training deep learning models ,facilating the creation of computational graphs and efficient execution on various hardware platforms.

3.1.4 Keras

Keras is an open source library that offers a python interface for artificial neural network(ANN) and acts as interface to TensorFlow library. Keras acts as host tools to simplify programming language in deep neural network area while working with texts and images data. Keras is implemented in various neural-network building block area such as layers ,optimizers and activation functions

3.2 Hardware Requirement

3.2.1 Kaggle

Kaggle provides computational infrastructure comprising two Tesla T4 GPUs, each capable of operating for 30 hours a week or 9 hours per session, and a P100 GPU with

similar usage parameters. Additionally, there is a TPU VM v3-2 available, offering 20 hours weekly or 3 hours per session. The system's RAM capacity stands at 29 GB, supporting computational tasks during 12-hour sessions.

4 SYSTEM ARCHITECTURE AND METHODOLOGY

4.1 System Block Diagram

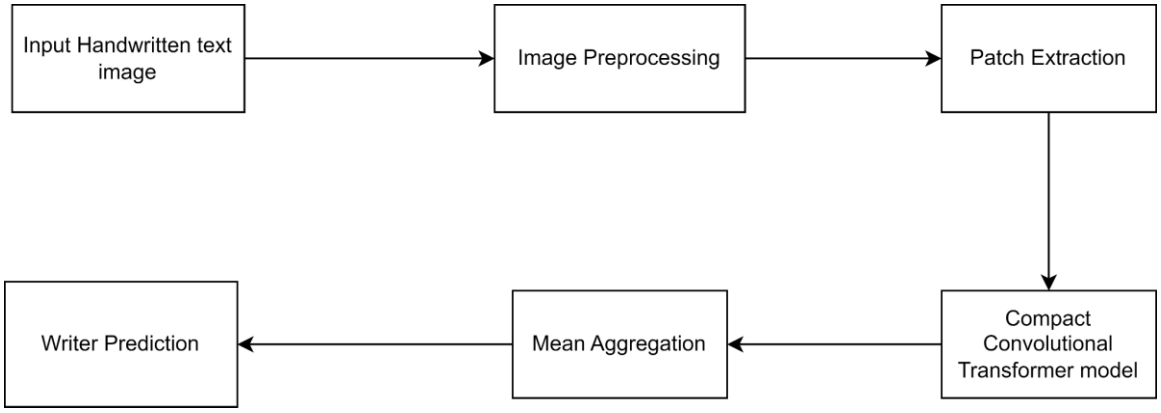


Figure 4-1: Block Diagram of Working of system

4.2 Working Principle

4.2.1 Image Preprocessing

Problems with handwritten text documents can arise from differences in paper quality, lighting, and writing styles. To improve the quality of images before sending them into the Compact Convolutional Transformer (CCT) model for writer identification, preprocessing is essential. The preprocessing includes the followings:-

4.2.1.1 Image Rescaling and Noise Reduction

Rescaling adjust the obtained images to a uniform resolution to guarantee uniformity throughout the collection. This aids in lowering computational complexity. Applying noise reduction techniques such as filtering smoothen the images which aids in improving readability and quality of handwritten texts images.

4.2.1.2 Binarization

Thresholding is used to transform grayscale image in to binary image. The CVT model can more easily concentrate on pertinent information once this stage separates the foreground (text) from the background. One way to deal with lighting differences in different areas of the image is to apply adaptive thresholding techniques. Pixels with

intensity values below the threshold are set to 0 (black), while pixels with intensity values above the threshold are set to 1 (white).

Figure 4-2 Image Binarization

4.2.1.3 Line Extraction

This project requires line cropped images as the input. The input image may consist of multiple lines of text. Each line needs to be segmented and cropped for further proceedings.

Figure 4-3 Binarized Cropped Line

4.2.1.4 Patch Extraction

The line cropped image is further cropped into small segments of fixed width and height. For this project, square segment of 120 x 120 is chosen. The line need to be resized to height 120px before extracting patches.

Figure 4-4 Patch Extraction

4.2.2 Attention

Attention can be describe as a similarity between the query and key. They take the form:

Attention = similarity (q, k)----------(4.1)

Where q represents a query and k represents a key. It's like accessing a database, where we query the database looking for the information we want. To find the similarity between queries and keys, dot product is normally used. It provide the value between 0 and 1. If they are different, obtained value is 0 otherwise it is 1. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

4.2.2.1 Self-Attention

In order to identify dependencies and relationships within input sequences, self-attention is a method employed in machine learning, especially in natural language processing and computer vision applications. By taking care of itself, the model is able to recognize and assess the relative value of various input sequence components. In self-attention, vector q, k, v which are actually neural networks (typically linear) have same input (q(x), k(x), v(x)), then they are self- attending.

4.2.2.2 Scaled Dot-Product Self-Attention

Here, the dimensionality of queries and keys are denoted by d_k and dimensionality of values is denoted by d_v. The scaled dot-product attention then receives these queries, keys, and input values and computes the dot-product of the queries with the keys. The attention scores are then obtained by scaling the result by the square root of d_k. After that, a collection of attention weights is obtained by feeding them into a softmax function. Lastly, a weighted multiplication operation is performed on the data using the attention weights to scale them. The complete procedure can be mathematically stated as follows, where Q, K, and V, represent the keys, values, and queries, correspondingly:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{Q.K^T}{\sqrt{d_k}}\right)V \quad \text{----------(4.2)}$$

4.2.2.3 Multi-Headed Self-Attention

Instead of performing a single attention function with keys, values and queries, it is more beneficial to linearly project the queries, keys and values *h* times with different, learned linear projections to d_k, d_k, and d_v dimensions, respectively. By performing the attention function in parallel on each of these projected versions of queries, keys and values, d_v –dimensional values are obtained as output. The ability of attending to input from several representation subspaces at different points is provided by multi-head attention to the model.

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1,\ldots\ldots\text{head}_h)W^O \text{ ----------(4.3)}$$

$$\text{Where head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

4.2.3 Compact Convolutional Transformer Model

With the rise of transformers for language processing and their advancement in the field of computer vision, there has been significant growth rate in the parameter as well as training datasets. Many of us believe that vision transformer required larger amount of data, it can't perform well for small amount of data. This leads to the concerns such as limited availability of data in different scientific domain and the exclusion of those with limited resource from research in the field.

To overcome this problem, compact convolution transformers is introduced. This assists to avoid overfitting and performs better than CNN for small datasets. This model is flexible to small model size and less parameter with achieving competitive results. [7]

The vector of patches feed into the model is processed by original ViT, with a positional embedding and a classification token. CVT introduces dedicated Sequence Pooling (SeqPool) layer after the final transformer block. It compresses the sequence dimension, before sending the resulting 2D tensor (batch x embedding dim) to an MLP classifier. CCT tokenizes the image using one or more convolution blocks (Conv, ReLu, max-pooling). This strategy better preserves local spatial relationships and introduces the inductive biases from convolutional layers. The number of filters of the last block matches the transformer's embedding dimension (number of tokens).

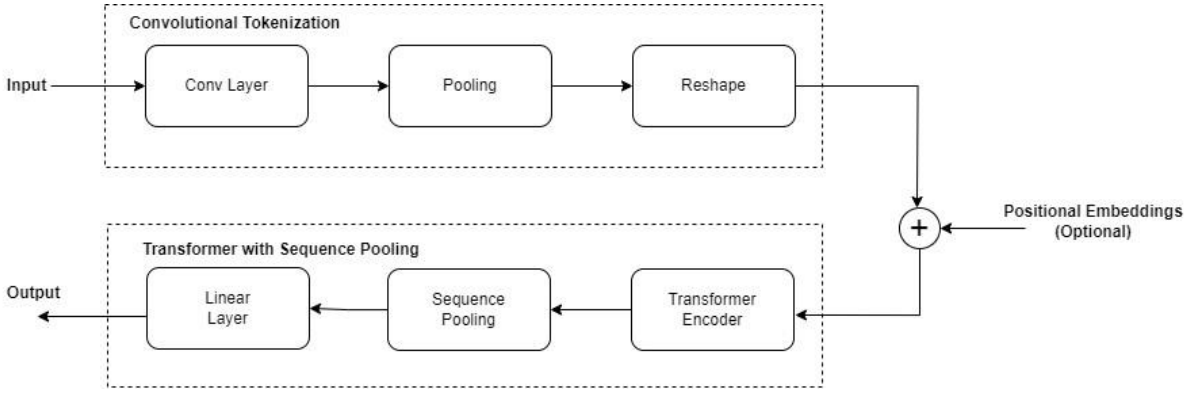


Figure 4-5: Block diagram of CCT

4.2.3.1 Convolutional Tokenization

The method of extracting tokenized representations from the input image by use of convolutional layers is known as convolutional tokenization. Compact convolutional transformers, which integrate convolutional and transformer architectures for processing image data, are mostly dependent on this technology. Compact convolutional transformers may efficiently extract both local and global information from input images, allowing them to perform image classification. The process involves merging convolutional layers for feature extraction with transformer-inspired tokenization methods.

4.2.3.2 Position Embedding

Position embedding is used to add spatial information to the image data. Since the model is actually uninformed of the token's spatial relationship, additional information expressing this relationship can be useful. Usually, this involves assigning tokens weights derived from two high-frequency sine waves, or using a learned embedding. This allows the model to understand that these tokens have a positional relationship.

4.2.3.3 Transformer Encoder

A transformer encoder consist of a series of stack encoding layers. Each encoder layer consists of two sub-layers: Multi-Headed Self-Attention (MHSA) and a Multi-layer Perceptron (MLP) head. Each sub-layer is led by a layer of normalization (LN), and followed by a residual connection to the next sub-layer. The residual connections are a common deep learning technique that allows us to build deeper networks in a more

stable manner. The normalization returns our data back into a normal distribution mean of 0 and variance of 1. Layer normalization assists to stabilize the network and increase our training speed.

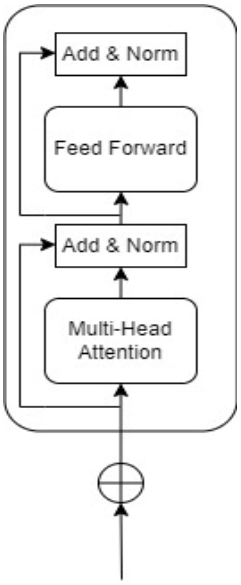


Figure 4-6: Transformer Encoder

4.2.3.3.1 Multi-Head Self Attention

Multi-head self-attention is a key component of transformer-based architectures that are widely used in natural language processing (NLP) and increasingly in computer vision tasks, which includes image classification. Transformers were originally intended to process sequential data, such as text. However, by adding convolutional layers and using self-attention techniques on 2D feature maps, they can be modified to process image data.

4.2.3.3.2 Multi-layer Perceptron

An MLP is a type of feed forward artificial neural network with multiple layers, including an input layer, one or more hidden layers, and an output layer. It is an Artificial Neural Network in which all nodes are interconnected with nodes of different layers. An input layer, an output layer, and one or more hidden layers with several neurons stacked on top of each other make up a multilayer perceptron. Additionally, neurons in a Multilayer Perceptron can employ any arbitrary activation function, unlike

neurons in a Perceptron, which must have an activation function that enforces a threshold, such as ReLU or sigmoid.

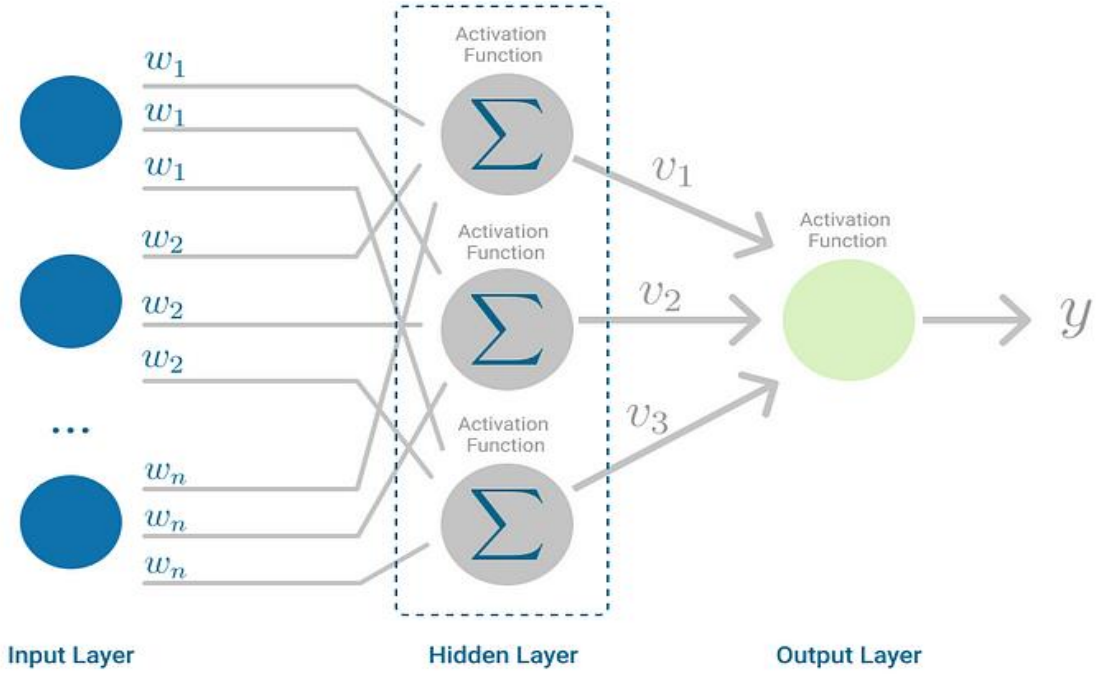


Figure 4-7: Multilayer Perceptron

4.2.3.3.3 Layer Normalization

In deep learning, layer normalization (LN) is a technique that helps to stabilize the training process and boost neural network performance. LN separately normalizes each layer's activations for every feature. This means that the activations are scaled and shifted to have a standard normal distribution (mean of 0 and variance of 1) after the mean and variance of the activations are determined independently for each layer.

4.2.3.4 Sequence Pooling

Sequence pooling refers to the process of aggregating information across the spatial dimensions of the feature maps produced by the transformer Encoder. The goal of this pooling process is to summarize the important information from the feature maps into a small representation that can be used as input for further layers and the resultant may be used for classification or other purposes.

4.2.4 Mean Aggregation

Mean Aggregation is a common approach to summarize sequence data into a fixed-size representation by computing their arithmetic mean, and it can be particularly useful when the length of the input sequences varies or when you need to reduce the dimensionality of the data.

4.3 Activation Function

Activation functions are mathematical equations that determine the output of a neural network model It can also be defined as transfer function. No matter how many hidden layers are attached, every layer behaves the same way since the composite of two linear functions is also a linear function. A non-linear activation function is required to learn the difference between desired output and generated output. Thus the following activation functions have been decided to be used.

4.3.1 Softmax Function

The Softmax function is sometimes called the soft argmax function, or multi-class logistic regression since it is a generalization of logistic regression that can be used for 11 multi-class classification. The softmax function is ideally used in the output layer of the classifier where the probabilities are required to define the input images' class. A softmax function calculates the probabilities of each class which the input belongs. The softmax units in the output layer always be equal to number of classes. The probability distribution is different for different classes and the summation value of all probability distribution is 1 . The softmax function 4.4 provides the probability values for each classes and class with highest probability value is consider as correct prediction.

$$\text{Softmax } \sigma(\vec{z})_{\text{I}} = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_i}} \text{----------(4.4)}$$

Where

$\vec{z}$ = input vector to the softmax function

z_{i} = elements of the input vector

e^{z_i} = standard exponential function applied to each element of the input vector

K = number of classes in the multi-class classifier

The Softmax function is a function that turns a vector of K real values into a vector of K real values that sum to 1. The input values can be positive, negative, zero, or greater than one, but the values are converted to values between 0 and 1 by softmax. Thus now they are interpreted as probability. Small or negative inputs are converted to a small probability value.

4.3.2 ReLU Function

ReLU is the most commonly used activation function in neural networks, whose mathematical equation is:

Relu f(x) = max (0,x) ----------(4.5)

So,if the input is negative, the output of ReLU is 0 and for positive values, it is x.

Relu Derivative $f'(x) = \begin{cases} 1 \; for \; x > 0 \\ 0 \; for \; x \leq 0 \end{cases}$ ----------(4.6)

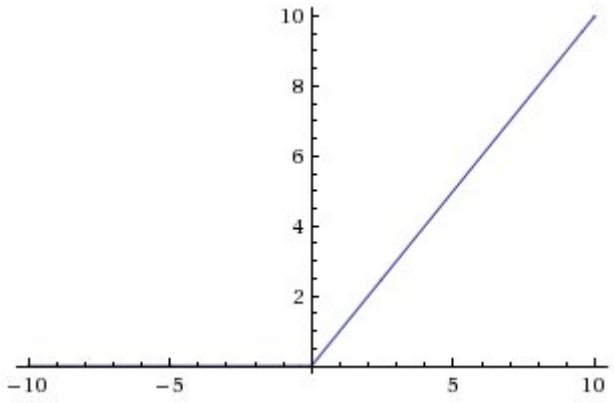


Figure 4-8: Graphical Representation of ReLU function

4.4 Loss Function and Optimizer

4.4.1 Categorical cross entropy

Categorical cross-entropy is a commonly used loss function in machine learning, particularly in classification tasks where the model predicts the probability distribution

over multiple classes for each input sample. The loss is calculated by the following formula:

Loss = $-\sum_{i=1}^{output\ size} y_i . \log \hat{y}_i$ ---------- (4.7)

Where

$\hat{y}_i$ = i[th] scalar value in the model output

y_i = Corresponding target value

output size = the number of scalar values in the model output.

4.4.2 Adam Optimizer

It is a popular optimization algorithm used in training neural network. It stands for Adaptive Moment Estimation. It maintains two moving average vectors: the first moment m (the mean) and the second moment v (the uncentered variance) of the gradients. These moving average are used to adaptively adjust the learning rates for each parameters during training. Due to its simplicity of use, computational efficiency, low memory requirements, and invariance to diagonal rescaling of the gradients, this approach is well suited for scenarios involving high volumes of data and parameters.

4.5 Performance Matrix

4.5.1 Precision

Precision for a given class in multi-class classification is the fraction of instances correctly classified as belonging to a specific class out of all instances the model predicted to belong to that class. In other words, precision measures the model's ability to identify instances of a particular class correctly.

$$Precision_{Class\ A} = \frac{TP_{Class\ A}}{TP_{Class\ A} + FP_{Class\ A}} \quad \text{-------- (4.8)}$$

4.5.2 Recall

In multi-class classification is the fraction of instances in a class that the model correctly classified out of all instances in that class.

$$Recall = \frac{TP_{Class\ A}}{TP_{Class\ A} + FN_{Class\ A}} \text{ ---------- (4.9)}$$

4.5.3 F1 Score

The F1-score combines the precision and recall of a classifier into a single metric by taking the value of their calculated harmonic mean. It is primarily used to compare the performance of two classifiers. Higher the F1 score, the better will be the performance.

$$F1\ Score = \frac{2.Precision\ .Recall}{Precision + Recall} \text{ ---------- (4.10)}$$

4.6 Flowchart

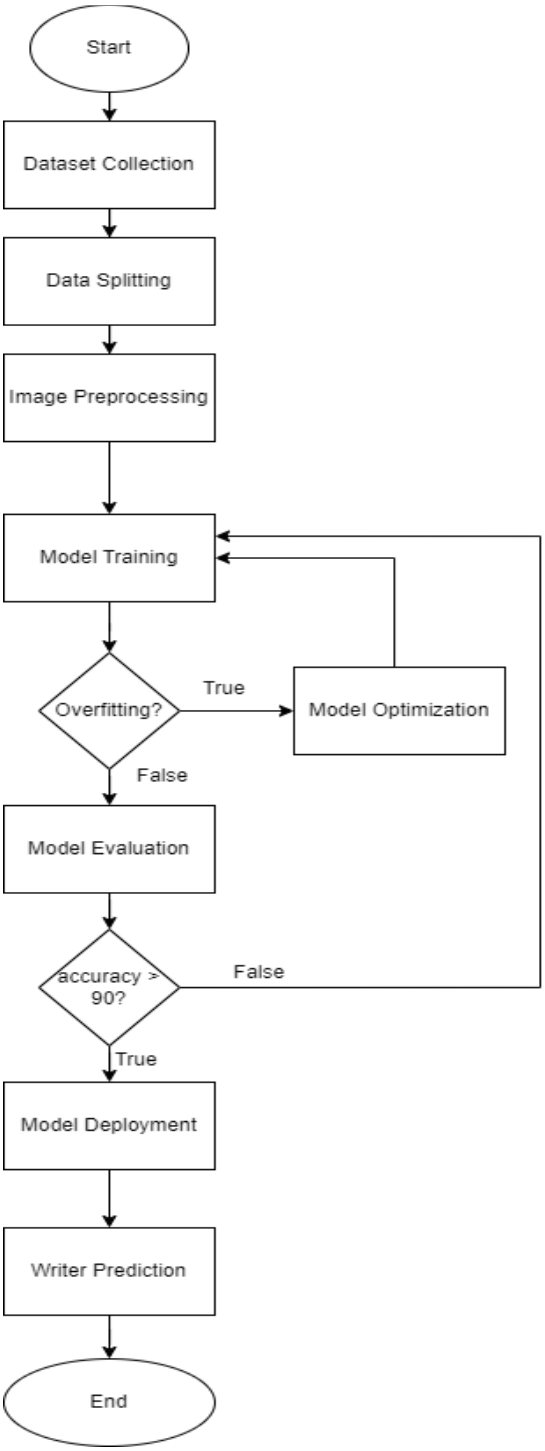


Figure 4-9: Flowchart of proposed system

5 IMPLEMENTATION DETAILS

5.1 Data Collection Mechanism

For developing the model which predict the writer using handwriting, we require a large dataset consisting image of handwritten text along with label to identify its writer. We are going to use some public database such as IAM Database, CVL database and custom made dataset in our project.

5.1.1 IAM Handwritten Database

As we require handwritten text for our project, IAM database is the go-to database. It contains forms of handwritten English text which is best suited to train and test our model. This database provide us with line segmented image which is saved as PNG image. 657 writers have contributed in this database consisting 1539 total pages containing 115320 words. Meta information such as label for the image is provided in the XML file, which can be used to identify the writer for given image. Each line image is organized in the labeled folder based on the writer information on XML file.

5.1.2 CVL Database

We have used the CVL database which is a public database consisting English text written by 310 writers containing 5 page text for each writer. Each text in database is represented as a color image in RGB format, having resolution of 300 dots per inch. Database provide us with unique ID which can be used to identify the writer for given page.

5.1.3 Custom Dataset

We have also created a dataset containing the images of handwritten text from the student of our batch. Dataset contains 41 writer which consist 3 pages of handwritten text. Each page contains minimum of 12 lines. We have collected the handwritten text in A4 paper and extracted line from the pages. Images are properly labelled and pre-processed before using it in training and testing.

Table 5-1 Dataset

Dataset	Number of writers	Number of pages	Number of words	Language
IAM handwriting dataset	657	1539	115320	English
CVL dataset	310	930	69680	English
Custom dataset	41	120	10500	English

5.2 Image Pre-processing

This project aims to predict the writer of the handwritten text using line text. Images in the dataset contain isolated line segments, which are saved in PNG format containing 256 gray levels. Image in the datasets don't require any noise removal but for the images which is present in the custom dataset, we have to remove noises and straighten the text in the image. The height of the images is not uniform. The images are rescaled to fixed height (120px) by maintaining original aspect ratio to make height uniform.

To account for the variation in pen color, the handwritten text is binarized so that there is no color variation. The image is reduced to a single channel image. The image is binarized using image thresholding technique. The threshold value of 180 is found to be suitable for an IAM database. The image is inverted so that the background is black and handwritten text is white. The image is normalized by dividing pixels by 255. Now, the image is represented by binary values only (0 and 1).

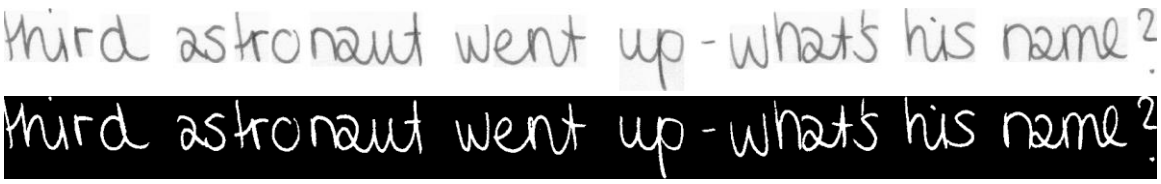


Figure 5-1: Binarization

5.3 Line Removal

As we are creating our custom made dataset to train and test our model, we need to process the image collected from the different sources. As all the images obtained were not in blank paper, some are also in the lined paper so we have to process it before including it in our custom dataset.

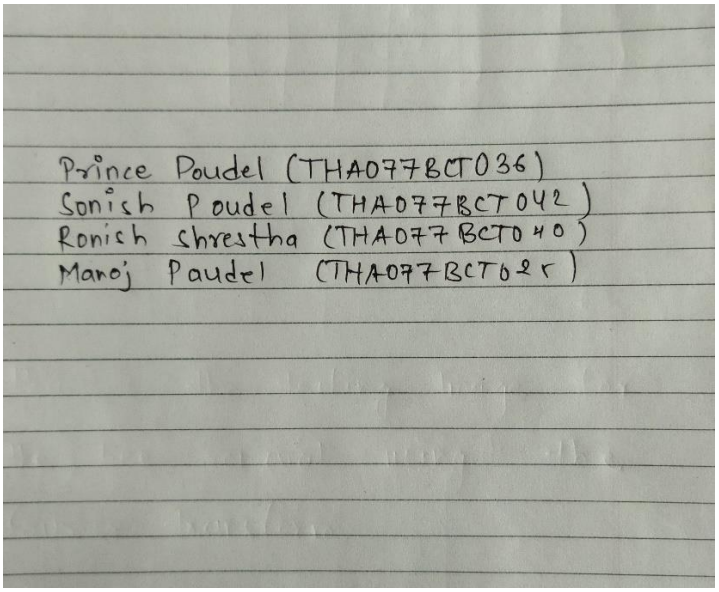


Figure 5-2: Original Image with Lines

If the text is written in the lined paper as shown in the above figure, then we have to remove those lines to extract the required line of text and patches. In order to remove the lines from the page image, we first find out the Fourier transform of the given image using the openCV library. Fourier transform of an image gives the image in the frequency domain. Line is easily distinguishable in frequency domain and they would look like a very bright line.

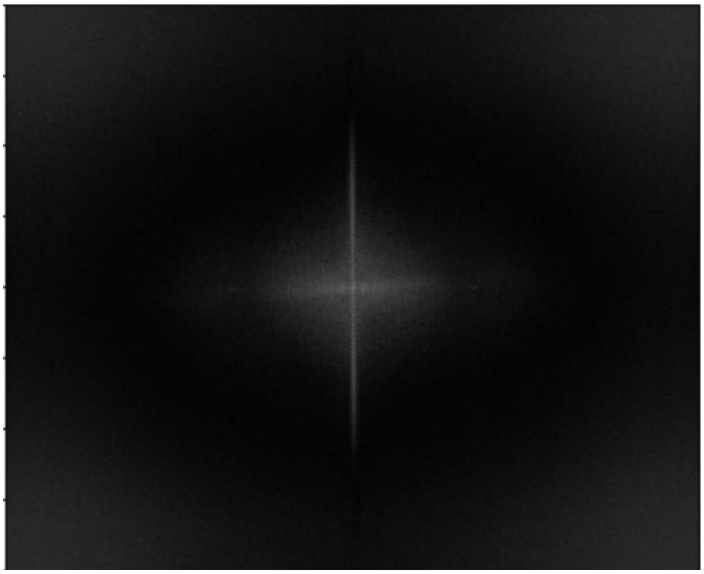


Figure 5-3: Fourier transform of original image

We, then removed the line in frequency domain and then again transform it back to the space domain to get the required cleared image.

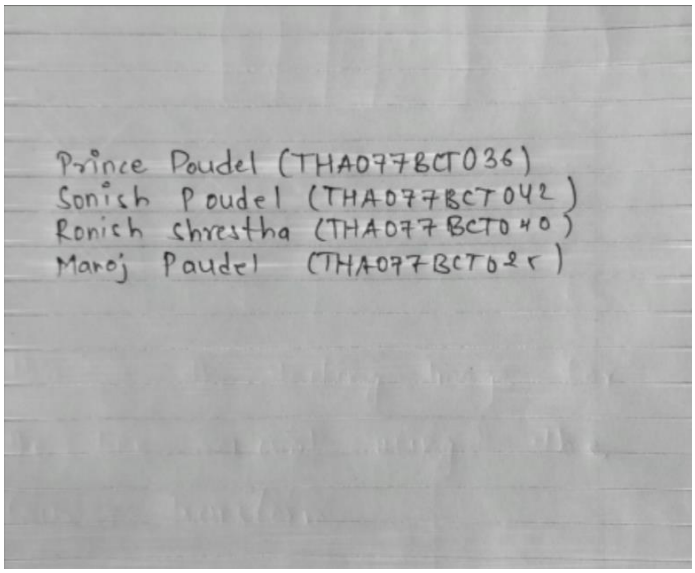


Figure 5-4: Line removed image

Image may contain some noises but after binarization and thresholding, we can obtain the filtered image as shown in figure below.

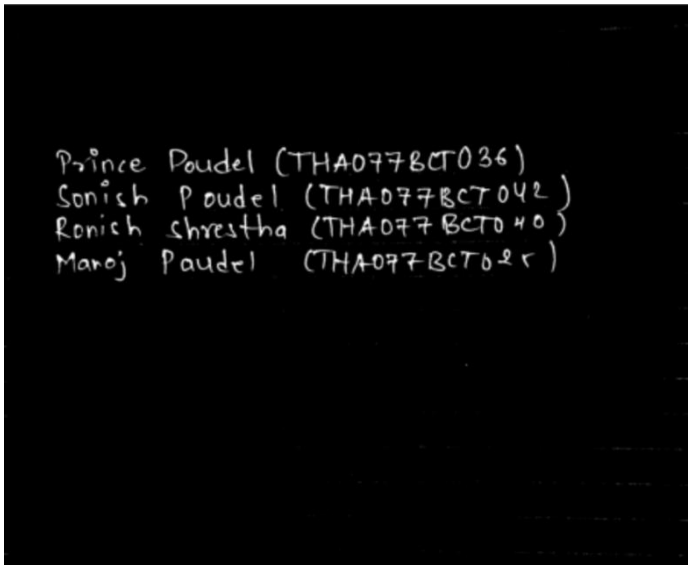


Figure 5-5: Clear Binarized Image after line removal

5.4 Patch Extraction

Rather than using the whole image for classification, the line segment is further cropped into smaller patches. The classification is based on mean aggregation of all the patches. Patches of size 120x120 px are extracted from the line segment.

From the extracted lines, patches of 120*120 are extracted by using sliding window approach, which involves the sliding the window by certain pixel value from left to right until the image is covered. The stride is introduced and initialized with the value which is one fourth the value of the extracted patches.

During training, sliding window strategy is not used. Patches are sampled randomly in each epoch. Random permutations are generated for each line and only 1.5% patches are sampled.

5.5 Implementation of Model

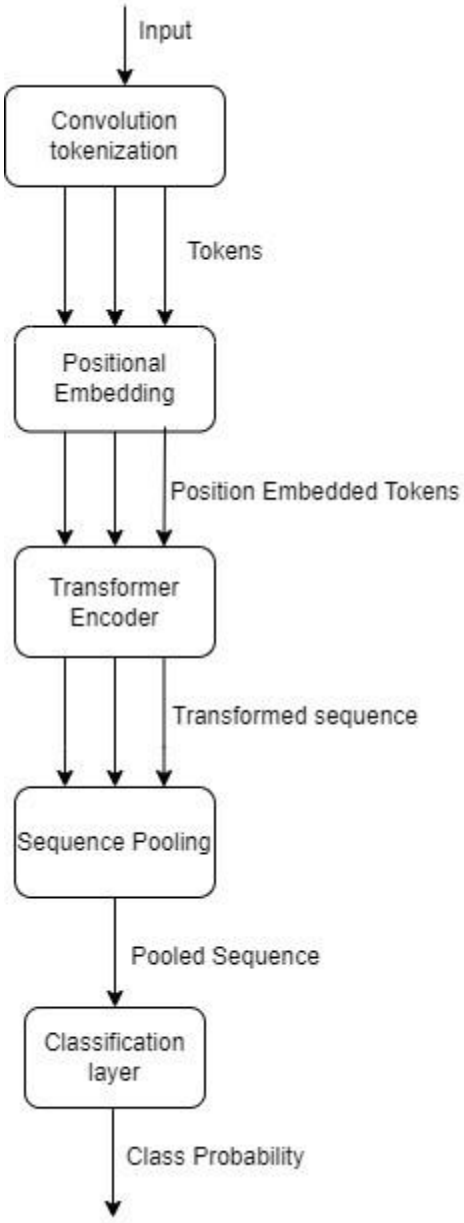


Figure 5-6: Model Implementation Block Diagram

5.5.1 Convolutional Tokenization

In Vision transformer, input image is divided into several patches of equal size which is called patch embedding. But, in compact Convolutional Transformer, we replace patch with a simple convolutional block which can also help in introducing inductive bias into the model that is missing in vision transformer. The convolutional block is

designed using conventional technique, which consist single convolution layer and a max pooling layer.

$$X=\text{MaxPool}(\text{RELU}(\text{Conv2d}(x))) \quad \text{----------(5.1)}$$

Input image of size 120 x 120 is first passed through the convolution layer with 64 filter of size 3 and stride 1, which gives output of size 118 x 118. Then we add the padding which changes the output from size 118 x 118 to 120 x 120. After that, we use MaxPooling layer which changes the (120,120,64) into (60,60,64). Again, we apply convolution layer with 128 filter that gives output of size 58 x 58. Then, we again add padding to change it into (60, 60, 128). After which MaxPooling layer changes it into size of (30, 30,128). At last, we reshape the output from MaxPooling to get 900 tokens of size 128.

Convolutional Tokenization technique used in our model helps in maintaining locally spatial information of the image. Our transformer encoder uses self-attention in which time and space complexity increases as we increase the number of tokens and number of tokens depends upon the resolution of the image. As we down sample the image using convolutional block, computation time and space decreases and we can train our model with limited resources.

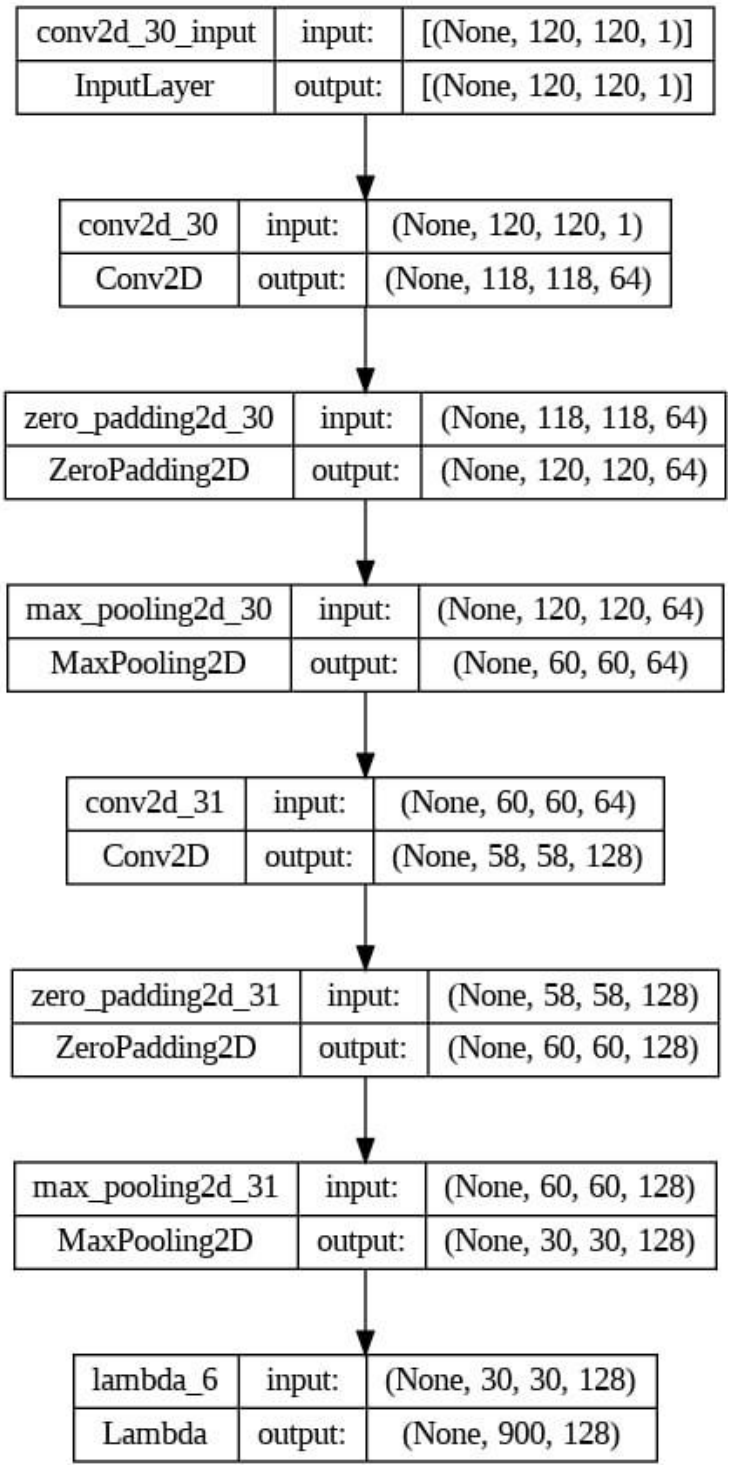


Figure 5-7: Convolution Tokenization layer

5.5.2 Adding Position Embedding

We added the positional embedding to the embedding vector obtained from the convolutional block to retain positional information. Normally, there are two types of positional embedding, learnable positional embedding or fixed positional embedding. We have used standard learnable 1D position embedding as its performance gain is more significant than fixed positional embedding.

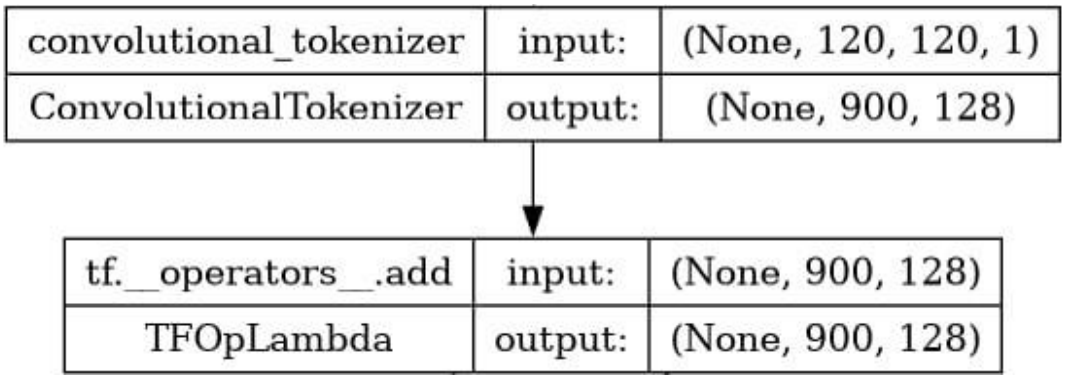


Figure 5-8: Adding Positional Embedding

5.5.3 Layers in Transformer Encoder

We have used 5 transformer layer where each transformer layer contains multi-head self-attention layer, normalization layer and feedforward neural network.

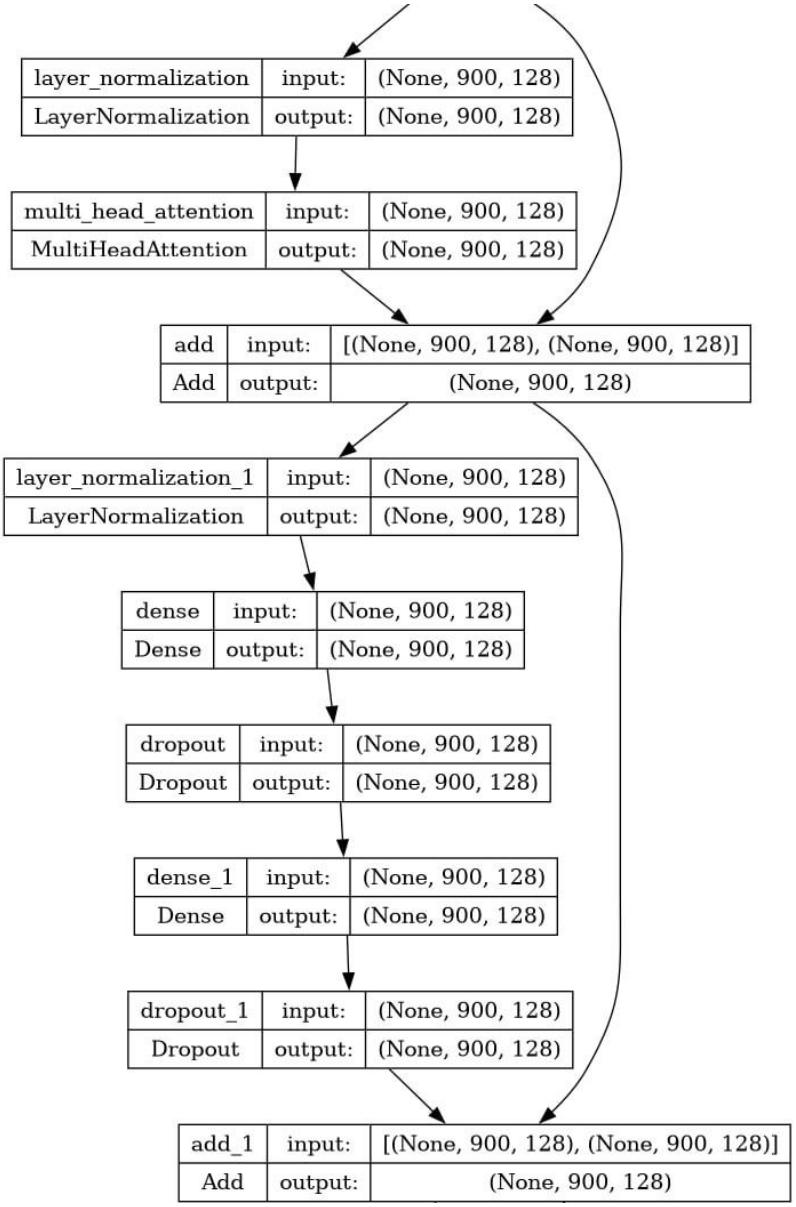


Figure 5-9: Layers in Transformer Encoder

5.5.3.1 Multi-head Self Attention Layer

We are using multi-Head self-attention with 5 head and 128 key dimensions. It contains three vector i.e. queries, keys, and values. Here, each vectors are derived from the input sequence or embedding through linear transformation. In each attention head, the similarity between the query vector and key vector is computed using a dot product. The dot product are scaled by the square root of the dimensionality of the key vectors to stabilize the gradients during training. Then, it is passed through a softmax function to obtain attention weights, representing the importance of each value vector for corresponding query.

5.5.3.2 Multi-Level Perceptron (MLP)

Multi-level perceptron is added after the self-attention layer which helps in introducing the non-linearity into the Transformer Encoder. This allows the model to capture patterns and relationship within data. Our model contains two perceptron layer with 128 inputs each.

5.5.3.3 Skip Connection (Residual Connection)

Skip connection is added after self-attention layer and after multi-layer perceptron layer. A skip connection after self-attention layer is established by adding the attention output with the original input. This connection helps to remove the vanishing gradient problem and improves the flow of information through the network. Another skip connection after MLP is established by adding the output of MLP with the output of the first skip connection.

5.5.3.4 Normalization Layer

Normalization is done before multi-head self-attention layer and multi-level perceptron in our transformer encoder. It helps in stabilizing the learning process and accelerating convergence during training.

5.5.4 Sequence Pooling

We are using sequence pooling, which is an attention-based method that pools over the output sequence of tokens. Sequence pooling contains normalization layer which normalizes the output of the transformer encoder block, then we pass it to linear layer which applies a linear transformation to the normalized output and projects the input features into one dimension. Softmax activation is applied to obtain a probability distribution over the output from linear transformation. Then, weighted average of the input sequence is computed and the obtained result is squeezed into 1-dimensional output vector. We obtain the sequence-pooled output which we will feed into the final classifier.

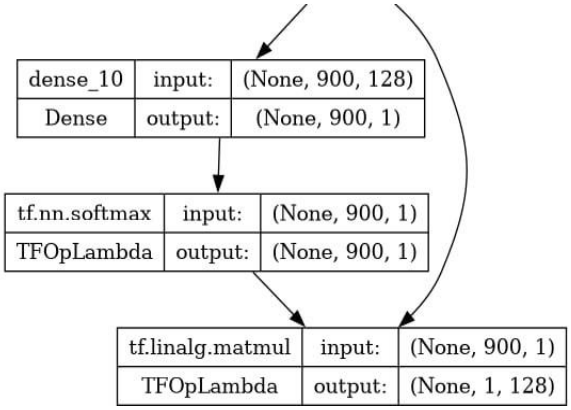


Figure 5-10: Sequence Pooling

5.5.5 Classifier

The pooled vector is connected to the linear layer with softmax activation function which gives output representing the probability for corresponding classes in our dataset.

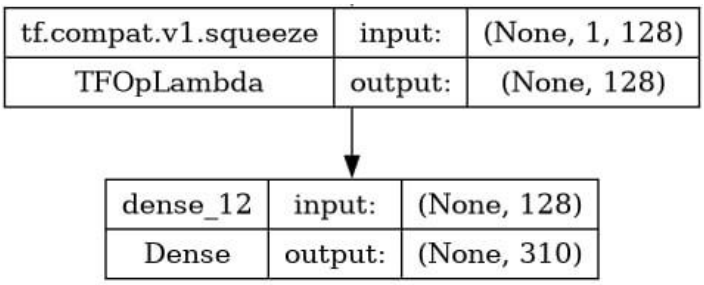


Figure 5-11: Classification Layer

5.6 Training Details

Training and testing set is created by splitting the data. The testing set is 30% of original data. The data is distributed uniformly for all the classes. The data consists of line cropped images for each class.

During each epoch, random 1.5% patches are extracted for each line in the training set. Each patch is of shape (120, 120, 1). The extracted patches are shuffled and provided to model for training in batch size of 32. After each epoch, loss and accuracy is calculated for testing set in same manner.

The loss function used for training the model is sparse categorical cross entropy function. The labels do not need to be one hot encoded. For optimizing the weight, Adam optimizer is used with learning rate of 0.001 and weight decay of 0.0001.

5.7 Testing Details

After training the model, the model is evaluated using testing set. For each line in testing set, patches are extracted using sliding window. Each patch is passed through the trained model. The output is mean aggregated to get probability for each class. The line is predicted to be of class with maximum probability. The performance of the model is evaluated using Top k accuracy and F1 score.

6 RESULT AND ANALYSIS

6.1 IAM Handwriting Dataset

6.1.1 Model 1

This model consists of 5 transformer layers with 5 multi head attention in each layer.

6.1.1.1 Loss and Accuracy

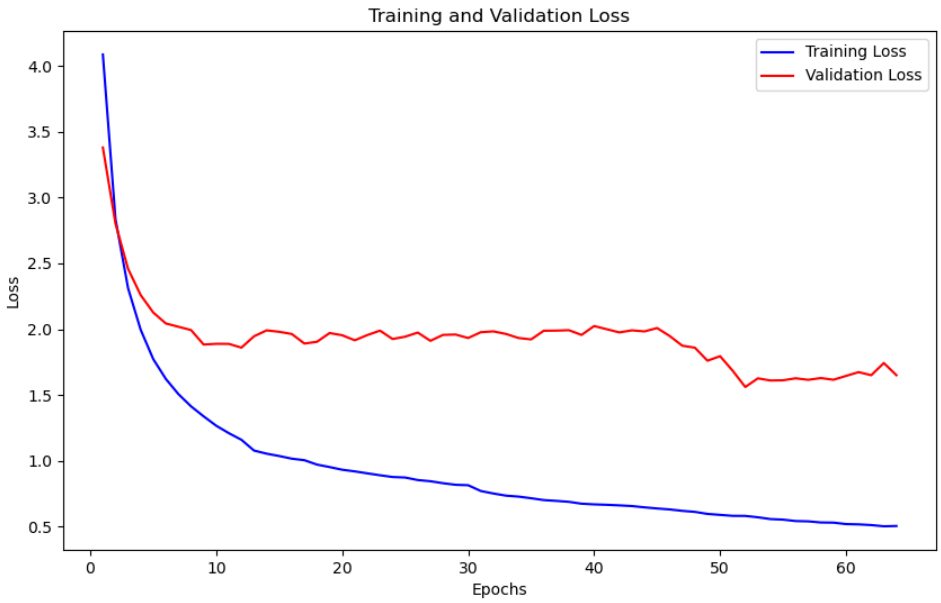


Figure 6-1: Loss vs epoch graph for IAM Dataset (Model 1)

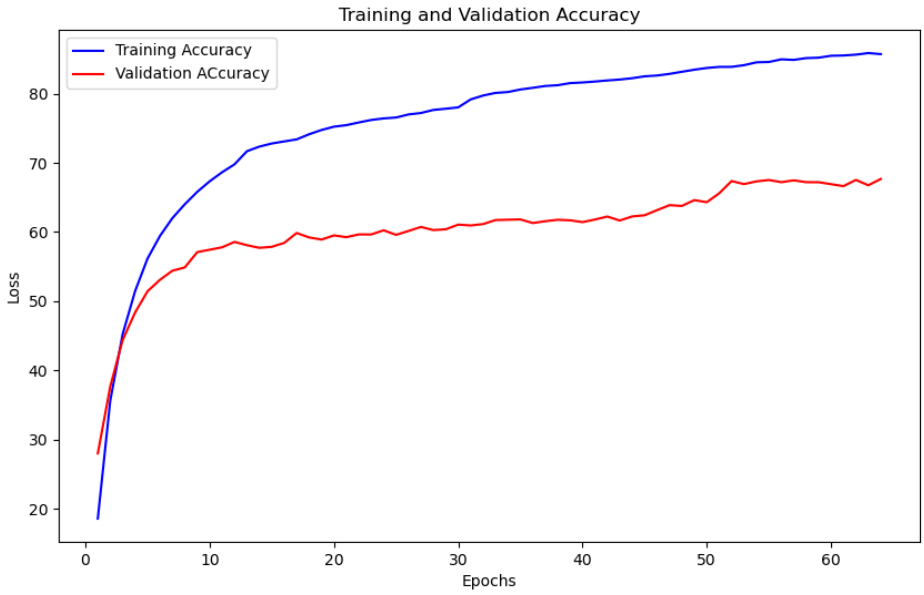


Figure 6-2: Accuracy vs Epoch graph for IAM Dataset (Model 1)

The model was trained for 64 epochs using a batch size of 32. The graph for loss and accuracy during training is shown above. Initially, loss was high and accuracy was low. Validation accuracy and loss was higher than accuracy and loss for the training set. The accuracy increased quickly from 18% to 51% in the first four epochs. After that accuracy increased slowly. Along with increase in accuracy, loss also dropped significantly from 4. The loss and accuracy continue to change for the training set. But for the validation set, there is no significant change in loss and accuracy after the 50th epoch. Training was stopped after 64 epochs due to no significant change in validation accuracy and loss. At the end, the validation accuracy and training accuracy is 66% and 86% respectively.

6.1.1.2 Precision, Recall and F1-Score

In addition, the model was evaluated using precision, recall and f1-score for each class. The average precision, recall and f1-score are 0.943, 0.917 and 0.920 respectively. The 0.920 f1-score indicates that the model is performing well and there is a good balance with precision and recall. The model is able to predict 91.7% of the testing data of different classes correctly. The 94.3% precision indicates the model is good at predicting positive classes.

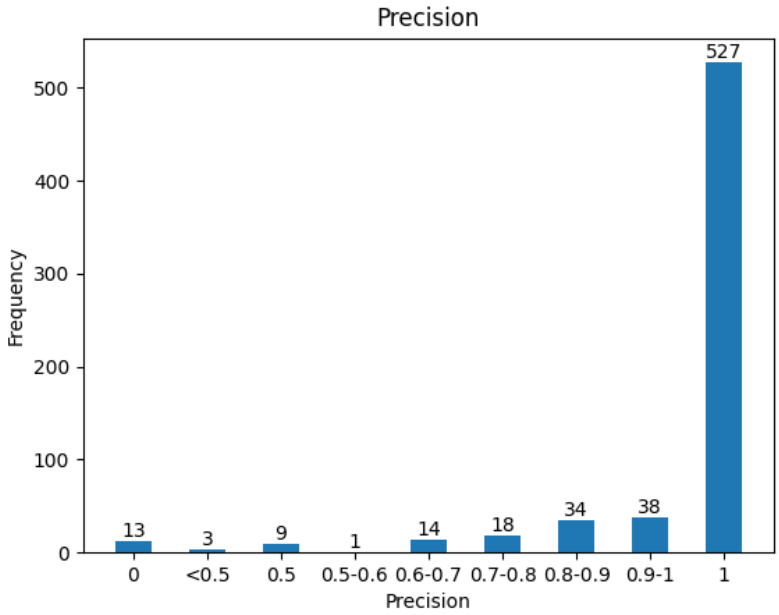


Figure 6-3: Different range of precision values for IAM Dataset (Model 1)

From above graph, model is not able to predict the correct classes of 13 classes. Among 657 classes, 527 classes has precision value 1, all the predicted classes for those classes are accurate. There are only 50 classes which has precision less than 70%.

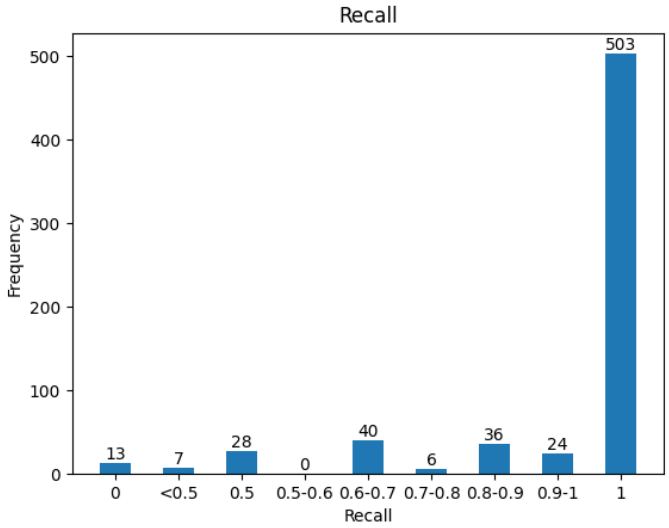


Figure 6-4: Different range of recall values for IAM Dataset (Model 1)

From above graph, model is not able to predict the correct classes of 13 classes. Among 657 classes, 503 classes has recall value 1, this indicates the model is able to predict all the data from those classes accurately. 48 classes has recall value below 60%.

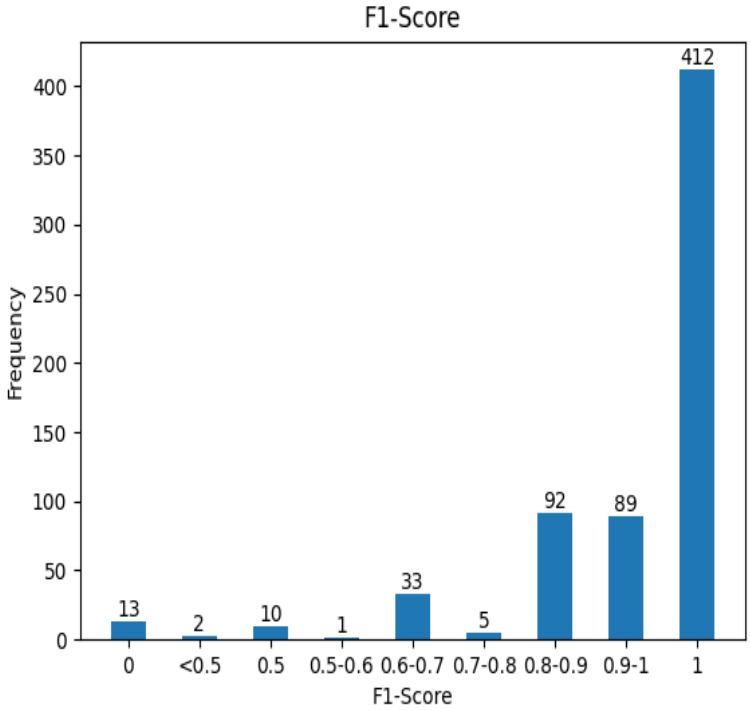


Figure 6-5: Different range of F1-Score values for IAM Dataset (Model 1)

From above graph, most of the classes have good f1 score. The model was able to predict 412 classes without any errors. There are fewer classes (59) below 0.8 f1-score. This indicates good performance of our model.

6.1.2 Model 2

This model consists of 3 transformer layers with 7 multi head attention in each layer.

6.1.2.1 Loss and Accuracy

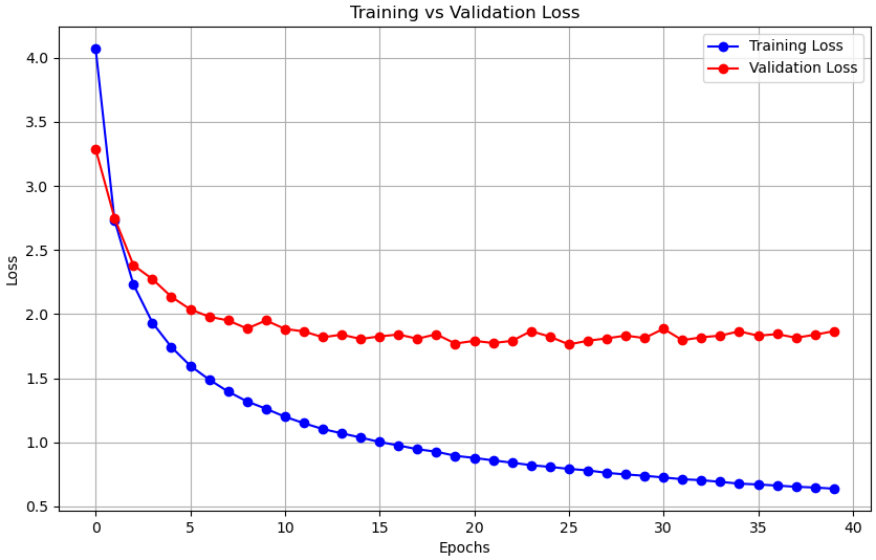


Figure 6-6: Loss vs epoch graph for IAM Dataset (Model 2)

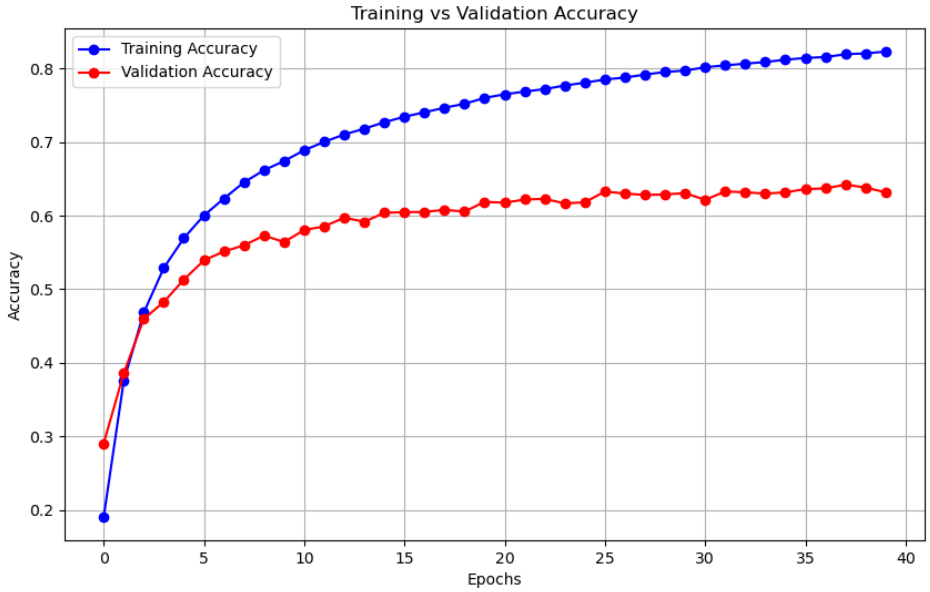


Figure 6-7: Accuracy vs epoch graph for IAM Dataset (Model 2)

The model was trained for 40 epochs using a batch size of 32. The graph for loss and accuracy during training is shown above. Initially, loss was high and accuracy was low. The accuracy and loss changed drastically in first few epochs and rate of change gradually slowed down. For training set, the loss and accuracy continue to drop to better values. But for validation data, there is no much change in loss after 10^{th} epoch. The accuracy continue to change slowly. The training is stopped after there is no significant change in validation accuracy, although accuracy for training data continue to change. At the end, the validation accuracy and training accuracy is 64% and 82% respectively.

6.1.2.2 Precision, Recall and F1-Score

The average precision, recall and f1-score are 0.924, 0.887 and 0.893 respectively. The model is able to predict 88.7% of the testing data of different classes correctly. The 92.4% precision indicates the model is good at predicting positive classes.

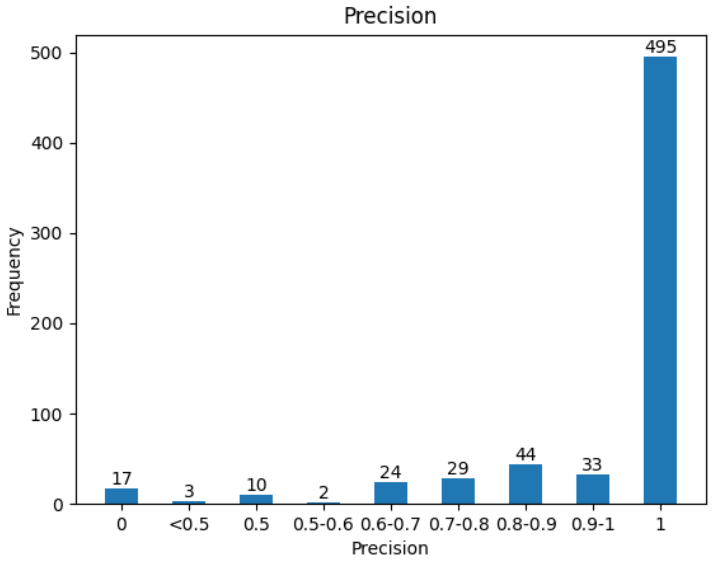


Figure 6-8: Different range precision values for IAM Dataset (Model 2)

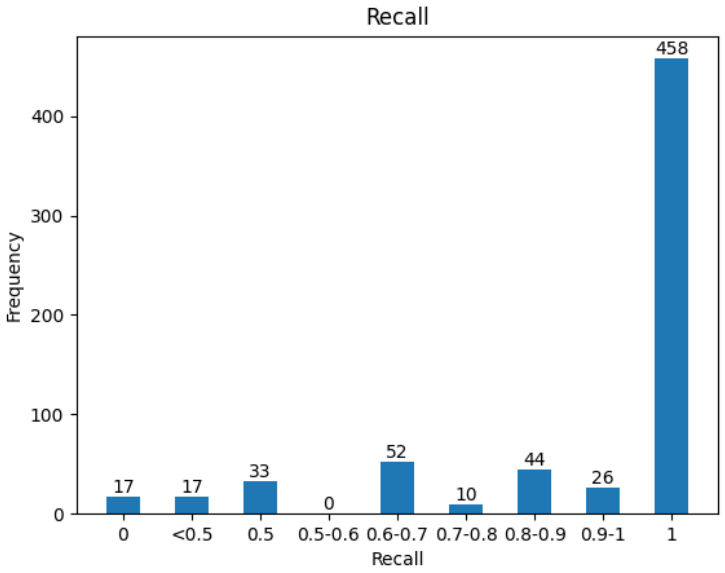


Figure 6-9: Different range recall values for IAM Dataset (Model 2)

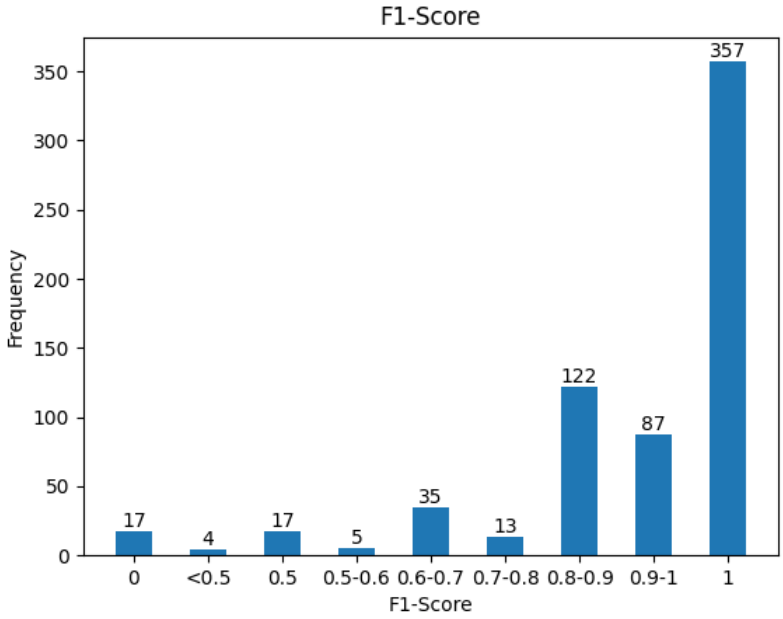


Figure 6-10: Different range F1-Score values for IAM Dataset (Model 2)

The model is not able to correctly identify 17 classes. Among 657 classes, the model is able to predict 357 classes without any false positives. Also, the model was able to predict writers of 458 classes correctly. There are 43 writer classes with the F1-score below 0.6. This indicate that the model is good at predicting writers as most classes has F1 scores above 0.8.

6.1.3 Model 3

This model consists of 7 transformer layers with 3 multi head attention in each layer.

6.1.3.1 Loss and Accuracy

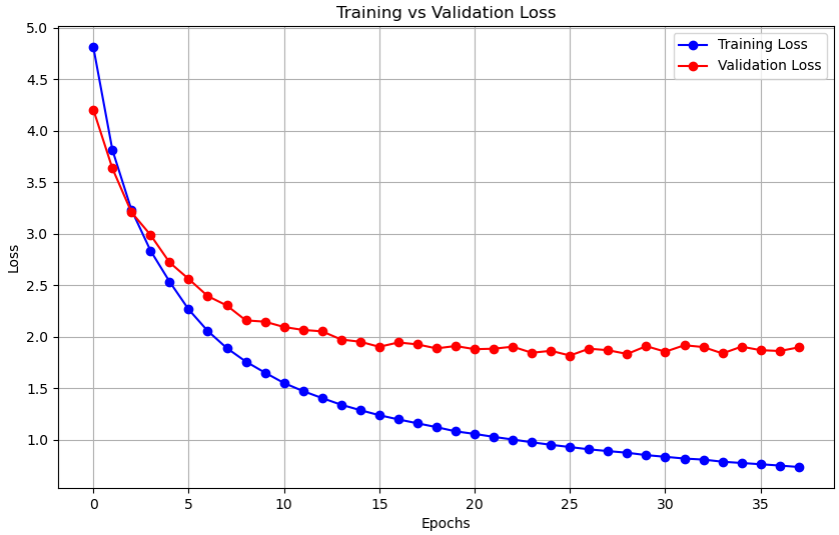


Figure 6-11: Loss vs epoch graph for IAM Dataset (Model 3)

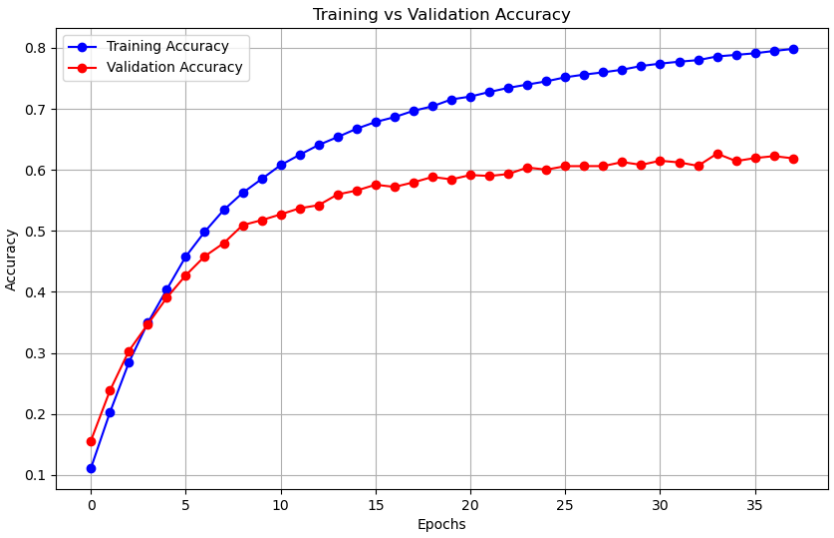


Figure 6-12: Accuracy vs epoch graph for IAM Dataset (Model 3)

The model was trained for 38 epochs using a batch size of 32. The graph for loss and accuracy during training is shown above. The nature of graph is similar as in model 1 and model2 .The training is stopped after there is no significant change in validation

accuracy and loss, even though the training accuracy and loss continue to change. At the end, the validation accuracy and training accuracy is 62% and 80% respectively.

6.1.3.2 Precision, Recall and F1-Score

The average precision, recall and f1-score are 0.926, 0.884 and 0.891 respectively. The model is able to predict 88.4% of the testing data of different classes correctly. The 92.6% precision indicates the model is good at predicting positive classes.

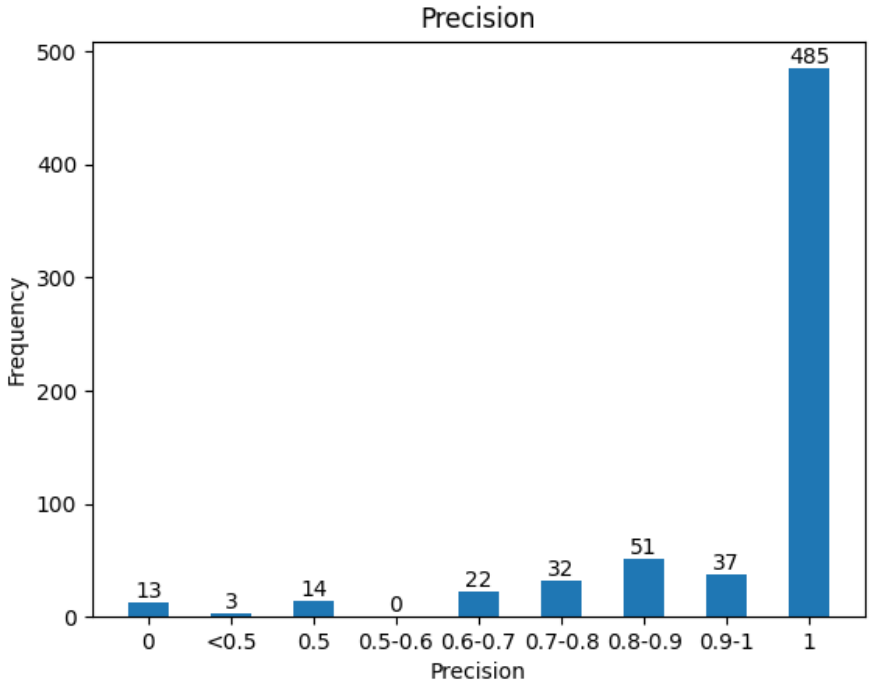


Figure 6-13: Different range of precision values for IAM Dataset (Model 3)

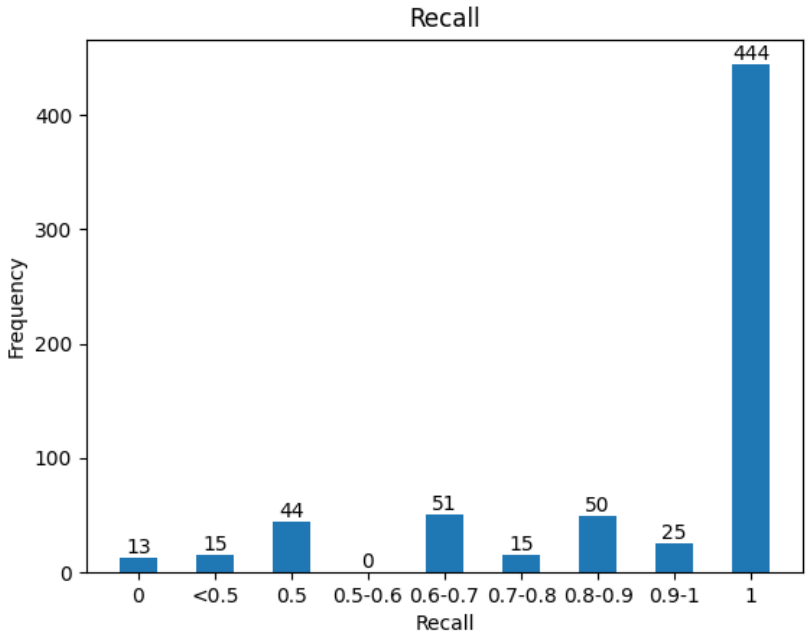


Figure 6-14: Different range of recall values for IAM Dataset (Model 3)

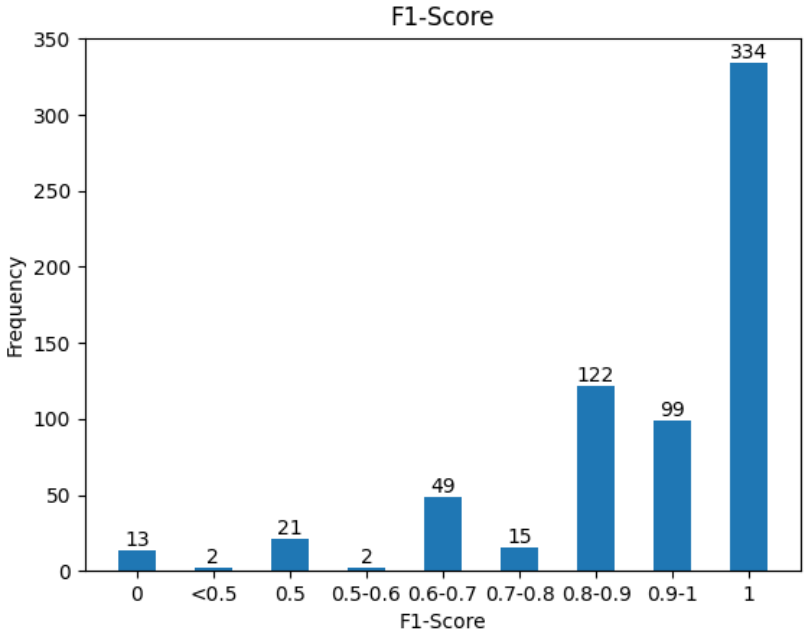


Figure 6-15: Different range of F1-Score values for IAM Dataset (Model 3)

The model is not able to correctly identify 13 classes. Among 657 classes, the model is able to predict 334 classes without any false positives. Also, the model was able to predict writers of 444 classes correctly. There are 38 writer classes with the F1-score

below 0.6. This indicate that the model is good at predicting writers as most classes has F1 scores above 0.8.

6.1.4 Comparison between Model 1, Model 2 and Model 3

So, 3 models are trained on IAM dataset by giving same data. Each model vary in no. of attention head in each layer and no of transformer layers. All three model are able to provide satisfactory result. Model 1 is able to perform better with average f1-score of 0.92 than Model 2 (0.893) and Model 3 (0.891). For each model, there are some classes with 0 F1-score. This is due to unbalanced data size for each class. Data size ranges from min 2 to max 527 lines. Model was not able to perform well on some classes with small data size. Model is not able to work well on some lines with fewer texts. The model will work even better if whole page of text is used to predict the writer.

It is clear that the no of transformer layers and no of multi-attention head affect the result of the model. Choosing appropriate value result in better results.

So, the better model Model 1 is used for training CVL Dataset and Custom Dataset. For training both datasets, the weights of Model 1 is used as initial weight.

6.2 CVL Dataset

6.2.1 Loss and Accuracy

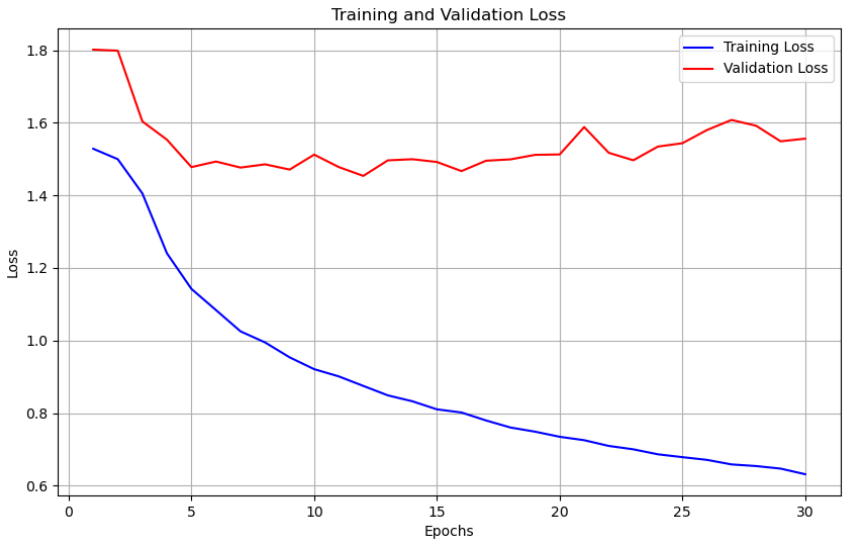


Figure 6-16: Loss vs Epoch graph for CVL Dataset

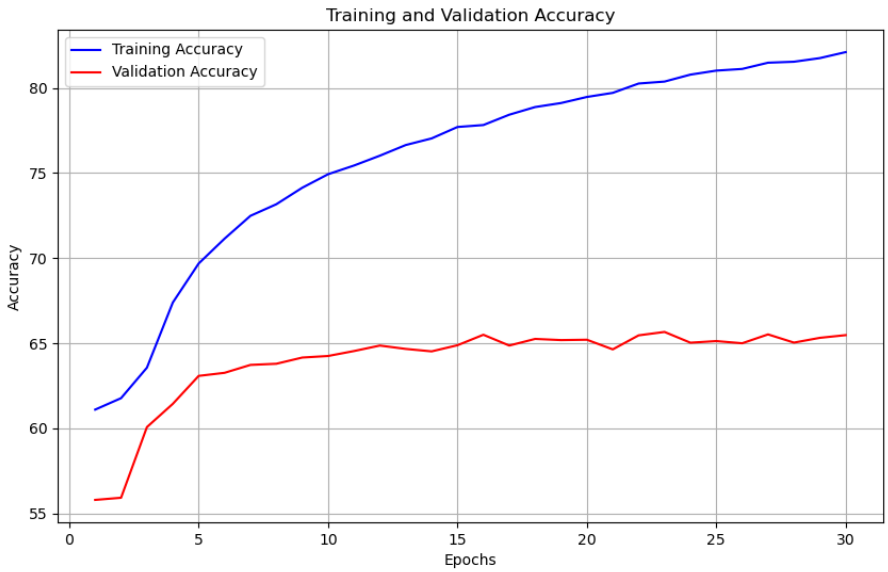


Figure 6-17: Accuracy vs Epoch graph for CVL Dataset

6.2.2 Precision, Recall and F1-Score

In addition, the model was evaluated using precision, recall and f1-score for each class. The average precision, recall and f1-score are 0.979, 0.976 and 0.9707 respectively. The 0.9707 f1-score indicates that the model is performing well. The similar f1-score, recall and precision model is good at predicting classes.

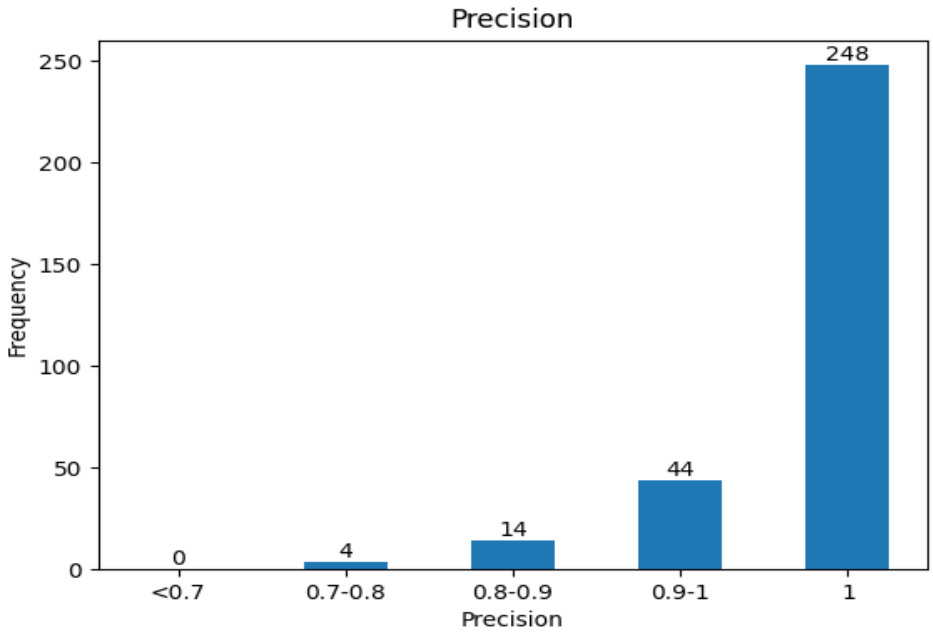


Figure 6-18: Different range of precision values for CVL Dataset

From above graph, the model is able to get good result for all the classes. Most of the classes have precision value 1. There is no precision score below 0.7 which indicates good performance of the model. Only 18 out of 310 classes, have precision value below 0.9.

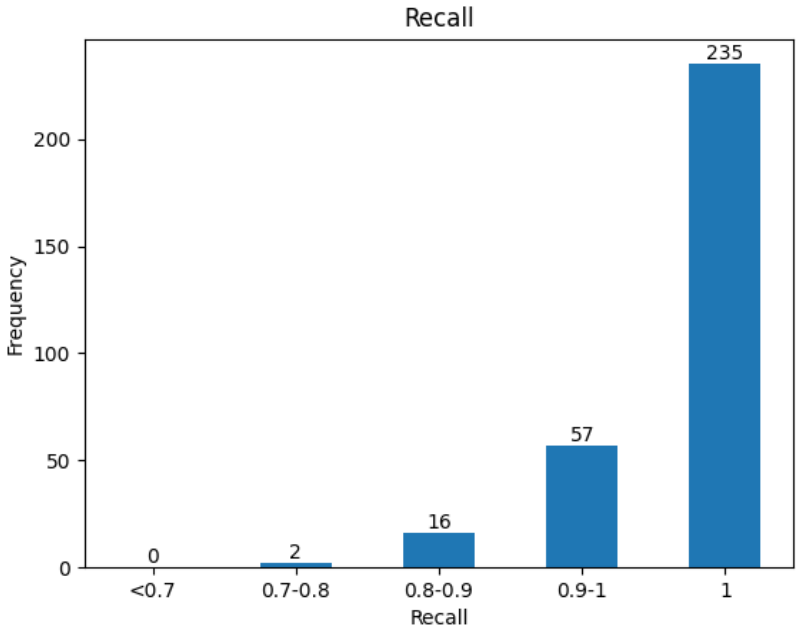


Figure 6-19: Different range of recall values for CVL Dataset

From above graph, the model is able to predict correctly for 235 classes. There are fewer recall scores below 0.9. There is only 2 recall scores between 0.7 and 0.8 and all the classes have recall score above 0.7. This indicates our model is good at prediction.

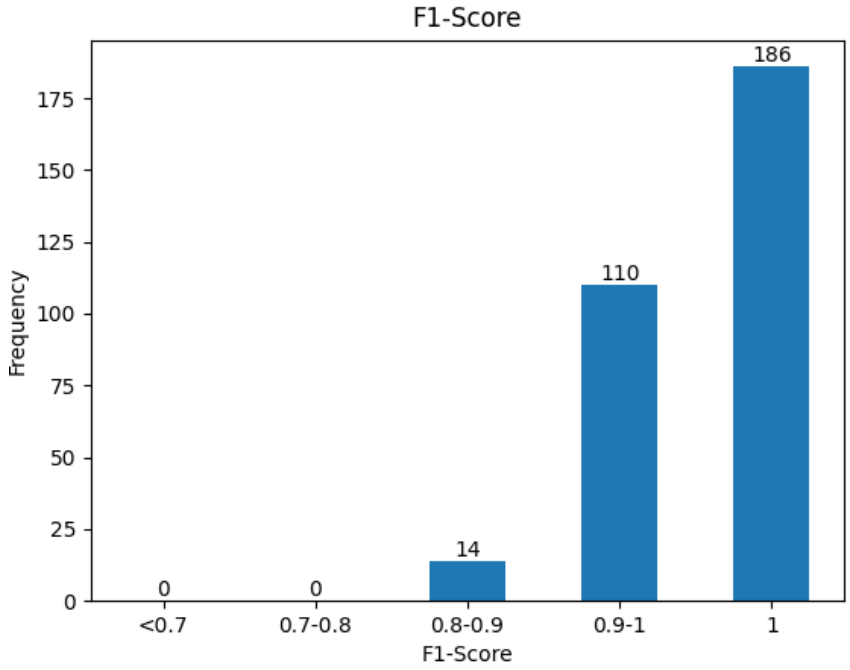


Figure 6-20: Different range of F1-Score values for CVL Dataset

From above graph, most of the classes have good f1 score. The model was able to predict 186 classes without any errors. There are fewer classes (14) below 0.9 f1-score. This indicates good performance of our model.

The model was able to predict all the classes with good precision and recall. The model is perform well for training on CVL dataset than IAM dataset. This is because of balance in data in CVL dataset. 283 writer has 5 texts and 27 writers has 7 texts.

6.3 Custom Dataset

6.3.1 Loss and Accuracy

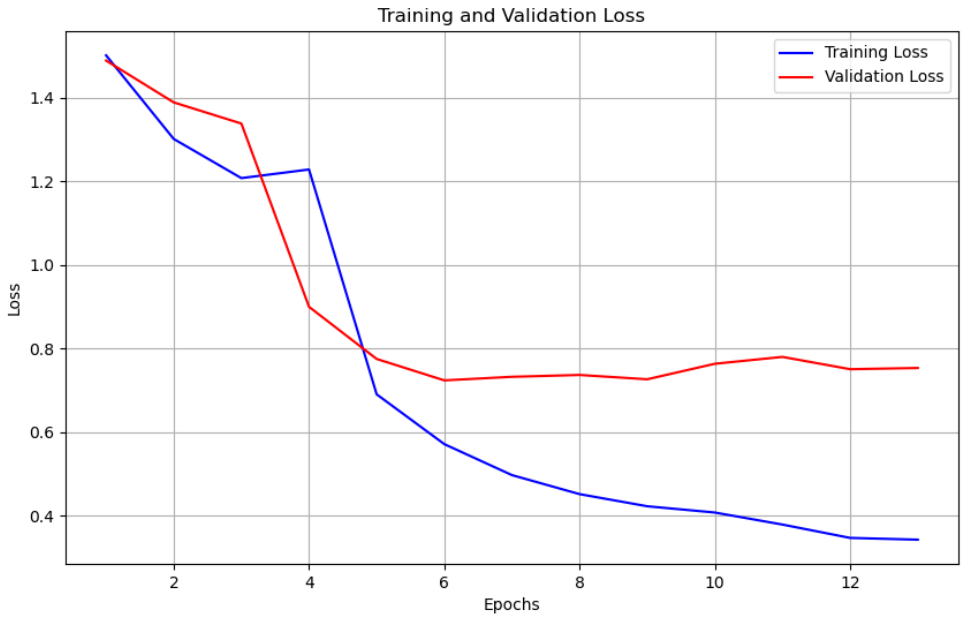


Figure 6-21: Loss vs Epoch graph for Custom Dataset

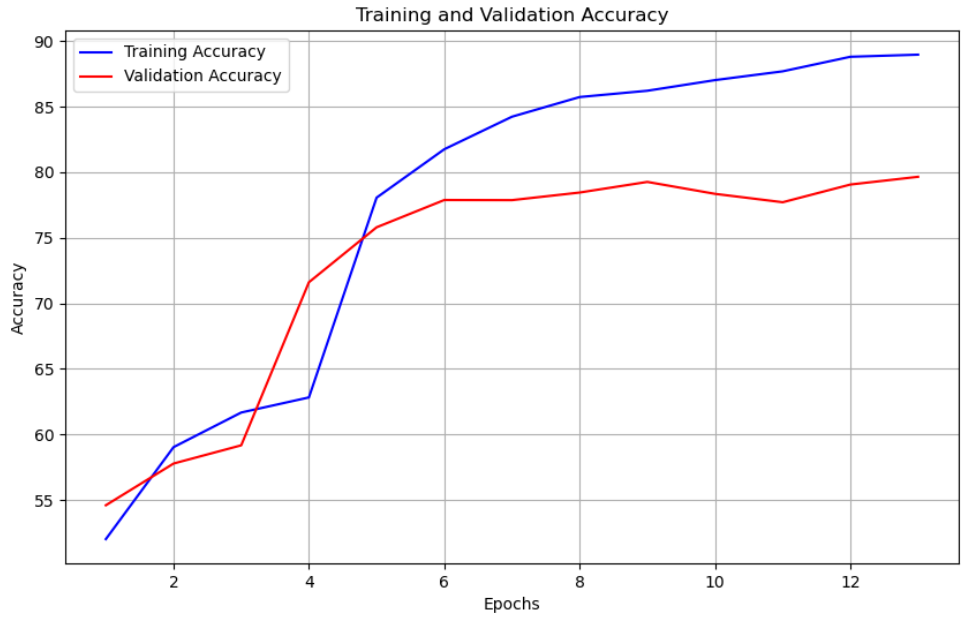


Figure 6-22: Accuracy vs Epoch graph for Custom Dataset

6.3.2 Precision, Recall and F1-Score

Also, the model was evaluated using precision, recall and f1-score for each class. The average precision, recall and f1-score are 0.9869, 0.9875 and 0.9867 respectively. The 0.9867 f1-score indicates that the model is performing well also for training data. The 98% precision, recall and f1-score indicates the model is performing very well.

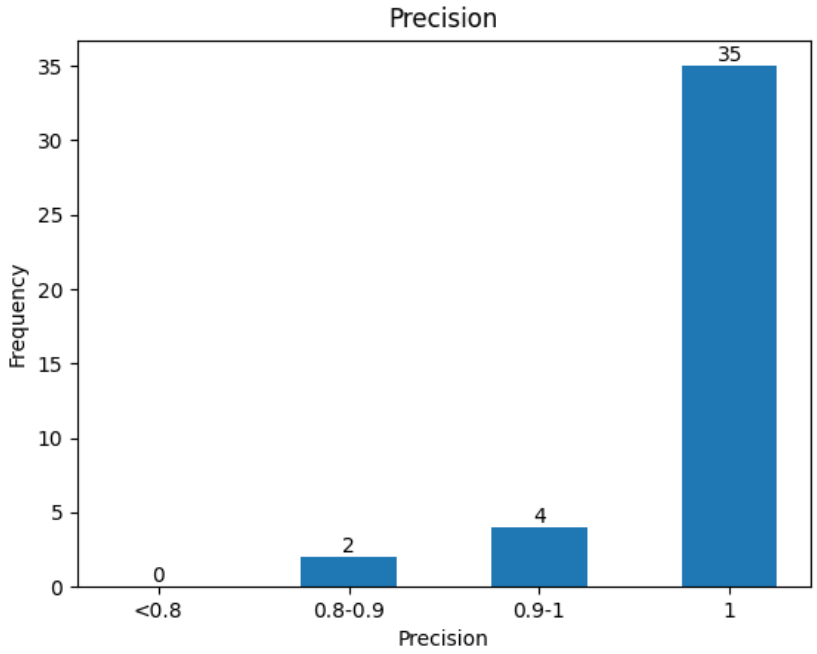


Figure 6-23: Different range of precision values for Custom Dataset

From above graph, the model is able to precisely predict the classes of most of the writers. Only 6 out of 41 classes have precision below 1 and they lie between 0 and 1. So, there is less error in prediction of testing data.

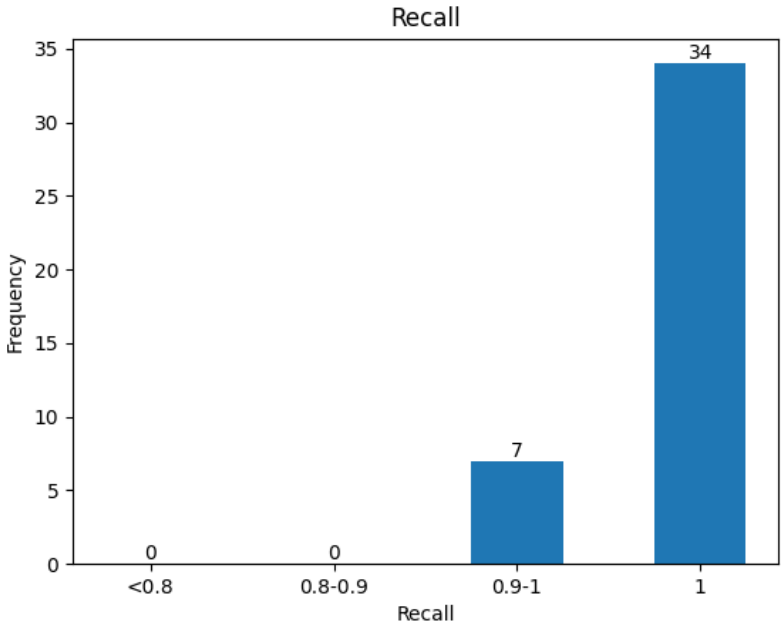


Figure 6-24: Different range of recall values for Custom Dataset

From above graph, all the writers have recall value above 0.9 and only 7 writers have below 1 recall value. This indicates the model is able to perform excellent on testing set which was not used for training.

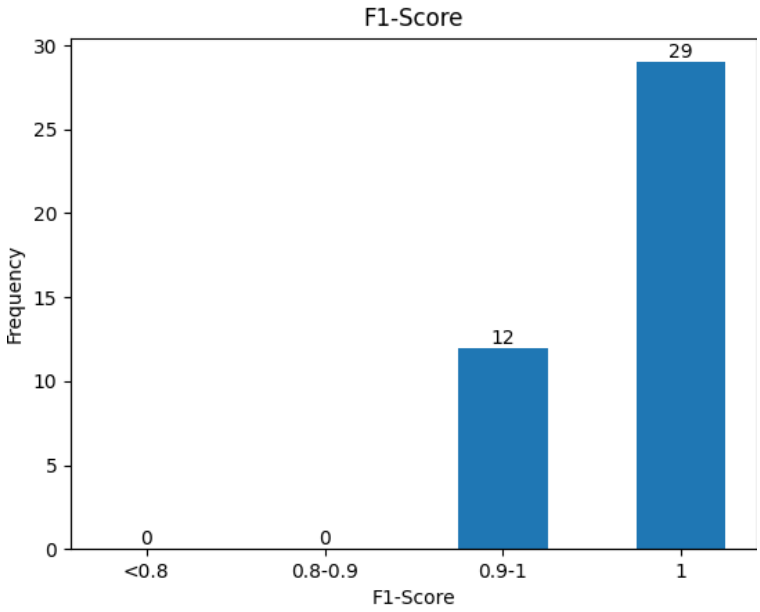


Figure 6-25: Different range of F1-Score values for Custom Dataset

From above graph, all the f1-scores for the writers are above 0.9. This indicates accurate and precise prediction of the writers from the data in training set.

Here, the model is trained on the custom dataset having 41 writers which is less as compared to public datasets IAM and CVL datasets. The model is fine-tuned using the trained model from IAM dataset and took less time. The model is able to predict most of the writings having similar variations from the writer. Adding more variations for each writer will result in good prediction.

6.4 Output of Custom Model

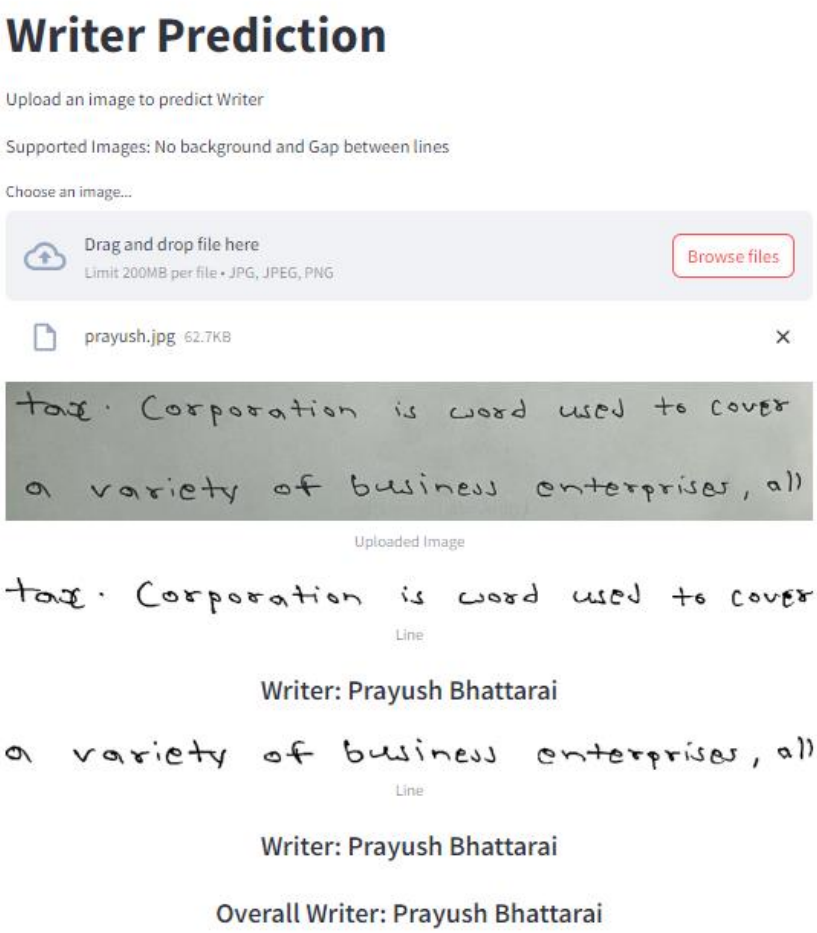


Figure 6-26: Web interface

The model using custom data is deployed for testing. The user must upload the handwritten texts with multiple line. The model predicts and displays the writer for each line. The uploaded image must be clear, properly aligned and should not contain any background. The text must be written in straight manner for accurate results.

7 FUTURE ENHANCEMENS

7.1 Limitations

Some of the limitation of our project are

- Input image must be preprocessed finely.
- Wrongly identifies the data of unseen writer.
- Handwritten text must be in straight line.

7.2 Further Works

The main objective of this project is to predict the writer with high accuracy. But yet our model is not perfectly intelligent to predict writer accurately. So the enhancement that can be done in the future to make this model more accurate and reliable are as follows:

- It shall accept images with small flaws in orientation and lighting.
- It shall identifies the data which are not present in custom dataset.
- It shall be deployed as a mobile application for easy access.

8 CONCLUSION

After exploring convolution neural network for classification task of images, we obtained the highest accuracy for large number of classes of image using Compact Convolution Transformer. It is the compact form of the vision transformer which reduces the number of parameters and decreases the time and space complexity. CCT changes the “data hungry” transformer into compact form which can be train in our local computer, even without use of GPU. We have obtained the accuracy of 95.38% using the IAM dataset, 97.67% using CVL dataset and 98.697% using custom dataset.

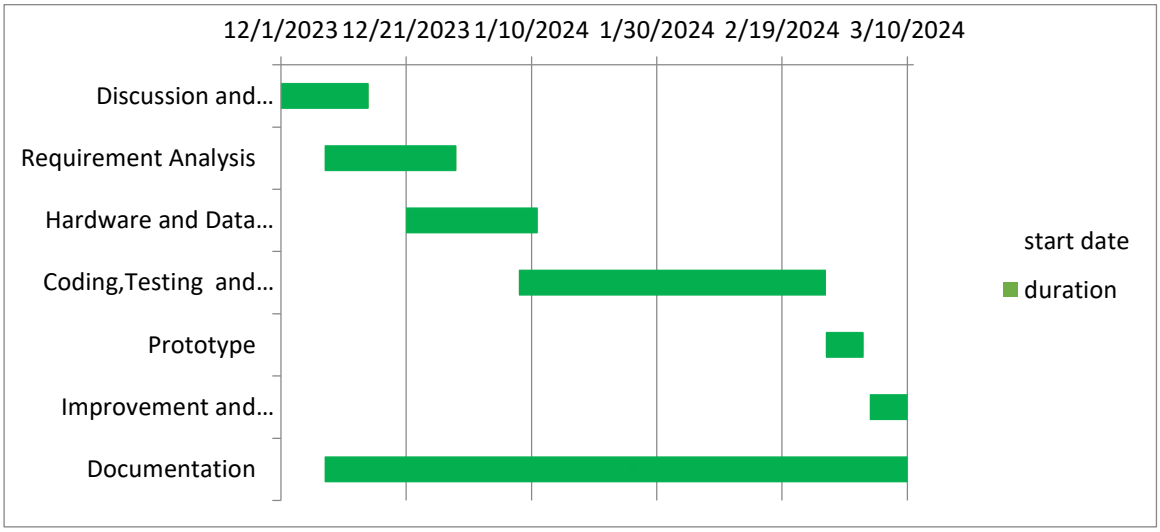
Despite the limitations that exists in our project, this project helped us to learn about transformer, how they can be use not only in Natural language Processing but also in Computer Vision Problem. It also helped us to learn about different image processing steps needed to extract the feature from the image.

Hence, this project work has exhibited a successful output for the course ‘Minor Project’ in the partial fulfillment of the requirements for the award of the Degree of Bachelor of Engineering in Computer Engineering.

9 APPENDICES

Appendix A: Project Timeline

Table 9-1: Gantt chart



Appendix B: Code Snippets

```
def on_epoch_end(self):
    sampling_rate=0.015
    self.datas=[]
    self.labels=[]
    for i in range(len(self.data)):

        h,w,c=self.data[i].shape
        patch_size=h
        total_samples=int((w-h//3)*sampling_rate)
        perms=np.random.permutation(w-h//3)
        for j in range(total_samples):
            st=perms[j]
            patch=self.data[i][:,st:st+h]
              print(patch.shape)
            h,w,c=patch.shape
            if w<patch_size:
                patch=cv2.copyMakeBorder(patch, 0, 0, 0, patch_size-w, cv2.BORDER_CONSTANT, value=[0])
                patch = np.expand_dims(patch, axis=-1)
            self.datas.append(patch)
            self.labels.append(self.label[i])


    self.perms=np.random.permutation(len(self.labels))
```

```
def CompactConvolutionalTransformer(image_size=32,num_classes=10,
                                    input_shape=(32, 32, 3),projection_dim=128,
                                    num_heads=2,L=2,transformer_units=[128,128]):

    inputs = layers.Input(input_shape)

    # Encode patches.
    conv_tokenizer = ConvolutionalTokenizer()
    embedded_patches = conv_tokenizer(inputs)

    # Adding positional embedding.
    pos_embed, seq_length = conv_tokenizer.pos_embeddings(image_size)
    positions = tf.range(start=0, limit=seq_length, delta=1)
    position_embeddings = pos_embed(positions)
    embedded_patches += position_embeddings

    # Feeding embedded patches after adding embeddings to transformer encoder block
    embedded_patches=Transformer_Encoder(L,embedded_patches,num_heads=num_heads
                                         ,projection_dim=projection_dim,transformer_units=transformer_units)

    # Applying sequence pooling to output of the transformer encoder block
    sequence_pooling=SeqPool(embedded_patches)

    # Adding a dense layer for predictions
    output= layers.Dense(num_classes,activation='softmax')(sequence_pooling)

    model = keras.Model(inputs=inputs, outputs=output)
    return model
```

REFERENCES

[1] L. Xing and Y. Qiao, "DeepWriter: A Multi-Stream Deep CNN for Text-independent Writer Identification," arxiv.org, 2016. Available: http://arxiv.org/abs/1606.06472

[2] O. Sudana, I. W. Gunaya, and I. K. G. D. Putra, "Handwriting identification using deep convolutional neural network method," telkomnika.uad.ac.id, 2020. Available: http://telkomnika.uad.ac.id/index.php/TELKOMNIKA/article/view/14864

[3] P. A. Lim and D. M. Ommen, "Twin Convolutional Neural Networks to Classify Writers using Handwriting Data," dr.lib.iastate.edu, 2022. Available: https://dr.lib.iastate.edu/handle/20.500.12876/erLKZYnv

[4] Y. M. Elbarawy and W. A. Ghonaim, "Hybridized Convolution Neural Network and Multiclass-SVM Model for Writer Identification," engineeringletters.com, 2021. Available: http://www.engineeringletters.com/issues_v29/issue_1/EL_29_1_37.pdf

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al "An image is worth 16x16 words: Transformers for image recognition at scale", 2021.Available: https://arxiv.org/abs/2010.11929

[6] M. Koepf, F. Kleber, and R. Sablatnig, "Writer Identification and Writer Retrieval Using Vision Transformer for Forensic Documents," Document Analysis Systems, 2022. Available: https://www.researchgate.net/publication/360644833_Writer_Identification_and_Writer_Retrieval_Using_Vision_Transformer_for_Forensic_Documents

[7] A. Hassani et al., "Escaping the Big Data Paradigm with Compact Transformers," arxiv.org, 2021. Available: https://arxiv.org/abs/2104.05704