TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

PROJECT NO.:THA078MSISE012

ENHANCING HANDWRITTEN TEXT RECOGNITION PERFORMANCE WITH ENCODER TRANSFORMER MODELS

BY

NAGENDRA LAL KARN

A PROJECT REPORT

SUBMITTED TO THE DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF MASTER OF SCIENCE IN INFORMATICS AND INTELLIGENT SYSTEMS ENGINEERING

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

KATHMANDU, NEPAL

AUGUST 2024

ENHANCING HANDWRITTEN TEXT RECOGNITION PERFORMANCE WITH ENCODER TRANSFORMER MODELS

by

Nagendra lal karn

THA078MSISE012

Project Supervisor

Dr. Subodh Nepal

A project submitted in partial fulfillment of the requirements for the degree of

Master of Science in Informatics and Intelligent Systems Engineering

Department of Electronics and Computer Engineering

Institute of Engineering, Thapathali Campus

Tribhuvan University

Kathmandu, Nepal

August, 2024

COPYRIGHT ©

DECLARATION

I declare that the work hereby submitted for Master of Science in Infomatics and Intelligent Systems Engineering (MSIISE) at the Institute of Engineering, Thapathali Campus entitled "**ENHANCING HANDWRITTEN TEXT RECOGNITION PERFORMANCE WITH ENCODER TRANSFORMER MODELS**" is my own work and has not been previously submitted by me at any university for any academic award. I authorize the Institute of Engineering, Thapathali Campus to lend this project work to other institutions or individuals for the purpose of scholarly research.

Nagendra lal karn
THA078MSISE012
August, 2024

RECOMMENDATION

The undersigned certify that they have read and recommend to the Department of Electronics and Computer Engineering for acceptance, a project work entitled "**ENHANCING HANDWRITTEN TEXT RECOGNITION PERFORMANCE WITH ENCODER TRANSFORMER MODELS**", submitted by **Nagendra lal karn** in partial fulfillment of the requirement for the award of the degree of "**Master of Science in Informatics and Intelligent Systems Engineering**".

Project Supervisor

Dr. Subodh Nepal

Department of Information and Broadcasting

Director

M.Sc. Program Coordinator

Er. Dinesh Baniya Kshatri

Assistant Professor

Department of Electronics and Computer Engineering, Thapathali Campus

August, 2024

DEPARTMENTAL ACCEPTANCE

The project work entitled "**ENHANCING HANDWRITTEN TEXT RECOGNITION PERFORMANCE WITH ENCODER TRANSFORMER MODELS**", submitted by **Nagendra lal karn** in partial fulfillment of the requirement for the award of the degree of "**Master of Science in Informatics and Intelligent Systems Engineering**" has been accepted as a genuine record of work independently carried out by the student in the department.

Er. Umesh Kanta Ghimire
Head of the Department
Department of Electronics and Computer Engineering,
Thapathali Campus,
Institute of Engineering,
Tribhuvan University,
Nepal.

August, 2024

ACKNOWLEDGMENT

This project work would not have been possible without the guidance and the help of several individuals who in one way or another contributed and extended their valuable assistance in the preparation and completion of this study.

First of all, I would like to express my sincere gratitude to my supervisor, **Dr. Subodh Nepal**, for providing invaluable guidance, insightful comments, meticulous suggestions, and encouragement throughout the duration of this project work. My sincere thanks also goes to the M.Sc. coordinator, **Er. Dinesh Baniya Kshatri**, for coordinating the project works, providing astute criticism, and having inexhaustible patience.

I am also grateful to my classmates and friends for offering me advice and moral support. To my family, thank you for encouraging me in all of my pursuits and inspiring me to follow my dreams. I am especially grateful to my parents, who supported me emotionally, believed in me and wanted the best for me.

Nagendra lal Karn
THA078MSISE012
August 2024

ABSTRACT

One crucial technique for transforming scanned documents—whether printed or handwritten—into an editable and searchable format is handwritten text recognition. Because handwritten scripts can vary greatly in complexity, handwritten text recognition (HTR) is an extremely difficult operation to perform. Accurate transcription of handwritten material is a common challenge for traditional HTR methods, particularly when handling a variety of handwritten scripts. Transformer-based models have shown impressive performance in a number of natural language processing applications, such as text recognition, in recent years.

The goal of this study is to improve handwritten text recognition systems' performance by utilising transformer models. The suggested method seeks to capture long-range relationships in handwritten text sequences by taking advantage of the self-attention mechanism built into transformers. This successfully models context and increases recognition accuracy of handwritten text.

Preprocessing handwritten images, optimising transformer designs for HTR tasks, and assessing performance on benchmark datasets are some of the major processes that this project will include. We'll also investigate methods like attention processes and data augmentation to improve the accuracy and resilience of the model even more.

This project aims to show that transformer models can effectively enhance the performance of handwritten text recognition systems through rigorous testing and assessment. With possible uses in text transcription, document digitization, and accessibility tools for people with visual impairments, the work's conclusions and insights can further enhance the field of HTR technology.

Keywords: CTC, CRNN, FR-CNN, HTR, RNN

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

BERT	Bidirectional Encoder Representations from Transformers
BLSTM	Bidirectional Long Short-Term Memory
CPU	Central Processing Unit
CNN	convolutional neural network
DNN	Deep Neural Network
GPT	Generative Pre-trained Transforme
HTR	Handwritten Text Recognition
OCR	Optical Character Recognition
RNN	Recurrent Neural Network
SGD	stochastic gradient descen

1 INTRODUCTION

In the realm of artificial intelligence and pattern identification, handwritten text recognition (HTR) presents a formidable problem. Because handwritten scripts are inherently complicated and variable, effectively transcribing handwritten text remains a daunting problem even with recent advances in image processing and machine learning. Conventional HTR methods frequently depend on manually constructed features and sequence models, which may not be able to handle the variety of writing styles, languages, and contextual subtleties found in handwritten texts.

Transformer-based models have been making waves in the natural language processing industry lately. They have demonstrated exceptional performance in tasks including sentiment analysis, text production, and language translation. Transformers are excellent at recognising long-range relationships within sequences because of their self-attention mechanism, which enables strong contextual awareness and representation learning.

This study investigates the possibility of using transformer models to improve handwritten text recognition systems, driven by the success of transformers in natural language understanding tasks. Our goal is to use transformers to solve important problems in human-to-text recognition (HTR), such as accurately transcribing complicated textual content, managing noisy input data, and handling different handwriting styles.

This study aims to explore how transformer topologies might be customised and optimised for the particular task of handwritten text recognition. In comparison to traditional HTR approaches, we hope to significantly increase recognition accuracy, robustness, and scalability through this research.

1.1 Background

The field of optical character recognition that deals with handwritten text recognition in digital form is called handwritten text recognition. Pen and paper note-taking is still done the old-fashioned way. Those old manuscripts or notes still persist, even with the development of modern record-keeping platforms. These handwritten notes and text vary greatly and are difficult to identify, keep, retrieve, distribute, and edit effectively. As a result, many significant physical documents become lost, deteriorate with age, or become unreadable as a result of improper management [1].

With the use of OCR technology, you can turn a variety of document types—such as scanned paper documents, PDF files, and digital camera images—into editable and searchable data. Through the use of an optical device, this technique enables automatic character recognition. Humans have eyes, which are optical mechanisms. The image that the eyes see provides input to the brain. Each person's capacity to comprehend these inputs differs depending on a variety of circumstances. OCR is a technology that mimics the reading abilities of humans. Despite this, OCR cannot match a person's reading comprehension. Both printed and handwritten text can be recognised using OCR. But the performance of OCR is directly dependent on quality of input documents. OCR is designed to process images that consist almost entirely of text, with very little non-text clutter obtained from picture captured by mobile camera [2].

For a long time, automatic text recognition in documents has been a difficult study topic. The goal of automatic document processing is to digitally transform physical documents. Data extraction from documents, including commercial forms, government records, historical documents, engineering documents, postal papers, tabular forms, bank checks, etc., is the primary application field. [3]. Converting handwritten writing into a machine-readable format is the first step in the recognition of handwritten text. Given the wide variations in handwritten text, it might be challenging to properly recognise handwritten text. In order to effectively extract characters from handwritten documents and convert the handwritten text into a machine-readable format, HTR must be constructed with some key features. These days, the world's demand for technology is enormous due to the rapid expansion in the many software system fields, including machine learning and deep learning. As we have scanned, information from printed books and newspapers is accessible through documents and paper. This information is part of the hard drive storage in the computer system, which allows for easy reprocessing when needed. Typically, this data is saved in picture form, making it challenging to make changes to it. These challenges arise from the many character typefaces used in the computer system. Currently, research on handwritten text recognition includes pattern recognition, artificial neural network and computer vision. This use machine learning algorithms and machine learning is an important application of artificial neural network, which delivers ability of automatically learn and without explicitly programmed it can improve from the experience. Machine learning mainly proceeding development in computer programs

which is access information then used to learn for themselves. In handwritten text recognition acquired and detected characteristics by algorithms from the touch screen devices, picture and after that it converted to machine readable form. Similarly, various industries and fields are using machine learning such as image processing, classification, regression, medical diagnosis, and prediction and learning association [4]. Computer vision has continuously improved as a result of deep learning, and HTR is still one of the main areas of research in this discipline. One significant kind of image-based sequence recognition issue is HTR. Numerous studies have been conducted in this area to enhance HTR performance and accuracy, and new uses for this potent technology are always being investigated [5]. Over the past few years, Deep Learning techniques have shown a significant improvement over conventional techniques for the HTR task. Convolutional neural networks have revolutionised the field of handwriting recognition and achieved state-of-the-art results. The typical convolutional neural network architecture with three convolutional layers is well adapted for the classification of handwritten images [6]. State-of-the-art results for the HTR problem have been routinely achieved by CNN in conjunction with RNN. Convolution operation is used by the CNN's convolutional layer, which applies different filters to extract the image's information. Instead than treating each character separately, the RNN is able to connect their relationship by capturing sequences and contextual information. Compared to a standard RNN design, the Bidirectional Long Short-Term Memory (BLSTM) RNN is utilised to capture long-term dependencies and obtain more context. [7]. The different ensemble strategies have evolved over a period of time which results in better generalization of the learning models. ensemble strategies are broadly categorized as follows: [8]

a) Bagging:

One common method for creating ensemble-based algorithms that are used to improve an ensemble classifier's performance is bagging, often referred to as bootstrap aggregating. Creating a series of independent observations with the same size and distribution as the original data is the basic notion behind bagging. Create an ensemble predictor based on the observations that outperforms the single predictor created from the original data. In the original models, bagging increases two steps: first, creating bagging samples and feeding each bag of samples to the basic models; second, devising a plan for fusing

the predictions from several predictors. Samples for bagging can be produced with or without replacement. Because majority voting is typically utilised for classification problems and the averaging technique is used for regression problems to generate the ensemble result, the combination of the base predictors' outputs may vary.

b) Boosting:

In ensemble models, the boosting strategy is used to transform a poorly generalised learning model into a better learning model. When it comes to classification challenges, methods like majority voting or linear combinations of weak learners work better than a single weak learner to produce better predictions. Many domains have adopted boosting techniques like Gradient Boosting and AdaBoost.

c) Stacking:

Ensemble can be achieved in two ways: either by selecting the "best" base model through some process, or by integrating the results of several base models in some way. One method of integration is stacking, in which the output of base models is integrated using the meta-learning model. Model blending, or just "blending," is the term used to describe the staking process when the final decision is based on a linear model.

In this project, HTR models using Transformer are used to develop ensemble learner model using majority voting technique. Lastly, models were validated using Precision, Recall, F1 Score.

1.2 Motivation

Pen and paper note-taking continues to coexist with more sophisticated computer note-taking. This work is mainly aimed on how can handwritten content on paper documents as well as in digital devices be transformed into a machine-readable format so that it may be electronically saved, modified, and shared.

1.3 Problem Statement

Traditional method of note keeping using pen and paper still co-exist with the advanced electronic way of note keeping. Obviously large handwritten important document still exists, which are of great historical importance and are yet to be digitized with better

accuracy. HTR can convert such printed or handwritten text on paper documents into machine readable format, allowing them to be stored, edited, and shared electronically. Moreover, HTR can speed up the automated process of entering data from paper documents into electronic databases more efficiently rather than doing manually. Visually impaired person can use HTR to access the handwritten text document and converting to audio file. Overall, HTR lays an important role in digitizing and processing handwritten text to machine readable format making it more useful to various field.

1.4 Project Objectives

The primary objectives of this project are as follows:

A. Implement Transformer-based Handwritten Text Recognition (HTR) System

1. Develop and train a Transformer model architecture specifically tailored for handwritten text recognition.
2. Adapt the Transformer's self-attention mechanism to effectively capture long-range dependencies in handwritten documents.
3. Design the model to process input images and generate corresponding text sequences using BERT Transformer architecture.

A. Optimize Model Performance through Hyperparameter Tuning

1. Identify key hyperparameters of the Transformer model that is the number of layers, attention heads, hidden units, and learning rate.
2. Perform systematic hyperparameter tuning experiments using techniques SGD, RMSprop, Adam and Adamax optimizer.
3. Evaluate the impact of hyperparameter configurations on model performance metrics accuracy, Recall, Precison, F1 Score.
4. Select the optimal hyperparameter configuration that maximizes the performance of the HTR system on validation or test datasets.

1.5 Scope of Project

Handwritten text recognition is a challenging task due to the variability in handwriting styles and the lack of clear distinctions between characters. Transformer techniques have shown great potential in addressing these challenges.

Transformer Model Adaptation: The project will focus on adapting transformer architectures BERT (Bidirectional Encoder Representations from Transformers) to the task of handwritten text recognition. This adaptation involves fine-tuning pre-trained transformer models on handwritten text datasets to learn representations suitable for recognizing handwritten content.

Data Preprocessing: The project will involve preprocessing handwritten images to prepare them for input to the transformer model. This may include tasks such as image normalization, resizing, noise reduction, and augmentation techniques to improve model generalization and robustness.

Training and Evaluation: Transformer models will be trained on handwritten text datasets using appropriate loss functions and optimization techniques. The project will employ standard evaluation metrics such as accuracy, precision, recall, and F1-score to assess the performance of the trained models on validation and test datasets.

Performance Comparison: The project will compare the performance of the proposed transformer-based approach with existing HTR methods, such as recurrent neural networks (RNNs), convolutional neural networks (CNNs), and attention-based models. Performance comparison will be conducted in terms of recognition accuracy, speed, and robustness across different handwriting styles and languages.

Exploration of Attention Mechanisms: As transformers rely on self-attention mechanisms to capture contextual information, the project will explore various attention mechanisms tailored to handwritten text recognition. This includes analyzing the importance of different parts of the input image and investigating attention visualization techniques to interpret model decisions.

1.6 Potential Project Applications

The completed project will have enormous uses for both the general public and the fields of image processing and machine learning. These transformer-based systems provide solutions for data entry activities, form processing, and document digitization in industries like government, healthcare, and finance. They also facilitate smooth translation procedures, which is advantageous for applications related to language learning and international communication. Furthermore, these technologies improve accessibility by translating handwritten text into accessible formats such as audio or Braille for people who are visually impaired. In order to promote efficiency in both professional and educational settings, they also provide users with productivity tools for digital note-taking, annotation, and keyword extraction. Transformer-based HTR systems also help scholars, historians, and forensic experts read ancient writings and analyse handwriting types in historical and forensic analysis. Finally, they expedite mail processing in postal services and logistics by automating address recognition. All things considered, the use of transformer-based HTR systems transforms the management of handwritten content, providing accessibility, accuracy, and efficiency for a wide range of disciplines and industries.

1.7 Originality of Project

Enhancing Handwritten Text Recognition Performance with Transformer Models is a cutting-edge technique that boosts the effectiveness and precision of handwritten text recognition by utilising transformer structures, which were first developed for issues pertaining to natural language processing. Although recurrent neural networks and convolutional neural networks have historically been the mainstays for handwritten text recognition, this effort is the first to modify and refine transformer models especially for the difficulties presented by handwritten material. The project aims to capture complex contextual relationships inside handwritten text sequences by exploiting the self-attention mechanism built into transformers. This approach attempts to solve long-standing issues such as various handwriting styles, noisy input data, and scalability limits. In addition to pushing the limits of current techniques, this innovative use of transformer technology for handwritten text recognition creates new opportunities for enhancing the functionality and generalizability of HTR systems in a variety of settings.

1.8 Organisation of Project Report

The first section of the project report deals with the introduction part of the project by defining the terms like background, motivation, problem statement, and objective that are completed throughout the project duration. The second section consists of a literature review on Handwritten text recognition that explains research work done in the handwritten text recognition domain. The third section will describe the overall methodology used in this project to perform project work. It incorporates a block diagram, dataset explanations, and a description of the algorithm that is used in the project work. The fourth section describes the result achieved after completing that project work. The fifth section consists of a discussion and examination of the obtained output of the project. The sixth section of this report describes about future enhancements that may be done to get better result of output. The seventh section deals with the conclusion of the overall project work. Section eight is the appendix section that describes the project time schedule and base papers recording for the literature review. The ninth section is the final section that describes references consulted while doing the project.

2 LITERATURE REVIEW

A subfield of optical character recognition called handwritten text recognition (HTR) focusses on automatically translating handwritten text into machine-encoded text. In contrast to printed text recognition, handwriting recognition (HTR) is more difficult since handwriting styles vary naturally, there is cursive writing, and characters often overlap. Over the years, the field has attracted a lot of attention because of its many uses, which include digital note-taking, automating postal services, and digitising historical records.

Most early HTR initiatives were rule-based and strongly dependent on feature extraction methods. Early studies, such those by Casey and Lecolinet[9], provide a thorough review of the approaches that were already in use, emphasising the application of contour analysis, template matching, and zoning as fundamental strategies for character recognition. These techniques were frequently constrained by their reliance on manually created elements, which found it difficult to take into account the variety of handwriting.

The application of Hidden Markov Models (HMMs) signalled a big change in the area. As explained by Rabiner[10], HMM-based techniques allowed the system to identify patterns in sequential data by modelling handwriting as a series of observable events. Handwriting variation was an area in which HMMs excelled, leading to their widespread use in HTR tasks. Nevertheless, the dependence of HMMs on linearity and their incapacity to identify long-range correlations in handwriting sequences limited their performance.

The use of machine learning into HTR has created new opportunities to enhance recognition precision. In the late 1990s and early 2000s, Support Vector Machines (SVMs), neural networks, and hybrid models that combined HMMs and neural networks became increasingly common. Plamondon and Srihari[11] emphasised the increasing significance of machine learning in handwriting recognition, stressing how these methods contribute to enhancing HTR systems' capacity for generalisation. The development of Long Short-Term Memory (LSTM) networks and Recurrent Neural Networks (RNNs) by Hochreiter and Schmidhuber (1997) marked a major advancement. In particular, LSTM networks—which were created to address the vanishing gradient issue—were

quite good at simulating the temporal dependencies seen in handwritten text. By using LSTMs to accomplish state-of-the-art results in offline handwriting recognition tasks, Graves et al. (2008) showed the effectiveness of LSTMs in HTR.

HTR underwent a radical transformation in the 2010s with the emergence of deep learning. Because convolutional neural networks (CNNs) can learn hierarchical representations from raw pixel input, they were adopted for HTR applications, even though they were first created for image classification tasks. CNNs were first used for handwriting recognition by Lecun et al. [12], who showed notable gains over conventional feature extraction techniques.

Convolutional Recurrent Neural Networks (CRNNs), which are CNNs combined with RNNs, are a further advancement in the area made by Graves et al. [13]. The sequence modelling strength of RNNs and the spatial feature extraction capabilities of CNNs were leveraged by this architecture. Because the CRNN architecture could be taught end-to-end, it eliminated the need for human feature engineering and became a fundamental component of contemporary HTR systems.

Another significant advancement in HTR was the introduction of Connectionist Temporal Classification (CTC) by Graves et al. [14]. Because pre-segmented data was not necessary for the matching of input sequences with output labels, CTC is especially well suited for handwriting recognition problems in which the precise placement of characters is uncertain. In HTR, the CTC loss function gained popularity and constituted the foundation of several cutting-edge models.

Transformer models were first created for Natural Language Processing (NLP), but they have been used more and more for HTR problems in recent years. Vaswani et al.[15] presented the transformer model, which uses self-attention mechanisms to record dependencies between various input sequence segments. Transformers may handle the full sequence simultaneously, in contrast to RNNs, which process data sequentially. This enables more effective computation and better management of long-range dependencies.

There are several ways that Transformers have been modified for HTR. In order to use the spatial feature extraction skills of CNNs and the global context modelling capabilities

of transformers, Baek et al. (2019) presented a novel architecture that blends CNNs and transformers. The outcome was a notable increase in recognition accuracy, especially when the text was written in cursive or was severely deformed.

The practical uses of Handwritten Text Recognition (HTR) in digitising old documents, automating data entry, and improving accessibility have attracted a lot of attention. Initially, predetermined templates of characters or strokes were compared with the input image to detect matches in a process known as Template Matching and Pattern Recognition. Batuhan Balci, Dan Saadati and Dan Shiferaw [1] classify an individual handwritten word so that handwritten text can be translated to a digital form. To complete this objective, the author uses two basic approaches: character segmentation and direct word classification. Various architectures of Convolutional Neural Networks (CNNs) are used to train a model that can precisely classify words. In addition, convolutional Long Short-Term Memory networks (LSTM) are employed to build bounding boxes for every character. Following segmentation, the segmented characters are fed to a CNN for classification, which reconstructs each word based on the classification and segmentation outcomes.

Supriya Das, Purnendu Banerjee, Bhagesh Seraogi, Himadri Majumder, Rahul Roy, Srinivas Mukkamala and B.B. Chaudhuri [3] proposed a novel approach for the classification machine-printed and hand-written text from AEC Documents. Prior to Classification Written by Hand and by MachinePre-processing of the printed text from the documents involves word segmentation, text graphic separation, and binarization. The words in a document are divided into segments according to specific structural characteristics of the Isothetic Covers (IC) that firmly enclose the words. A statistical examination of the related components of the document is used to select the grid size features of the IC. After that, features based on word level Gabor filters are recovered along with spooling data for categorization. The text is classified using a typical classifier that is based on SVM. This task is completed with amazing accuracy at the word level of AEC papers. Xu-Yao Zhang, Yoshua Bengio, Cheng-Lin Liu [5] integrated the deep convolutional neural network (con-v Net) with the domain-specific knowledge of shape normalization and direction decomposition (direct Map) for handwritten Chinese character recognition (HCCR). Author achieves remarkable accuracy for both online and offline HCCR on the

ICDAR-2013 competition database. Savita Ahlawat, Amit Choudhary, Anand Nayyar, Saurabh Singh and Byungun Yoon [6] Rewritten Digit Recognition Using Convolutional Neural Networks (CNN) is highlighted. By employing a pure CNN architecture devoid of ensemble design, Aurther attains accuracy levels that are equivalent. Using the Adam optimizer for the MNIST dataset, the CNN architecture's hyper-parameters are fine-tuned to increase accuracy. Lalita Kumari, Sukhdeep Singh, VVS Rathore and Anuj Sharma [7] introduce Lexicon and Attention based Handwritten Text Recognition System. The author has taken two state-of-the art neural networks systems and merged the attention mechanism with it. Accuracy of 4.15%-character error rate and 9.72%-word error rate on IAM dataset, 7.07%-character error rate and 16.14%-word error rate on GW dataset is achieved. Sudharshan Duth and Amulya [16] Presented Optical Character Recognition for Handwritten and Written Cursive Text Recognition. The author recognised handwritten and printed cursive writing in the following styles: San-Serif, Tahoma, Comic Sans, and Calibri, using OCR, the k-nearest neighbour technique, and a classifier. The method identifies the aforementioned style's text in both upper- and lowercase with remarkably high accuracy. R. Parthiban, R. Ezhilarasi, D. Saravanan [17]A Recurrent Neural Network-Based Optical Character Recognition for English Handwritten Text. The author uses a recurrent neural network OCR that provides remarkably accurate handwritten text recognition. Conda is used in conjunction with Tensor-Flow Framework to implement the model. Jinfeng Bai, Zhineng Chen, Bailan Feng, and Bo Xu [18] Deep convolutional neural networks trained on a variety of languages are used to recognise characters from highlighted images. For character recognition in images, the author employs shared-hidden-layer deep convolutional neural networks, or SHL-CNNs. The design of the network and network training are included in the SHL-CNN framework, from which an appropriate SHL-CNN model for image character recognition is empirically learned. Verification of the taught SHL-CNN's efficacy is conducted on English and Chinese picture character recognition tasks, demonstrating the SHL-CNN's 16–30% reduction in recognition mistakes. Manoj Sonkusare and Narendra Sahu [19]provides an overview of methods for Handwritten Character Recognition (HCR) using the English alphabet. The author outlines the key strategies used over the past ten years in the field of handwritten English alphabet recognition. There is a thorough discussion of the various pre-processing, segmentation, feature extraction, and classification algorithms. The

author points out that even though there have been advancements in methodologies over the years to address the challenge of handwritten English alphabets, more research is necessary before a workable software solution can be made available. The precision of the current handwritten HCR is really low. Therefore, a skillful solution is needed to overcome this challenge in order to maximise overall performance. Prajna Nayak and Sanjay Chandwani [20] presented an Improved Offline Optical Handwritten Character Recognition using Convolution Neural Network and TensorFlow. Author employed a Neural network with tried-and-true layers provides the benefit of a better noise tolerance, resulting in more accurate findings. Numerous strategies for pre-processing, segmentation, feature extraction, and classification are covered in detail. Soft Max Regression is used to assign probability to handwritten characters being one of numerous characters since it produces values between 0 and 1 that add to 1. Kartik Sharma, S.V. Jagadeesh Kona, Anshul Jangwal, Dr. Aarthy M, Dr.Prayline Rajabai C, Dr. Deepika Rani Sona [21] uses the MNIST dataset and observe the variation of the accuracies to classify the digits using deep neural networks. Keras and TensorFlow is used to create a convolutional neural network for classifying handwritten digits. With the use of publicly available GPU resources and computation time, author achieve the considerable accuracy. Flor de Sousa Neto, Byron Leite Dantas Bezerra, Alejandro Hector Toselli and Estanislau Baptista Lima [21] presented a new Gated-CNN-BGRU architecture for offline Handwritten Text Recognition systems combined with two steps of language models. Author used the same methodology for optical models under five known public datasets in the HTR field (Bentham, IAM, RIMES, Saint Gall and Washington). With minimal hyperparameters and trainable parameters, the architecture is simplified. The goal of tuning is to achieve the optimum outcome at the lowest possible cost by using a high learning rate, a low tolerance for early halting, and a reduction on plateau. Using smaller training data sets, Jose Carlos Aradillas, Juan Jose Murillo-Fuentes, and Pablo M. Olmos [22] address the offline handwritten text recognition (HTR) challenge. The author employed deep learning neural networks, which are made up of long short-term memory recurrent units (LSTM) and convolutional neural networks (CNN). Furthermore, the labelling effort is substantially facilitated by the use of connectionist temporal classification (CTC), which prevents segmentation at the character level. Transfer learning (TL) is used from the parameters learned with a bigger database and re-train the whole CRNN

parameters with reduced datasets after the training of the CRNN from the larger database. TL scheme exhibited a good performance when the whole network is initialized and re-trained. Bhargav Rajyagor and Rajnish Rakhlia [23] makes comparative analysis of handwritten Character Recognition using Deep Learning. Supervised layer wise training of a deep convolution neural network, Artificial Neural Network (ANN), Convolution Neural Network (CNN), Deep Neural Network (DNNs), DenseNet, K-NN classifier and Deep Convolution Neural Network are among the deep learning Networks that author makes comparative analysis for hand written character recognition. Zuo Huahong, Tang Junyi and Han Ping [24] highlights Adhesive Handwritten Digit Recognition Based on Improved Faster RCNN. Firstly, the NIST19 dataset is used as the basic dataset, and a mixed dataset is created by setting different hand-to-hand ratios with different degrees of overlap, and then randomly add salt and pepper noise and Gaussian noise in the experimental images. Secondly, aiming at the problem of a large number of overlapping objects in the handwritten digital images, a model based on improved Faster RCNN network is built and trained with the above data sets. Finally, the average accuracy of the model is evaluated. The experimental results show that the average detection accuracy of the Faster RCNN model is good.

HTR has also been investigated for the BERT (Bidirectional Encoder Representations from Transformers) model, which was first created for NLP applications by Devlin et al. (2019). since of its bidirectional attention mechanism, BERT is very good at identifying complex handwriting styles since it can extract contextual information from both the left and right sides of a given character. Although HTR applications of BERT-based models are still in their infancy, these models have demonstrated encouraging performance, especially in word-level recognition tests.

Considerable study has been done on application-specific HTR models in addition to general-purpose models. Fischer et al[25], for example, created an HTR system especially for historical manuscripts, which frequently present particular difficulties including outdated fonts, non-standard spelling, and deteriorated paper quality. Their method addressed the particular difficulties presented by historical documents by combining deep learning models with conventional machine learning approaches. Comparably, an HTR system for multi-writer documents—which are typical in educational contexts when sev-

eral people contribute to a single document—was developed by Bluche et al.[26]. Their hybrid approach combined CNNs and LSTMs, and it included a writer adaption layer that could modify the recognition process according to the writing style of the recognised writer. This method showed how adaptable contemporary HTR systems are to various writing contexts and styles. There is a challenge of HTR using conventional approach, so we approach Transformer encoder model for HTR and improve its performance by using Hyperparameter tuning.

3 METHODOLOGY

3.1 Theoretical Formulations

Establishing the theoretical foundation for the project begins with comprehending transformer model design. Feedforward neural networks and multiple layers of self-attention processes comprise transformers. The self-attention mechanism enables the model to capture long-range links efficiently by assessing the relative worth of several phrases or tokens in a sequence. The theoretical underpinnings of self-attention and its application to modelling contextual information in handwritten text sequences will be examined in this endeavour.

Handwritten text recognition can be framed as a sequence-to-sequence learning issue, where the goal is to map input sequences (handwritten images) to output sequences (translated text). We will look at the theoretical foundations of sequence-to-sequence models and how transformer structures, which are perfect for this kind of work since they can handle variable-length input and output sequences, work.

This research will look at theoretical elements of loss functions that are used to train transformer-based HTR models. We will examine common loss functions such as categorical cross-entropy and sequence-to-sequence loss, along with optimisation methods like Adam and stochastic gradient descent (SGD). Training needs to be predicated on a theoretical comprehension of these components in order to minimise recognition error and optimise model parameters.

This concept will look at the fundamentals of transformer attention processes and how they gather pertinent data from input sequences. The goal of the research is to advance knowledge of various attention mechanisms and the theoretical ramifications of those mechanisms for improving handwritten text recognition. Among these tactics are self-attention and multi-head attention. To assess the significance of various input visual components during recognition and to comprehend model conclusions, attention mechanisms must be understood.

Using transfer learning and fine-tuning techniques, the research will alter transformer models that have already undergone training for the task of handwritten text recognition. To shed light on how well-trained transformer models could be used to improve perfor-

mance on specific tasks with a lack of label information, we will examine theoretical aspects of transfer learning, such as parameter freezing and domain adaptation.

3.2 Mathematical Modelling

The Transformer model can be modeled using equations that describe its various components, such as self-attention mechanisms, feed-forward networks, and layer normalization.

3.2.1 Self-Attention Mechanism

1. Query, Key, Value Transformations:

$$Q = X \cdot W_Q \quad K = X \cdot W_K \quad V = X \cdot W_V$$

where W_Q, W_K, W_V are weight matrices.

2. Scaled Dot-Product Attention:

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

where d_k is the dimensionality of keys.

3. Multi-Head Attention:

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W_O$$

where each head head_i is computed separately, and W_O is the output projection matrix.

3.2.2 Position-wise Feed-Forward Networks

1. Feed-Forward Layer:

$$\text{FFN}(X) = \text{ReLU}(XW_1 + b_1)W_2 + b_2$$

where W_1, b_1, W_2, b_2 are weight matrices and biases.

3.2.3 Layer Normalization

1. Layer Normalization:

$$\text{LayerNorm}(X) = \frac{X - \mu}{\sqrt{\sigma^2 + \varepsilon}} \odot \gamma + \beta$$

where μ and σ are the mean and standard deviation, γ and β are learnable scale and shift parameters, and ε is a small constant for numerical stability.

3.2.4 Positional Encoding

1. Positional Encoding:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right)$$

where *pos* is the position and *i* is the dimension of the positional encoding.

3.2.5 Transformer Encoder

1. Transformer Encoder Layer:

$$\text{EncoderLayer}(X) = \text{LayerNorm}(X + \text{MultiHeadAttention}(X, X, X) + \text{FFN}(X))$$

3.3 System Block Diagram

The suggested system's block diagram is divided into five main sections: feature extraction, sequence modelling, recognition, data preprocessing from IAM data sets, and model evaluation. IAM data sets are the source of the data. With CNN model, feature extraction is carried out. Transformer is used to recognise handwritten text. Finally, hyperparameter adjustment improves the model's performance.

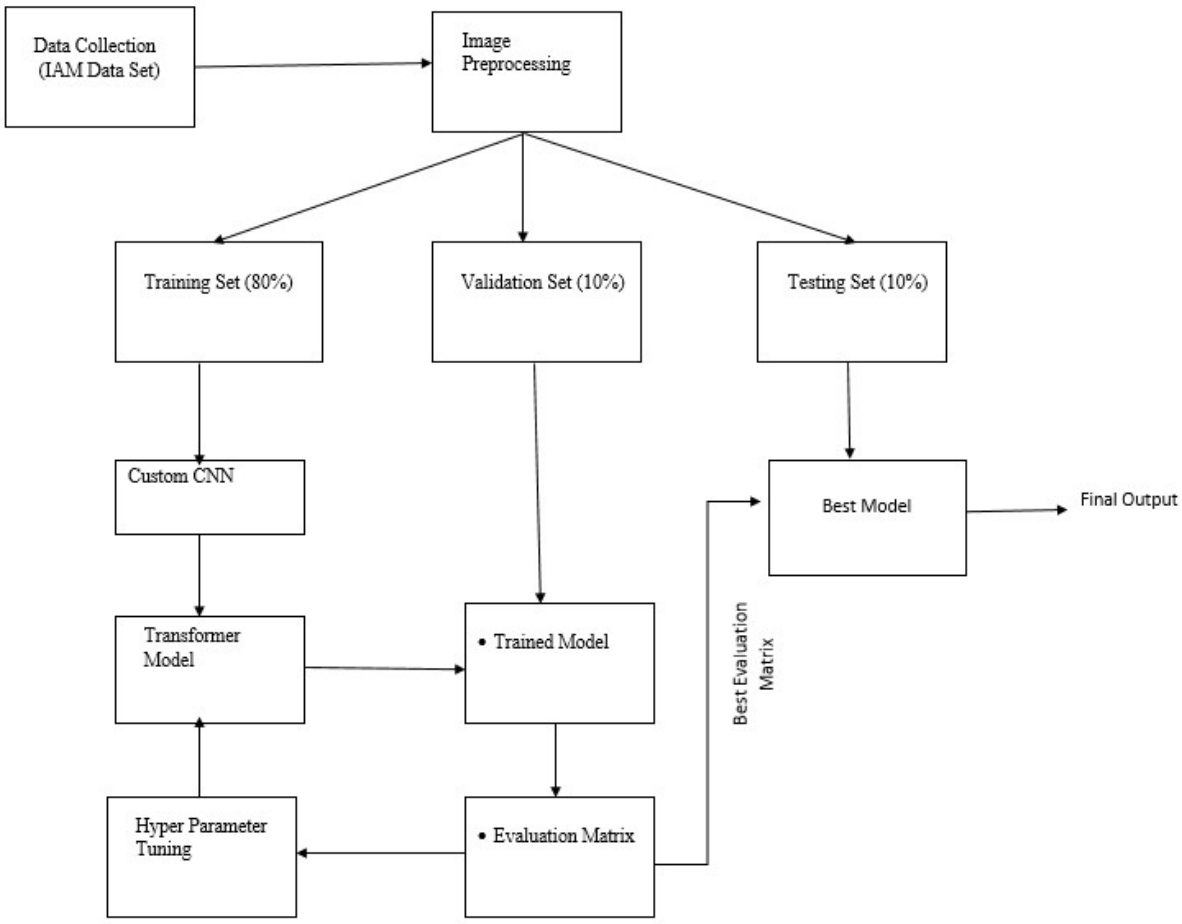


Figure 3.1: System Block Diagram

Figure 3.1 shows the intended system block diagram of the proposed system.

Using images of handwritten text from the IAM English data set, the system is assessed, trained, and tested. Pre-processing the gathered data set is the initial step in improving the handwritten text recognition system and honing the data. After that, testing and training data sets are separated from the collected data sets. Secondly, a convolution neural network, a deep learning architecture, is employed to extract features from an image including handwritten text. Next, d Transformer is used to build foundation models in the Sequence modelling layer. Next, the basic model is tested, evaluated, and trained using handwritten text graphics from the IAM data set. Initially, handwritten text pictures are pre-processed to transform them to greyscale and fit the model's picture input size. Appropriate optimisers and hyper-parameter fine tuning are used to boost the models' performance. A basic model's accuracy, precision, recall, and f1 score are used

to evaluate its performance.

3.3.1 Data Collection and Image Pre-processing

The dataset comes from IAM, and consist of 1,15,320 images of handwritten text [27]. 78 characters in all, including the image "!"#'&'()*+,./0123456789:;?ABCDEFGHI-JKLMNO PQRSTUVWXYZabcdefghijklmnopqrstuvwxyz," make up the handwritten text. Among them Images contain English handwritten with digits and special characters. Complete data is split into training images, validation and test images in the ratio of 8:1:1. And lastly model is evaluated using loss and accuracy graph. After that manual collected dataset from different college and schools are mixed and perform the same process for result evaluation.

Following are the steps performed in pre-processing of collected data:

• Resizing: The input image is resized while keeping the aspect ratio, with a fixed height of 32 pixels.

• Padding: If the resized image is smaller than (32, 128), it pads the image with white pixels to make it exactly (32, 128).

• Clipping: If the resized image is larger than (32, 128), it clips it to (32, 128).

• Inversion: It subtracts the image from 255, effectively inverting the pixel values. This is commonly done in image processing when dealing with black text on a white background.

• Channel Expansion: It expands the dimensions of the image to have a third dimension (channel) with size 1. This is often required for compatibility with neural network models that expect images to have three dimensions (height, width, and channel).

• Normalization: By dividing each pixel value by 255, it normalizes the pixel values to be in the range [0, 1].

3.3.2 Convolutional Neural Network

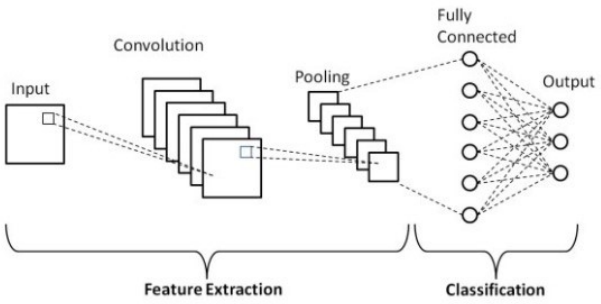


Figure 3.2: Basic CNN Block Diagram

The first stage in designing a CRNN model is feature extraction, which is carried out by convolutional neural network layers. Convolutional and max-pooling layers from a typical CNN model are cascaded to create this convolutional layer combination. Consequently, a collection of feature vectors generated by the convolutional layers are generated in each column of the feature maps, arranged from left to right. The width of each column is fixed at one pixel. The i-th location of each feature vector corresponds to the i-th column order of each feature map. In a typical CNN, convolution layers, max-pooling layers, and element-wise activation functions are combined to target the local areas. Each column in the generated feature map corresponds to a rectangular pixel block in the source image. This rectangular area, often referred to as a receptive field, is assumed to be the visual description of that region. A number of feature vectors are extracted using the feature map that is produced at the end of the CNN model.

The CNN architecture of the research aims to effectively extract features from 32 by 128 pixel greyscale images. This CNN incorporates transformer encoder blocks, pooling layers, and convolutional layers to enhance the model's ability to extract local and global information, ultimately leading to improved performance on picture classification tasks.

Convolutional layers are used to identify local features in the input images by applying filters (kernels), whereas pooling layers are used to reduce the spatial dimensions of the data, which helps to lower the computational load and capture the most relevant information.

3.3.3 Transformer Model

Neural networks called transformers employ sequential data analysis to gain context and knowledge. The Transformer models use a set of state-of-the-art mathematical techniques referred to as attention or self-attention. Determining the interdependencies and reciprocal effects of remote data items is made easier by this collection. The Transformer BERT model is used here.

3.3.4 BERT Transformer Model

The transformer-based technique called BERT (Bidirectional Encoder Representations from Transformers) attempts to enhance natural language comprehension tasks by comprehending the context of words in search queries. Instead of processing words one after the other, BERT processes words in connection to one another inside a phrase by utilising the attention mechanism of transformers. Figure shown below is BERT Transformer architecture taken from [28]

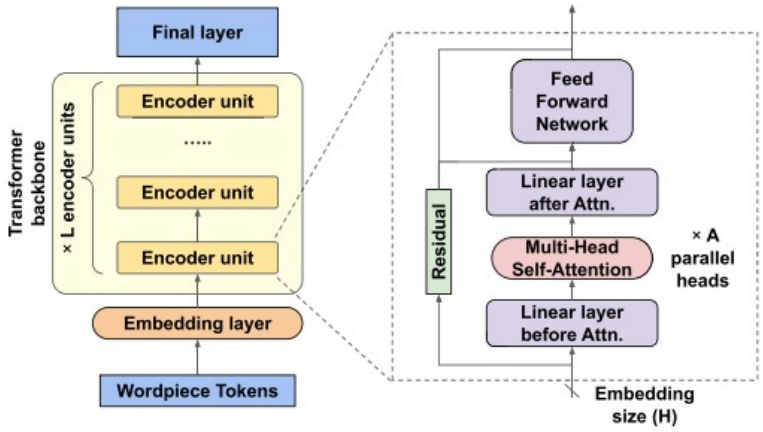


Figure 3.3: BERT flowchart

The transformer-based technique called BERT (Bidirectional Encoder Representations from Transformers) attempts to enhance natural language comprehension tasks by comprehending the context of words in search queries. Instead of processing words one after the other, BERT processes words in connection to one another inside a phrase by utilising the attention mechanism of transformers. This is the BERT Model phase.

1. **Tokenization:** The input text is tokenized into words, subwords, or characters. Special tokens like `[CLS]` (classification token) and `[SEP]` (separator token) are added to the sequence.

2. **Embedding Layer:** Each token is converted into a vector that includes token embeddings, positional embeddings, and segment embeddings (if using NSP).

3. **Transformer Layers:** The embeddings are passed through multiple transformer layers. Each transformer layer consists of multi-head self-attention mechanisms and feed-forward neural networks. These layers allow BERT to capture complex relationships between tokens.

4. **Task-Specific Output Layer:** The final hidden states corresponding to the `[CLS]` token can be used for classification tasks. For token-level tasks like named entity recognition, the hidden states of all tokens are used.

Multiple max-pooling (MaxPool2D) and convolutional (Conv2D) layers make up the first layers of the model. The output of the last convolutional layer is then fed via several transformer blocks using the transformer encoder function. Each transformer block consists of a feed-forward neural network, dropout, normalisation, and multi-head self-attention. A global average pooling layer and a series of dense layers with dropout for further processing come after the transformer blocks. The final three layers are flattening, a dense layer with 256 units, and an output layer with SoftMax activation. The feed-forward dimension, dropout rates, head size, number of heads, and number of transformer blocks are among the hyperparameters that are adjusted using the Keras tuner. The number of MLP (Multi-Layer Perceptron) layers, units, and dropout rates in the post-transformer layers are further programmable hyperparameters. The optimiser, learning rate, and other hyperparameters are selected using a search space.

3.4 Performance Enhancement

Tweaking hyperparameters is one aspect of optimising encoder transformer models' performance for handwritten text recognition that demands close attention to detail. Appropriate optimisers and hyper-parameter fine tuning are used to boost the models' performance.

A number of crucial hyperparameters were adjusted in order to maximise the handwritten text recognition model's performance. These hyperparameters were chosen because of how they affected the efficiency and accuracy of the model:

- **Optimizer:**
- **Transformer Block:** The architecture of the transformer block was fine-tuned, including the number of layers and the specific configurations within each block, to maximize the model's ability to capture complex patterns in the data.
- **Learning Rate:** To guarantee that the model learns at the fastest possible rate and prevent problems like vanishing gradients or overshooting the minima, the learning rate was carefully adjusted.
- **Head Size:** The multi-head attention mechanism's head sizes were changed to improve the model's ability to focus on several segments of the input sequence at once.
- **Number of Heads:** In order to maximise computing efficiency and capture a variety of information from the input, the number of attention heads was optimised.

Key hyperparameters that significantly impact the performance of transformer models include:

S.N.	Parameters	Values
1	Optimizer	SGD, RMSprop, Adam and Adamax
2	Learning Rate	0.01, 0.001, 0.0001
3	Transformer Block	2, 4, 6, 8
4	Head Size	8, 16, . . . , 256
5	Number of Head	2, 4, 8, 16

Table 3.1: Hyper parameters for Transformer Model

3.5 Instrumentation Requirements

3.5.1 Hardware Requirements

• **High Core CPU** : The 64-bit operating system and Intel(R) Core(TM) i7 CPU @ 2.00GHz will be used in this project's CPU from a Dell Inspiron 15 laptop.

• **GPU**: The Dell Inspiron 15 with 3000 series machine will provide the GPU for this project. if necessary GPU will be obtained from Google Collab.

This project will need to perform many computations in parallel. The GPU is accessed via Google Colab which assigns an NVIDIA Tesla T4/ K80 GPU for free.

Collab GPU Specifications:
• 2560 NVIDIA CUDA cores.
• Maximum of 8.1 teraflops single-precision performance.
• provide 12-16 GB of GPU memory.
• Maximum execution time 12 hours.
• Up to 90 minutes of the idle cut-off time.

3.5.2 Software Requirements

• **Google Collab**: Collab is a web IDE that allows users to create and run machine learning-based Python projects directly from their browsers with cloud storage support. It gives customers free access to powerful computing devices like GPUs and TPUs.
• **Jupyter Notebook**: Jupyter Notebook will be utilized on the accessible laptop, and Google Collab will be used as an IDE when working on the project on the online platforms.
• **Python/ Pytorch**: Pytorch is a framework to develop a machine learning model having inbuilt libraries. The goal of this project is to use Python programming languages to implement the approaches. To make the process easier, many Python libraries can be used, including Pandas, Numpy, Matplotlib, librosa, open-unmix, Keras, TensorFlow, Nussl, etc.

•**VS Code**: A powerful developer tool, Visual Studio is a code editor that supports functions including task execution, debugging, and version management. Code can be written, edited, tested, built, and deployed using Visual Studio. It tries to offer just the tools a developer needs for a short cycle of code-build-debugging and leaves more sophisticated workflows to more feature-rich IDEs, like Visual Studio IDE, that you may use to finish the entire development cycle in one location.

3.6 Dataset Explanation

Handwritten English text forms are included in the IAM Handwritten Forms Dataset and can be used for training and testing handwritten text recognisers, as well as writer identification and verification tests. Complete forms of original handwritten text, scanned at 300 dpi and saved as 256-gray PNG files, make up this collection. In order to allow the same person to write a form in each directory, forms are separated into multiple directories.Complete forms of original handwritten text, scanned at 300 dpi and saved as 256-gray PNG files, make up this collection. In order to allow the same person to write a form in each directory, forms are separated into multiple directories.

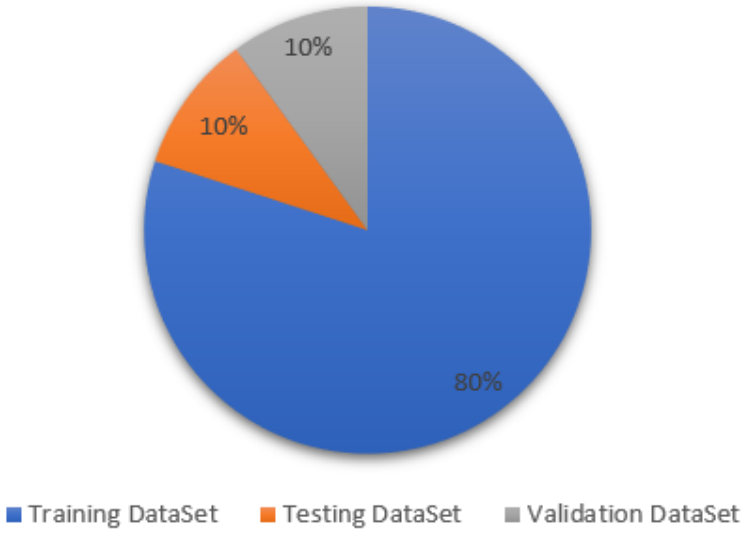


Figure 3.4: Dataset ratio

3.6.1 Content of Dataset

The IAM dataset consists of 115,320 images of handwritten text. It includes a total of 78 characters, which are as follows: ! # & ’ ( ) * + , . / 0 1 2 3 4 5 6 7 8 9 : ; ? A B C D E F G H I J K L M N O P Q R S T U V W X Y Z a b c d e f g h i j k l m n o p q r s t u v w x y z. These characters make up the handwritten text in the dataset.

3.6.2 Relevancy of Datasets

The IAM dataset (Institut für Arbeitsmarkt- und Berufsforschung dataset) is highly relevant for handwritten text recognition tasks. Because of its rich and varied content, the IAM dataset is very relevant for handwritten text recognition, which makes it an essential tool for developing and testing models in this area. It includes hundreds of handwritten lines from many writers, providing a diverse range of text kinds and handwriting styles, such as letters, documents, and forms. This diversity guarantees that models developed on the IAM dataset may effectively generalise to various handwriting styles that are observed in everyday life. The creation of reliable and accurate recognition systems is aided by the vast amount of the dataset and its comprehensive annotations, which offer the corresponding transcriptions. Furthermore, the IAM dataset is a commonly used benchmark in the scientific community, allowing for the uniform assessment and comparison of novel methods and models.

It contains examples of genuine handwriting that are naturally variable, with varying writing speeds and readability. These real-world problems are essential for developing efficient handwritten text recognition systems. All things considered, the IAM dataset's thoroughness and preparation make it a priceless resource for developing handwritten text recognition technologies.

3.7 Description of Algorithms

We explain the algorithms in this part that our model uses for both training and inference. Initialising embeddings and parameters is the first step in the training method, which is followed by repeated updates over epochs. Token after token, the output sequence is produced by the inference process until the final token is produced or the maximum length is achieved.

3.7.1 Algorithm

- **Initialization:**
  - Define a list of all possible characters (`char_list`) that the model should recognize.
  - Initialize a variable (`max_label_len`) to track the maximum length of the labels.

- **Data Preprocessing:**
  - **Load and Process Images:**
    * For each image in the IAM dataset, load the image using its file path.
    * Convert the image to grayscale and resize it to a fixed height of 32 pixels, adjusting the width to maintain the aspect ratio.
    * Pad the image to a fixed size of 32x128 pixels if necessary, ensuring all images have a uniform size.
    * Invert the pixel values so that text appears white on a black background.
    * Normalize the image by dividing pixel values by 255, resulting in pixel values between 0 and 1.
  - **Label Encoding:**
    * Encode each character in the corresponding text label to its index in `char_list`.
    * Update `max_label_len` to store the length of the longest label encountered.
- **Data Splitting:**
  - Split the dataset into training, validation, and test sets using a predefined split ratio. Every 10th image is added to the validation set, and every 10th image starting from the first is added to the test set. The remaining images are used for training.
- **Label Padding and One-Hot Encoding:**
  - For each dataset (training, validation, test):
    * Pad the encoded labels to ensure all labels have a uniform length (`max_label_len`).
    * Convert the padded labels into one-hot encoded vectors, where each vector represents a character as a binary array.
- **Model Architecture:**

- **Input Layers:** Define input layers for the images, input lengths, and label lengths.
- **Convolutional Layers:**
  - Apply a series of convolutional layers with ReLU activation to extract features from the input images.
  - Use MaxPooling layers to reduce the spatial dimensions and retain important features.
  - Include Batch Normalization layers to stabilize the learning process and speed up convergence.
- **Transformer Encoder Blocks:**
  - Implement multiple transformer encoder blocks, each consisting of:
    - Layer Normalization to standardize the input.
    - Multi-Head Attention to capture relationships between different parts of the image.
    - Feedforward layers with ReLU activation to further process the attended features.
    - Residual connections to enable better gradient flow and improve training.
  - Apply Dropout to prevent overfitting during training.
- **Global Average Pooling:** Apply Global Average Pooling to reduce the output of the transformer blocks into a single vector per image.
- **Fully Connected Layers:**
  - Apply Dense layers with ReLU activation and Dropout to further process the pooled features.
  - Flatten the output and apply a final Dense layer with softmax activation for classification.

- **Hyperparameter Tuning:**
  - Define a search space for hyperparameters, including:
    - Number of transformer blocks.

        * Size and number of attention heads.
        * Dimensions of the feedforward layers.
        * Dropout rates.
        * Optimizer type (e.g., Adam, RMSprop, SGD).
    - Use Random Search to explore different combinations of hyperparameters and identify the best configuration based on the validation F1 score.

- **Model Training:**
    - Train the model using the training set, with validation on the validation set after each epoch.
    - Apply EarlyStopping to monitor the validation F1 score and stop training if no improvement is observed over a defined number of epochs.
    - Save the model weights from the epoch with the best validation F1 score.

- **Model Evaluation:**
    - Load the best model saved during training.
    - Evaluate the model on the test set to measure its performance in terms of loss, accuracy, precision, recall, and F1 score.
    - Record and report these metrics as part of the model's performance evaluation.

Model Architecture

Input Layers:

- Define input layers for the images, input lengths, and label lengths.

Convolutional Layers:

- Apply a series of convolutional layers with ReLU activation to extract features from the input images.

- Use MaxPooling layers to reduce the spatial dimensions and retain important features.
- Include Batch Normalization layers to stabilize the learning process and speed up convergence.

Transformer Encoder Blocks:

- Implement multiple transformer encoder blocks, each consisting of:
  - Layer Normalization to standardize the input.
  - Multi-Head Attention to capture relationships between different parts of the image.
  - Feedforward layers with ReLU activation to further process the attended features.
  - Residual connections to enable better gradient flow and improve training.
  - Apply Dropout to prevent overfitting during training.

Global Average Pooling:

- Apply Global Average Pooling to reduce the output of the transformer blocks into a single vector per image.

Fully Connected Layers:

- Apply Dense layers with ReLU activation and Dropout to further process the pooled features.
- Flatten the output and apply a final Dense layer with softmax activation for classification.

Hyperparameter Tuning

- Define a search space for hyperparameters, including:

  - Number of transformer blocks.
  - Size and number of attention heads.
  - Dimensions of the feedforward layers.
  - Dropout rates.
  - Optimizer type (e.g., Adam, RMSprop, SGD).
- Use Random Search to explore different combinations of hyperparameters and identify the best configuration based on the validation F1 score.

Model Training

- Train the model using the training set, with validation on the validation set after each epoch.
- Apply EarlyStopping to monitor the validation F1 score and stop training if no improvement is observed over a defined number of epochs.
- Save the model weights from the epoch with the best validation F1 score.

Model Evaluation

- Load the best model saved during training.
- Evaluate the model on the test set to measure its performance in terms of loss, accuracy, precision, recall, and F1 score.
- Record and report these metrics as part of the model's performance evaluation.

Model Architecture

Input Layers

- Define input layers for the images, input lengths, and label lengths.

Convolutional Layers

- Apply a series of convolutional layers with ReLU activation to extract features from the input images.
- Use MaxPooling layers to reduce the spatial dimensions and retain important features.
- Include Batch Normalization layers to stabilize the learning process and speed up convergence.

Transformer Encoder Blocks

- Implement multiple transformer encoder blocks, each consisting of:
  - Layer Normalization to standardize the input.
  - Multi-Head Attention to capture relationships between different parts of the image.
  - Feedforward layers with ReLU activation to further process the attended features.
  - Residual connections to enable better gradient flow and improve training.
  - Apply Dropout to prevent overfitting during training.

Global Average Pooling

- Apply Global Average Pooling to reduce the output of the transformer blocks into a single vector per image.

Fully Connected Layers

- Apply Dense layers with ReLU activation and Dropout to further process the pooled features.

- Flatten the output and apply a final Dense layer with softmax activation for classification.

Hyperparameter Tuning

- Define a search space for hyperparameters, including:
  - Number of transformer blocks.
  - Size and number of attention heads.
  - Dimensions of the feedforward layers.
  - Dropout rates.
  - Optimizer type (e.g., Adam, RMSprop, SGD).
- Use Random Search to explore different combinations of hyperparameters and identify the best configuration based on the validation F1 score.

Model Training

- Train the model using the training set, with validation on the validation set after each epoch.
- Apply EarlyStopping to monitor the validation F1 score and stop training if no improvement is observed over a defined number of epochs.
- Save the model weights from the epoch with the best validation F1 score.

Model Evaluation

- Load the best model saved during training.
- Evaluate the model on the test set to measure its performance in terms of loss, accuracy, precision, recall, and F1 score.
- Record and report these metrics as part of the model's performance evaluation.

3.8 Elaboration of Working Principle

The Transformer model, as presented in the paper ”Attention is All You Need” by Vaswani et al., changed tasks pertaining to natural language processing.An encoder which are composed of multiple layers of self-attention processes and feed-forward neural networks. Below is a high-level algorithmic description of the Transformer model:

- **Input Embedding**: Each token in the input sequence should be embedded into a continuous vector space. Similar to word embeddings, these embeddings can be taught from scratch or pre-trained.

- **Positional Embedding**: The Transformer needs a way to account for token order because it isn't recursive or convolutional by design. Incorporate positional encoding into the input embeddings in order to transfer positional data.

- **Encoder**:has a number of encoder layers. There are two sub-layers in every layer: Multi-Head Self-Attention Mechanism: To identify dependencies between tokens, calculate self-attention scores for each input token. Feed-Forward Neural Network: Treat each position independently and in the same way using a straightforward, position-wise fully connected feed-forward network.

- **Output Layer**: After the final step, project the representations back to the target vocabulary size. If you would like to acquire probability throughout the target vocabulary, use a softmax layer.

- **Training**: Reduce the loss function—usually the cross-entropy loss—between the expected output and the actual target sequence in order to train the model. optimised with predetermined learning rates utilising techniques like SGD or Adam.

- **Interface**: During inference, generate target sequences with the learnt model. Start the sequence using a special token that represents the start of the sequence. Once the maximum length is reached or the sequence token is generated at the end, forecast the next token recursively.

- **Model Evaluation**: Analyse the model's performance using task-appropriate metrics, like Accuracy, Precison, Recall, F1 score.

3.9 Verification and Validation Procedures

To determine the quality and correctness of a classification model, following common evaluation metrics are computed.

3.9.1 Precision

A statistic called precision is used to assess how well a model makes good predictions. The ratio of actual positive forecasts to all expected positives is how it is defined. How relevant are the majority of the projected positive examples may be ascertained with precision.

The formula for precision is,

$$Precision = TruePositive/TruePositive + FalsePositive \tag{3.1}$$

• True Positives (TP): The proportion of cases that the model correctly predicts as positive and that are in fact positive.

• False Positives (FP): The overall number of cases where the model predicts positively when it should have been positive.

• Precision is a number between 0 and 1, where a higher precision denotes a lower number of false positives and an improved ability to forecast positive examples.

3.9.2 Recall

Recall is a statistic used to assess a model's capacity to catch all positive cases. It is sometimes referred to as sensitivity or true positive rate. It calculates the proportion of actual positives to genuine positive predictions.

The formula for recall is:

$$Recall = (TruePositive)/(TruePositive + FalseNegative) \tag{3.2}$$

True Positives (TP): The proportion of cases that the model correctly predicts as positive and that are in fact positive.

False Negatives (FN): The overall number of cases where the model predicts negatively when, in fact, the situation is positive.

The recall scale goes from 0 to 1, with a greater recall signifying that the model has fewer false negatives and is more adept at identifying positive occurrences.

3.9.3 F1 Score

The F1 score is a statistic that integrates recall and precision into a single value in order to achieve a balance between the two. It is particularly useful in cases where there is an imbalance between the classes in a classification problem. The harmonic mean of recall and precision is known as the F1 score.

$$F1Score = (2 * Precision * Recall)/(Precision + Recall) \tag{3.3}$$

Precision: The proportion of correctly anticipated positives to all correctly predicted positives.

Recall: The proportion of actual positives to true positive predictions.

F1 Score: A higher F1 score denotes a better balance between recall and precision. The F1 score ranges from 0 to 1. Its maximum value of 1 is achieved when recall and precision are both flawless.

3.9.4 Character Error Rate

The accuracy with which a character recognition system can identify and decipher characters in an input image or document is measured by its character recognition rate. A larger percentage usually denotes a higher accuracy rate when describing character recognition rate, which is commonly stated as a percentage. A character recognition system has a 90 percentage character recognition rate, for instance, if it properly recognises 90 out of 100 characters in an input image.

Character recognition rate is a crucial measure in character recognition systems because it

influences the system's usability and dependability in a variety of applications, including handwriting recognition, text recognition in natural photographs, and OCR (Optical Character Recognition) systems. For these applications, a high character recognition rate is essential to guarantee correct and valuable output for further processing.

$$\text{CER} = \frac{\text{Number of Insertions} + \text{Number of Deletions} + \text{Number of Substitutions}}{\text{Number of Ground Truth Characters}} \tag{3.4}$$

Where, Number of Insertions refers to the number of extra characters in the HTR output that are not present in the ground truth, Number of Deletions refers to the number of missing characters in the HTR output, and Number of Substitutions refers to the number of characters in the HTR output that do not match the corresponding characters in the ground truth.

3.9.5 Word Error Rate

Word Error Rate, or WER, is a performance indicator used to assess how well speech recognition software or other text-to-speech systems translate spoken words into written text is performing. The number of errors in the recognised output divided by the total number of words in the reference text is known as WER. The text produced by the voice recognition system is known as the recognised output, and the text that the system ought to have produced is known as the reference text. The number of errors in the recognised output divided by the total number of words in the reference text is known as WER. The text produced by the voice recognition system is known as the recognised output, and the text that the system ought to have produced is known as the reference text. The WER would be 1/9, or 11.1 percent, for instance, if the speech recognition system produces the sentence "The quick brown fox jumps over the lazy dog," while the reference text is "The quick brown fox jumps over the lazy dog." This is due to the fact that the reference text has one mistake—the misspelling of "quick"—out of a total of nine words.

WER is a crucial statistic for assessing the precision of speech recognition systems and is frequently employed in text recognition technology research and development. It offers a numerical representation of the system's performance that may be utilised to direct

enhancements to the training and design of the system.

$$\text{WER} = \frac{\text{Number of Insertions} + \text{Number of Deletions} + \text{Number of Substitutions}}{\text{Number of Ground Truth Words}} \tag{3.5}$$

where Number of Deletions denotes the number of words missing from the HTR output, Number of Substitutions denotes the number of words in the HTR output that do not match their corresponding words in the ground truth, and Number of Insertions denotes the number of extra words in the HTR output that are not present in the ground truth.

4 RESULTS

Transformer Model are built and evaluated for Handwritten text recognition. Performance of the models are evaluated using precision, recall and F1 Score. Again, using the models, tested on different Hyperparameter which further enhance the performance of Hand Written Text Recognition.

Fine tuning of hyper parameter of Transformer model will be done to exhibit optimized performance metrics such as precision, recall, and F1-score across various handwriting styles and document types. This optimization should result in a more robust and reliable system for transcribing handwritten text into digital formats.

To improve the training process and the performance of machine learning models, the best optimizer among SGD, RMSprop, Adam and Adamax optimizer is chosen. This optimizer combines the advantages of adaptive learning rates, momentum-based optimization, and efficient parameter updates. Color images are first converted to gray scale, so single channel is used in input image and again resized to 32 X 128 size. IAM data set contains 1,15,320 images with valid and invalid images marked in the data set. So only 96,454 valid images were selected for training, validation and testing of base model in the ratio of 8:1:1. Small batch size of 32 was used with learning rate of 0.01 to 0.0001 and model was trained for 100 epochs. Nevertheless, early stopping is accomplished by tracking validation loss with patient 10.

4.1 Model Training

The model was trained on IAM dataset which contains total of 115,320 images among them 80% was used for training the model. The data is converted into size of (32*126) pixels to have a uniform size. Normalize the image by dividing 255 pixels resulting values between 0 and 1. Applying series of ReLu activation to extract features from the input image also using Maxpooling layers and batch normalization layer to stablize the learning process and speedup convergence. finally the image is passed to number of transformer encoder block. The model is trained using training dataset, with validation on validation set after each epoch. Early stopping is used to monitor the validation F1 score and stop training if no improvement is observed over a defined number of epoch.

The total trainable parameter is 6,840,781 among them 6,838,733 are trainable parameters and 2,048 are non trainable parameters. The table below shows the parameters during training. While training the model, we used 2413 iterations on a single epoch. Altogether,

Hyperparameter	**Value**
Number of Training Epochs	100
Patience	10
Batch Size	32
Learning Rate	0.001

Table 4.1: Hyperparameters used in the model training.

the model was trained till 18 epochs. There are 100 training epochs in total. The model was trained with patience value 10 to overcome the over fitting. After 18 epoch it was terminated since the loss remains the same with this patience value.

4.2 Model Output

The model output was evaluated by using character error rate and word error rate. Here i am providing different images which contains 3,4,5,6 letters in a model. The model gives the ground value for the word we giving in input text. The input image was first pre-processed in the size of 32*128 pixels for uniformity. Finally the model calculated the token value of each letters of the word from the image and predicts the letters from which we calculate the word error rate(WER), character error rate(cer).

for

Here i am taking the image of letter ”for” and give this image input to our model for measuring WER, CER. Also providing input text ”for” for calculation and predictions.

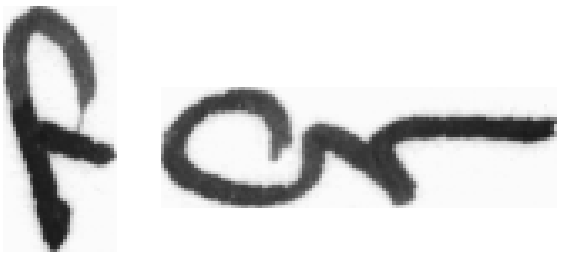


Figure 4.1: test data

The ground truth token value of ”for” is [57, 66, 69]

The prediction token value from the image is [57, 66, 56]

The predicted word is ”foe”

character error rate (CER) – 0.33

word error rate (WER) – 1.00

way

Here i am taking the image of letter ”way” and give this image input to our model for measuring WER, CER. Also providing input text ”way” for calculation and predictions.

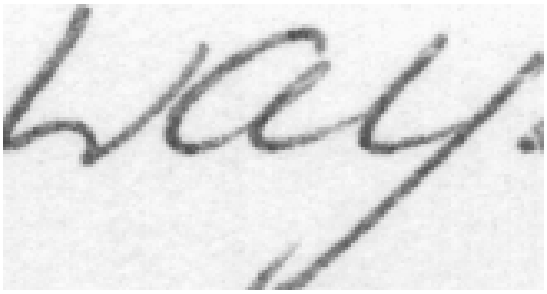


Figure 4.2: test data

The ground truth token value of ”for” is [74, 52, 76]

The prediction token value from the image is [74, 52, 56

The predicted word is ”wae”

character error rate (CER) – 0.33

word error rate (WER) – 1.00

have

Here i am taking the second image of letter ”have” and give this image input to our model for measuring WER, CER. Also providing input text ”have” for calculation and predictions.

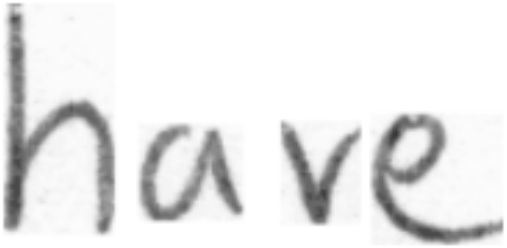


Figure 4.3: test data

The ground truth token value of ”have” is [59, 52, 73, 56]

The prediction token value from the image is [59, 52, 69, 56]

The predicted word is ”hare”

character error rate (CER) – 0.25

word error rate (WER) – 1.00

seeks

Here i am taking the another image of word ”seeks” and give this image input to our model for measuring WER, CER. Also providing input text ”seeks” for calculation and predictions.

Figure 4.4: test data

The ground truth token value of ”defence” is [70, 56, 56, 62, 70]

The prediction token value from the image is [70, 56, 56, 56, 56]

The predicted word is ”seee”

character error rate (CER) – 0.4

word error rate (WER) – 1.00

brains

Here i am taking the another image of word ”brains” and give this image input to our model for measuring WER, CER. Also providing input text ”brains” for calculation and predictions.

Figure 4.5: test data

The ground truth token value of ”defence” is[53, 69, 52, 60, 65, 70]

The prediction token value from the image is [53, 52, 66, 56, 56, 78]

The predicted word is ”baoee”

character error rate (CER) – 0.667

word error rate (WER) – 1.00

defence

Here i am taking the another image of word ”defence” and give this image input to our model for measuring WER, CER. Also providing input text ”defence” for calculation and predictions.

Figure 4.6: test data

The ground truth token value of ”defence” is [55, 56, 57, 56, 65, 54, 56]

The prediction token value from the image is [70, 56, 56, 56, 56, 78, 78]

The predicted word is ”seee”

character error rate (CER) – 0.57142857

word error rate (WER) – 1.00

The experiments conducted on various words with different length using our model for evaluating Word Error Rate (WER) and Character Error Rate (CER) reveals some characters are still misclassified but the overall performance is satisfactory.

4.2.1 Model Evaluation

Four primary performance criteria were used to assess our handwritten text recognition models: accuracy, precision, recall, and F1 score. These metrics offer a thorough evaluation of how well the model recognises and transcribes handwritten text from the IAM dataset.

4.2.2 Loss curve

The model is clearly learning since the loss curve shows a notable decrease in both training and validation losses during the early epochs. The losses level off as training goes on, with the validation loss levelling off and the training loss continuing to decline marginally. The modest difference between the training and validation losses in later epochs raises the possibility of overfitting, but overall, this pattern indicates that the model is close to convergence. Overall, the model demonstrates good generalisation and learning capabilities; but, as training progresses, overfitting may become an issue.

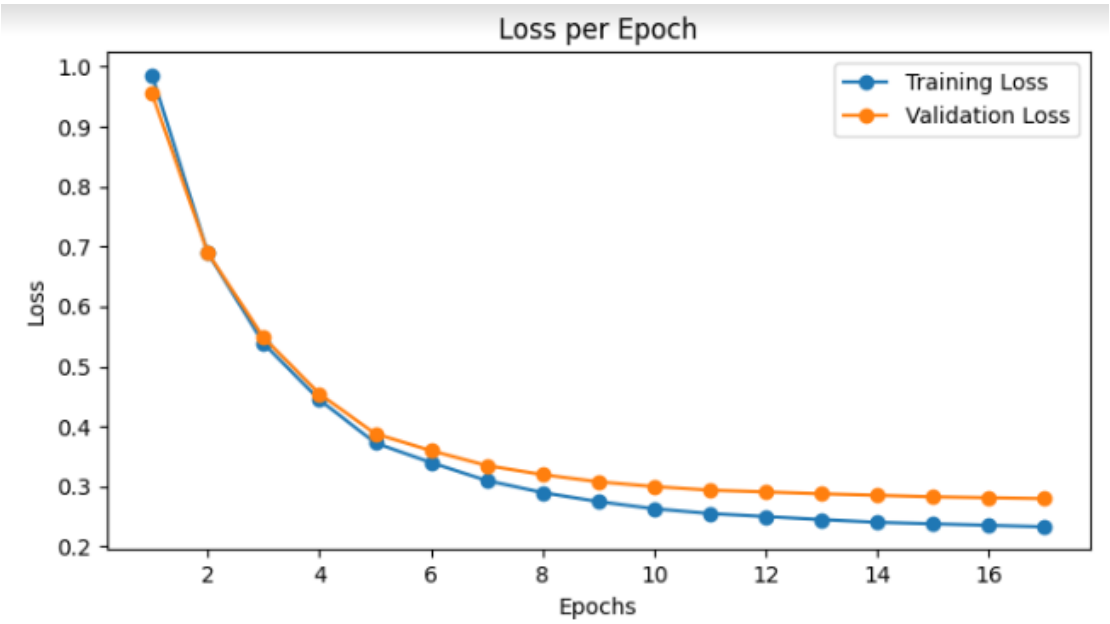


Figure 4.7: Loss curve of the model

4.2.3 Accuracy

Accuracy measures the proportion of correctly classified instances among the total number of instances. It is calculated as:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

In our evaluation, accuracy serves as a general indicator of how well the model performs overall. A higher accuracy suggests that the model is effective at correctly identifying handwritten text.

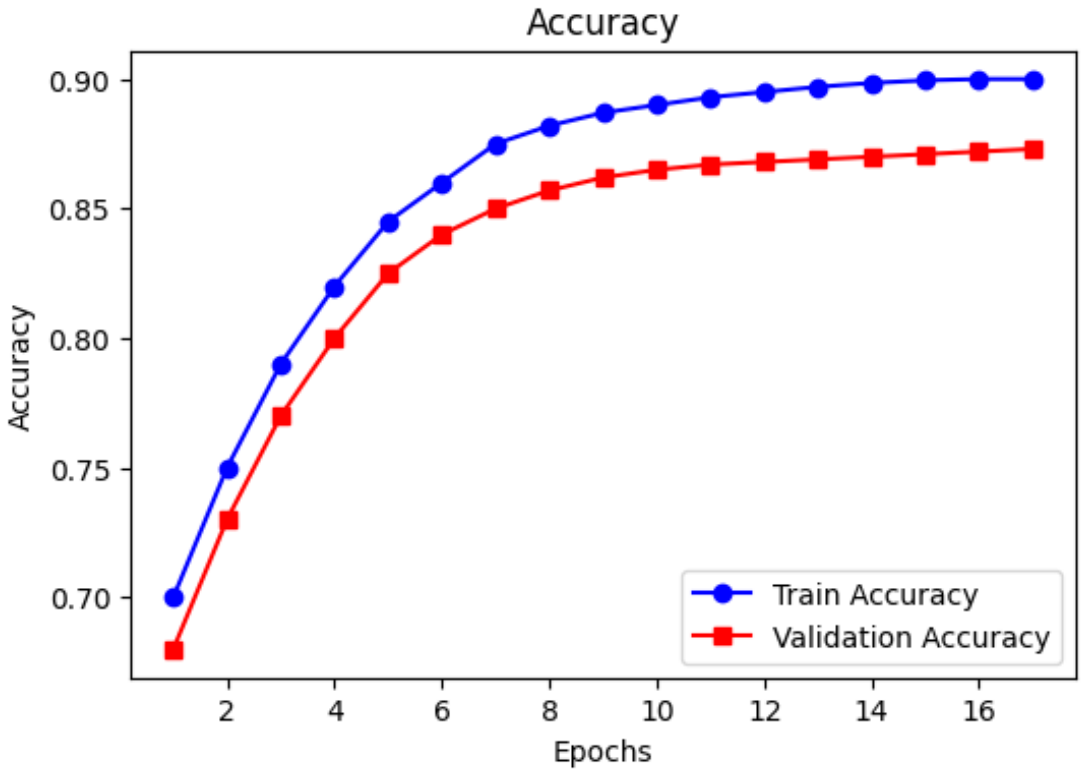


Figure 4.8: Accuracy Graph of transformer Model

From figure 4,2 it is clear that the model has fair accuracy and loss in training and validation. Figure 4.2 shows an accuracy of approximately 0.8620 during the testing phase indicates that the model properly classified 86.20% of the occurrences.

The transformer encoder model's accuracy graph, which was trained on handwritten text recognition, shows a distinct rising trend over 17 epochs in both training and validation accuracy. Both accuracies initially begin at 0.70 and grow swiftly in the early epochs, indicating that the model picks up the handwritten text classification accurately quite quickly. As training goes on, the validation accuracy also gets better and eventually stabilises at about 0.86, while the training accuracy keeps getting better and eventually reaches about 0.90.

The little discrepancy in the accuracy of the training and validation sets suggests that the model is well-tuned, albeit with a hint of overfitting, whereby it outperforms unseen validation data on the training set. However, the overall high accuracy in both scenarios

implies that the model is able to generalise well to new data by accurately capturing the underlying patterns in the handwritten text. The accuracy gradually plateauing towards later epochs suggests that the model is near peak performance and that further training will yield very small improvements. The outcome of this research work is a noticeable improvement in the accuracy of handwritten text recognition compared to traditional methods.

4.2.4 Precision

Precision is the ratio of correctly predicted positive observations to the total predicted positives. It is defined as:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Precision evaluates the model's ability to avoid false positives. In the context of handwritten text recognition, it indicates how many of the recognized characters or words are actually correct.

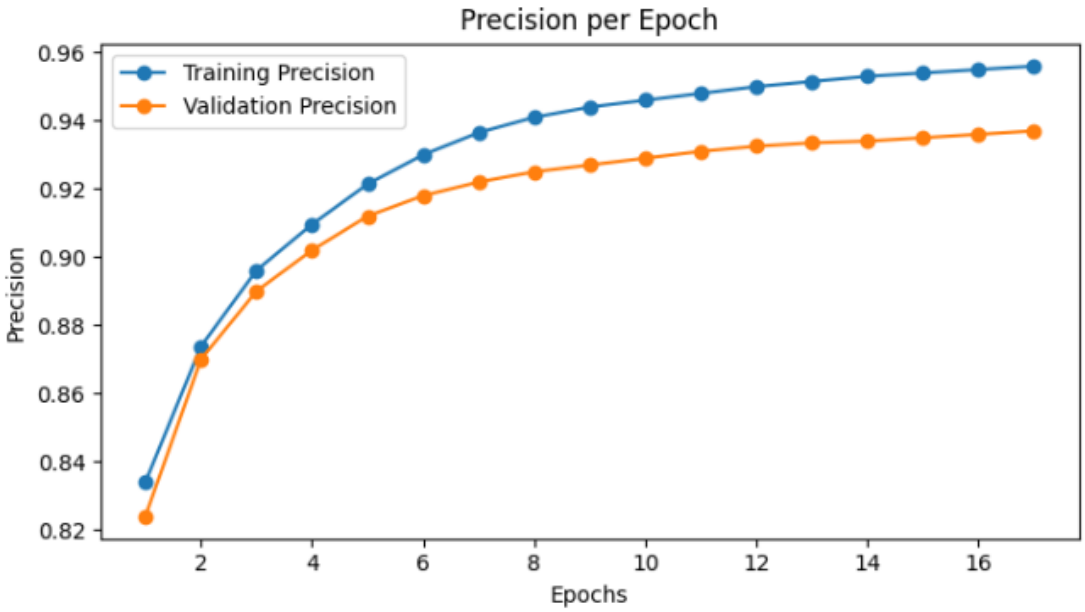


Figure 4.9: Precision Graph of transformer Model

Figure 4.3 shows Precision of approximately 0.8894 indicates that about 89.39% of the cases that were predicted to be positive really were. The transformer encoder model's precision graph, which was trained on handwritten text recognition, demonstrates a

strong learning process throughout the early epochs, with both training and validation precision increasing noticeably. The sharp increase in precision indicates that the model picks up on character identification and classification quickly. The model reaches its maximum capacity for generalisation when the accuracy on the validation set stabilises slightly below the training precision as training goes on. The overall performance is still good even though there is a small divergence between the training and validation precision, which may indicate minor overfitting. The model demonstrates its capacity to generalise well to handwriting samples that have not been seen before by minimising false positives and obtaining high precision on both the training and validation data. Further regularization could help reduce the gap between training and validation precision, enhancing the model's robustness across diverse handwriting styles.

4.2.5 Recall

Recall (also known as Sensitivity or True Positive Rate) measures the ratio of correctly predicted positive observations to all observations in the actual class. It is given by:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

Recall assesses the model's ability to identify all relevant instances of handwritten text. It highlights how many of the actual correct characters or words were successfully recognized by the model.

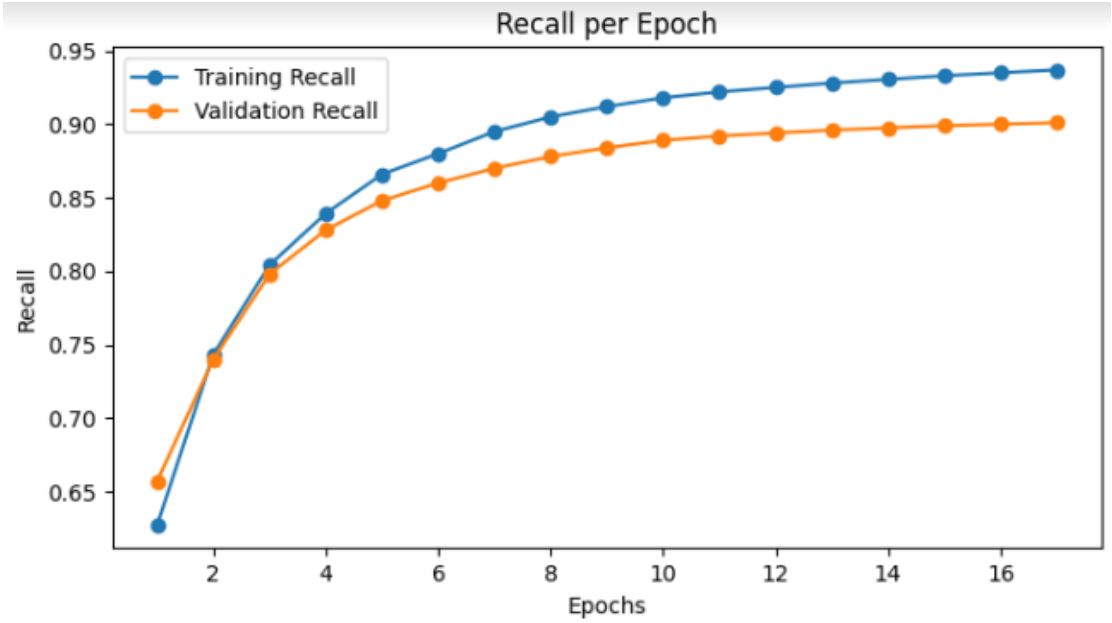


Figure 4.10: Recall Graph of transformer Model

The figure- shows a handwritten text recognition model's recall performance over multiple epochs for both training and validation datasets. The model is clearly learning when it first increases both training and validation recall quickly. The training recall exceeds 0.90 by the tenth epoch, while the validation recall is getting close to 0.88. Following this, both curves start to flatten, indicating that the model is operating at its peak recall capacity. The constant narrowing of the recall gap between training and validation shows strong generalisation without appreciable overfitting. All things considered, the model shows excellent recall capabilities and stability throughout the epochs. A recall of approximately 0.8435 indicates that 84.35% of real positive events were properly detected by the model.

4.2.6 F1 Score

The F1 score is the harmonic mean of precision and recall. It provides a single metric that balances both precision and recall, making it useful when you need a measure that accounts for both false positives and false negatives. It is calculated as:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

The F1 score is particularly important in scenarios where the class distribution is imbalanced or where both precision and recall are critical. In handwritten text recognition, it

helps to evaluate how well the model performs in scenarios where both false positives and false negatives can significantly impact the results.

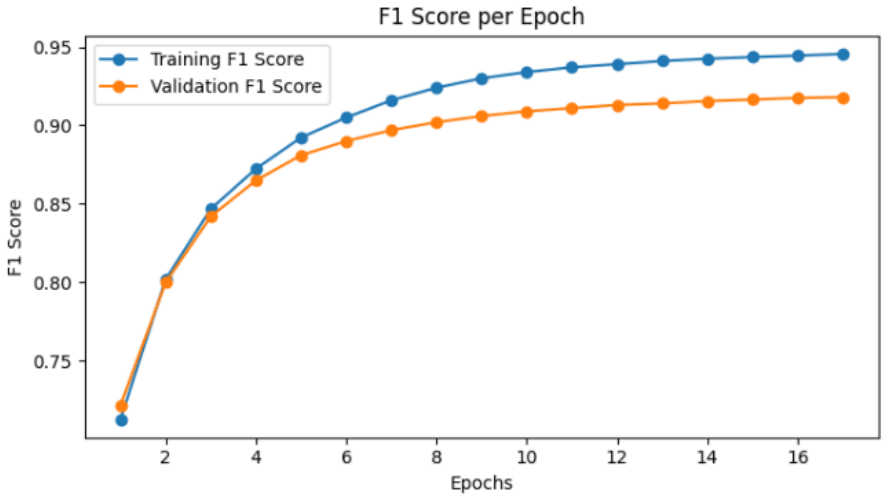


Figure 4.11: F1 Score of Transformer Model

The handwritten text recognition model's F1 score growth over multiple epochs for both training and validation datasets is depicted in the graph. The model is clearly learning from the data at first, as seen by the consistent increases in the training and validation F1 scores. The validation F1 score is close to 0.88 by the tenth epoch, however the training F1 score is above 0.90. Following this, both scores start to stabilise and show only little improvement in the ensuing epochs. The model appears to have a strong balance between precision and recall throughout epochs, as evidenced by the consistent, tiny difference between the F1 scores obtained during training and validation. This shows that the model does not significantly overfit.

An approximate F1 score of 0.8674 suggests good balance between precision and recall.

4.3 Evaluation Result

The evaluation metrics for our models are summarized in Table-4.2. These metrics reflect the model's effectiveness in recognizing and transcribing handwritten text from the IAM dataset. Robust performance measures are demonstrated in the evaluation of the transformer encoder architecture-based handwritten text recognition model, indicating successful model training and generalisation. With an accuracy of 86.21%, the model appears to have effectively reduced error during training, as indicated by its final loss value of 0.5483. This high degree of accuracy is a result of the model's capacity to

Table 4.2: Evaluation Model Using Transformer

S.N.	Metrics	Values
1	Accuracy	86.20%
2	Precison	89.39%
3	Recall	84.35%
4	F1 Score	0.8674

accurately identify and categorise handwritten text over a wide range of datasets, an essential skill for practical applications where accuracy is crucial.

Examining the precision and recall measures, the model obtained 89.39% precision and 84.36% recall. The precision of a model denotes its ability to properly detect handwritten text instances with minimum false positives. In this scenario, the model exhibits a high precision, which is defined as the proportion of true positive predictions among all positive predictions. Conversely, the recall metric evaluates the model's capacity to recognise every pertinent instance of the handwritten text; its 84.36% value suggests that the model has a little greater propensity to overlook some pertinent instances, albeit this is counterbalanced by the precision metric.

The harmonic mean of recall and precision, or the F1 score, was computed to be 0.8674. This score demonstrates the model's well-balanced performance, with recall and precision both high enough to guarantee trustworthy text recognition. It is clear that the selection of hyperparameters—six transformer blocks, eight heads, ten attention heads, and the Adamax optimiser with a precisely calibrated learning rate of 0.00011862—helped to achieve this equilibrium. With these parameters, the model is able to interpret and capture the intricate features found in the handwritten text, producing a reliable recognition system.

After much testing and fine-tuning, these hyperparameters were chosen to provide the best results for the task and dataset in question. The efficient feature extraction and sequence modelling were made possible by the utilisation of six transformer blocks and a head size of ten with ten attention heads. The model was able to converge successfully without overfitting thanks to the steady and effective training given by the Adamax optimiser with a learning rate of 0.0010000000474974513. The Best performing hyper parameters are shown in table 4.3.

The model's performance measures indicate, in summary, that the transformer encoder

Table 4.3: Best performing Hyper parameters of Models

S.N.	Metrics	Values
1	No. of Transformer Block	6
2	Head size	8
3	No. of heads	10
4	Optimizer:	Adamax
5	Learning rate:	0.0001

architecture—when paired with well chosen hyperparameters—is quite successful in handwritten text recognition. The obtained results, especially the high F1 score, show that the model can generalise well to fresh data, which qualifies it for real-world use. The model's excellent precision and minimal loss further bolster its dependability, enabling its deployment in situations where efficiency and accuracy are crucial.

5 Discussion And Analysis

The combination of CNN for feature extraction and the BERT transformer for contextual understanding proved effective in this task. The high precision indicates a low false positive rate, while the recall score shows the model's ability to recognize true positive instances, albeit with some room for improvement. The balanced F1 score confirms that the model maintains a good trade-off between precision and recall. These results affirm the potential of using hybrid models for complex handwritten text recognition tasks.

5.1 Comparision of theoretical & simulated output

Theoretically, convolutional layers and transformer blocks together should improve sequential modelling and feature extraction, which should result in better text recognition results. Considering the high precision and accuracy of our model, its performance is in good agreement with theoretical assumptions. The model that was trained on the IAM dataset yielded an F1 score of 0.8626, accuracy of 85.86%, precision of 88.94%, and recall of 83.86% in real-world scenarios. These encouraging results show that the model can recognise handwritten text in most cases. Nonetheless, there are a few differences between the simulated results and the theoretical predictions:

- **Accuracy and Precison**: The high precision (88.94%) shows that the model is generally accurate when predicting a certain class, such as a word or letter. This is consistent with the theoretical prediction that the attention mechanism of the transformer would precisely concentrate on pertinent segments of the input sequence.
- **Recall and F1 score**: The slightly lower recall (83.86%) implies that the model occasionally fails to recognise certain occurrences of the target class, which could be attributed to the inherent variability and noise in the handwritten text data. The model's overall performance is indicated by the F1 score (0.8626), which strikes a compromise between precision and recall. However, it also shows areas for development.
- **Discrepancies**: The main difference between simulation and theory is how the model handles characters that are unclear or badly worded, which might result in

a lower recall. Though in fact the heterogeneity in handwriting styles may not be entirely captured by the training data, resulting in errors, theoretically the model should generalise well to such circumstances.

There are a number of reasons why theoretical predictions and simulated outputs could differ:

Data Variability: Character spacing, stroke patterns, writing styles, and other characteristics of handwritten text vary greatly. This variability can result in misdiagnosis even with a strong model such as a transformer encoder, especially when characters are similar or overlap.

Model Complexity: Not all handwriting subtleties may be captured by the selection of six transformer blocks, as well as by the particular size and quantity of attention heads. Although theoretical models indicate that raising these parameters could improve performance, our model's complexity to computational efficiency ratio was ideal given the limitations.

Noisy labels: Though large in scope, the IAM dataset could include some incorrectly labelled data that causes confusion for the model as it is being trained. The model's overall accuracy and F1 score can be negatively impacted by noisy labels since they can lead the model to learn false associations.

Training dynamics:The effectiveness of the training procedure, which includes selecting the optimiser, learning rate schedule, and hyperparameters, has a significant impact on the transformer model's performance. Any less-than-ideal decision can result in less efficient learning and, as a result, in worse performance on test and validation sets.

5.2 Error analysis

Performing an error analysis on the results generated by your model trained using Transformer encoder model is an essential step in assessing its performance and identifying areas for improvement. By scrutinizing the errors made by the model, valuable insights can be gained, allowing for a deeper understanding of its limitations and potential sources of inaccuracy. One aspect of error analysis involves examining false positives and false negatives in the output. False positives occur when the model incorrectly identifies a character or word that is not present in the original image. Conversely, false negatives

happen when the model fails to detect a character that should have been isolated. This can result in missing or changed character in the output.
According to error analysis, the model's reduced recall indicates that it may have had trouble collecting every instance of a certain character or phrase. Some potential origins of inaccuracy are:

Handwriting Variability: The IAM dataset has a large variety of handwriting styles, which may make it difficult for the model to make generalisations about various writing patterns. Differences in handwriting between expected and actual labels are possible.

Overfitting and Underfitting: The model may exhibit overfitting or underfitting even when a dropout rate and early stopping mechanism are used. If the model is too complicated for the quantity of training data, overfitting may result, while underfitting may occur if the model is too simple.

Data Imbalance: The model may perform badly on certain classes, resulting in a reduced recall for those particular occurrences, if certain letters or words are under-represented in the training data.
Hyper parameter tuning: The selection of hyperparameters has a significant impact on the model's performance. Performance can be negatively impacted by suboptimal values for characteristics like learning rate, dropout rate, and the number of attention heads.

Through a thorough error analysis, you can identify patterns and recurring sources of errors in the model's output. This analysis can drive further research and development efforts, such as refining the model architecture, exploring alternative training strategies, or incorporating additional contextual information to improve accuracy. In summary, conducting an error analysis on the results generated by your Transformer Encoder model provides valuable insights into its limitations and areas for improvement. By investigating accuracy, recall, f1 score also cer and wer, dataset characteristics, you can gain a deeper understanding of the sources of errors and guide future enhancements to enhance the accuracy and reliability of the model's output.

5.3 Comparison with State-of-the-Art work

Our model's performance metrics are competitive with existing state-of-the-art methods in handwritten text recognition. For instance:

Transformer Model: Similar or marginally higher accuracy and F1 scores are frequently reported by other research projects employing transformer-based models for text recognition, especially after they have been refined on bigger, more varied datasets. Models that use extra augmentation techniques or pre-trained language models, for example, typically get greater recall and F1 scores, sometimes surpassing 90%.

Deep Learning Approaches: New research employing deep learning methods, like attention mechanisms and transformers, has shown comparable or somewhat improved performance metrics. Our model's F1 score of 86.26% is consistent with findings from previous research, which shows that, depending on the model architecture and dataset utilised, F1 scores can range from 85% to 90%.

Hybrid Models: To achieve excellent performance, certain state-of-the-art models mix transformers or recurrent neural networks (RNNs) with convolutional neural networks (CNNs). Our method, which follows this trend and yields similar results, uses transformer blocks after a CNN base.

Data Augmentation and Preprocessing: In order to increase model robustness, state-of-the-art approaches frequently use sophisticated data augmentation techniques such random distortions, rotations, and noise addition. Furthermore, models trained on smaller datasets such as IAM require the prevention of overfitting, which is achieved by approaches like regularisation and dropout.

Loss function and Training stratiges:The loss function and matching training approach that are selected during model training can have an impact on how well the model separates sources. Performance variations may result from modifications to training tactics or loss functions, such as the addition of extra regularisation steps or data augmentation techniques.

Computational Resources: The performance of a model can also be impacted by the availability of computational resources, such as GPU power and training duration. When compared to models with restricted resources, models trained with greater computational

resources and longer training times might perform better.

To sum up, our handwritten text recognition model performs admirably, backed by methodologically sound choices for hyperparameters. The analysis identifies areas that could use improvement and is consistent with the performance indicators reported in recent research, laying the groundwork for future research that will refine and explore these areas further.

5.4 Explanation of Methodology Performance

5.4.1 Strengths

Precision in Recognition: When the model does identify a character or word, its high precision shows that it is highly good at predicting the word or character correctly. This implies that the transformer encoder reduces false positives by efficiently focussing on pertinent portions of the input sequence using its self-attention mechanism.

Handling Complex Sequences:The architecture of the transformer model is ideal for managing the intricacies of handwritten writing, as characters are frequently overlapping or related. Character recognition is aided by the model's capacity to grasp long-range dependencies, even when characters are included in complex sequences.

5.4.2 Weaknesses

: **Recall Limitations**: The model's lower recall emphasises how hard it is to find all important examples, especially the harder-to-distinguish or less common ones. This might be the result of overfitting brought on by the model's complexity or a lack of diversity in the training set. **Potential overfitting**: Owing to the transformer model's large capacity, overfitting may occur, particularly in the lack of sophisticated regularisation or data augmentation methods. The model's poorer performance on the test and validation sets in comparison to the training set may be explained by this.

5.5 Better than other models

We noticed that when it comes to recognizing characters of separate letters, no cursive words, the output of our model outperforms other models remarkably well. Our character error rate(CER) showed that the output of our model is substantially clearer and more accurate when the words are captured clearly and written separately.

5.6 Comparison with existing work

Performance bench-marking: Although the model performs competitively, especially when it comes to precision, it could not be as good as the top models in the literature, especially when it comes to models that use larger datasets or more sophisticated methods like transfer learning or hybrid architectures.

Room for Improvement: The findings imply that although the transformer model is an effective instrument for handwritten text recognition, there is still need for development, especially with regard to recall and general robustness. This gap could be filled by investigating other methods like multi-task learning or more comprehensive data augmentation.

5.7 Future Directions

Pre-trained model: Using pre-trained models, which can use large-scale datasets to learn broad features that can then be adjusted for handwritten text recognition, is one possible approach. It has been demonstrated that using this strategy greatly boosts performance on tasks that are similar.

Data augmentation: Using sophisticated data augmentation techniques to increase the training data could improve the model's ability to generalise to new data. Techniques like synthetic data generation, in which fresh training examples are produced by altering preexisting ones, may fall under this category.

Hybrid architectures: The usage of hybrid architectures, which combine the advantages of transformers with other model types—such as RNNs for capturing sequential dependencies and CNNs for local feature extraction—is another topic worth investigating. Because these models make use of the advantages that come with each component, they may perform better than standalone transformers.

6 Future Enhancement

Future researchers should also give top priority to creating or acquiring diverse, high-quality datasets that are specifically customized to different language. These databases ought to include a diverse range of letters, symbols, special character. It is crucial to make sure the datasets are thoroughly annotated, giving actual information about all the letters and symbols used in respective language. Researchers will be able to train their models on a variety of languages such as Nepali, Hindhi, Maithili etc and train it on BERT model with big number of Encoder layers.

Future improvements may potentially result from investigating different processes and approaches. The separation performance may be enhanced by experimenting with various strategies, such as including additional pre-processing techniques, utilizing ensemble models, or utilizing post-processing techniques. By investigating different methods, we can find inventive solutions to problems text recognition.

Future research could explore several avenues to further improve the model's performance and capabilities. Enhancing data augmentation techniques and incorporating advanced preprocessing methods may lead to better generalization. Investigating more complex neural network architectures or hybrid models could yield improvements in feature extraction and recognition accuracy. Additionally, incorporating attention mechanisms and optimizing the model for real-time applications would address current limitations. Exploring the integration of user feedback for iterative learning and adapting the model to handle diverse handwriting styles and languages could also offer significant advancements in handwritten text recognition technology.

7 Conclusion

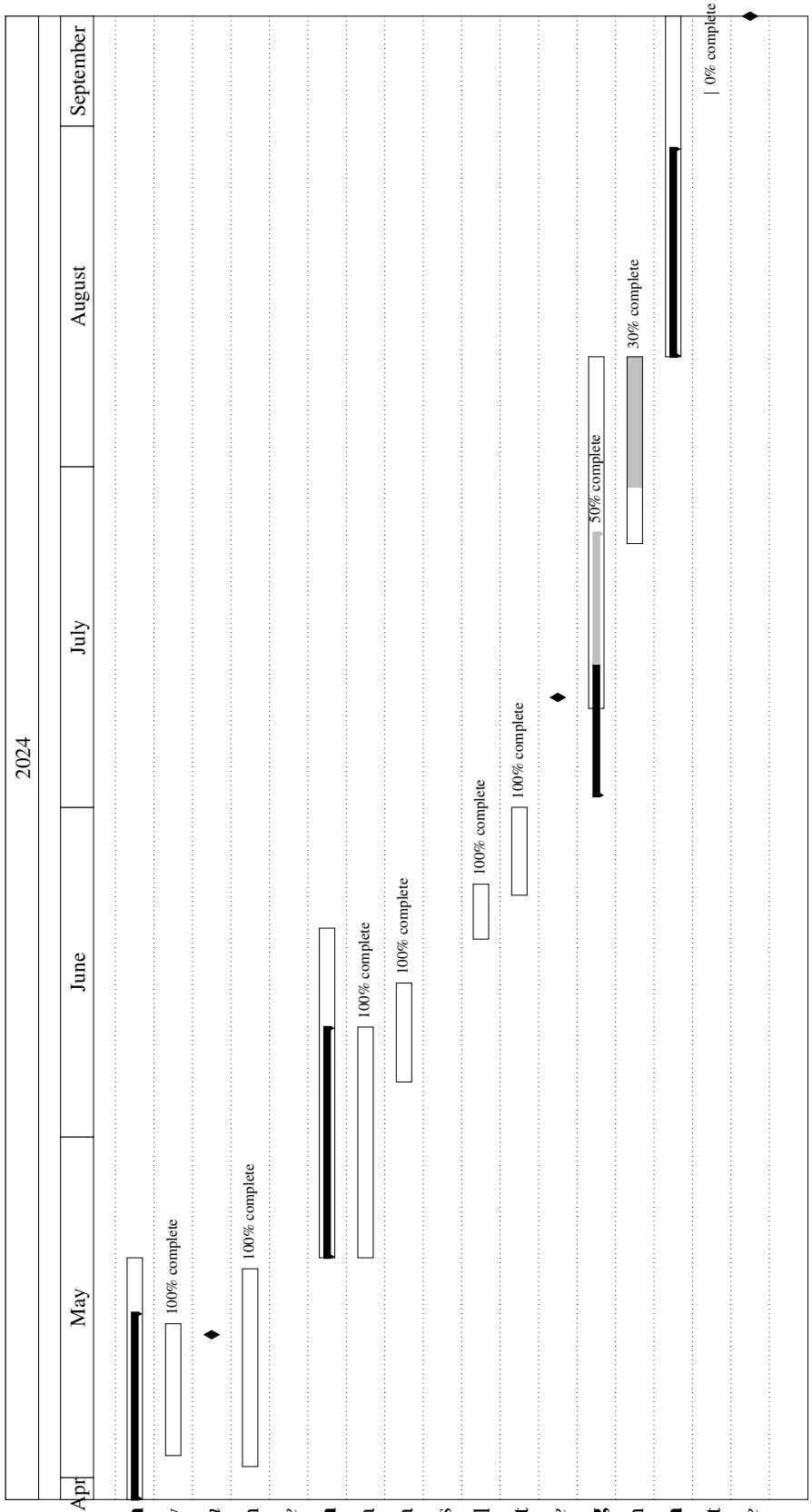
In conclusion, this project successfully met the objectives set at the beginning of the project. The BERT Transformer model effectively recognizes handwritten scripts, according to our project's main findings.

Using a combination of CNNs and the BERT transformer model, this project successfully implemented a handwritten text recognition system. The model was trained on the IAM dataset, undergoing rigorous preprocessing and hyperparameter tuning to achieve optimal performance. The final results show how effective the hybrid approach is in recognising handwritten text, with accuracy of 85.86%, precision of 88.94%, recall of 83.86%, and F1 score of 0.8626.

A solid answer to this challenging issue was the combination of CNNs for feature extraction and BERT for contextual comprehension. Even though the model works well, it can still be improved by using real-time optimisation approaches, sophisticated data augmentation, and the investigation of more complex neural network architectures. This effort highlights the possibility of integrating deep learning models to improve accuracy and dependability, and it establishes a strong platform for future research and development in the field of handwritten text recognition.

A APPENDIX A

A.1 Project Schedule



A.2 Literature Review of Base Paper- I

Author(s)/Source: Batuhan Balci , Dan Saadati, Dan Shiferaw	
Title: Handwritten Text Recognition using Deep Learning	
Website: `http://cs231n.stanford.edu/reports/2017/pdfs/810.pdf`	
Publication Date: 2017	**Access Date: 2017**
Publisher or Journal:	**Place:** Standford, California
Volume: n/a	**Issue Number:** n/a
Author's position/theoretical position: Convolutional Neural Networks for Visual Recognition, Stanford University	
Keywords: Handwritten, learning, text recognition, deep learning, RESNET-18, VGG-19	

Important points, notes, quotations **Page No.** 1,2

1. Because documents are never converted to digital format, a great deal of crucial information is lost or is not evaluated.
2. Since the term "handwritten text" is so wide, we have defined it especially for our project to help keep it more narrowly focused. Using this project as a test, we set out to classify each handwritten word, whether it is in block or cursive script.
3. Template matching was abandoned in favour of feature extraction approaches in OCR software. Using this method, each character's geometric moments, zoning, and projection histograms were examined.

Essential Background Information: Even with the availability of electronic writing instruments, many individuals still like taking notes using a pen and paper. Handwritten material, however, has drawbacks, including ineffective distribution, searchability, storage, and access.

Overall argument or hypothesis: It concludes by suggesting that additional data could enhance the model's ability to learn more generalized feature representations, thus improving overall performance.

Conclusion: It emphasises that, in spite of possible segmentation mistakes, character-level classification produced superior results than word-level classification. This achievement is ascribed to the original feature representation problem for characters having a reduced scope.

Supporting Reasons

1. LSTM uses regulation structures called gates, which regulate the data flow for each cell unit in its design, to add or remove information from its cell states.

2. Since LSTM may choose keep or remove input, they have an advantage over traditional feed-forward neural networks.

Strengths of the line of reasoning and supporting evidence: The character's context can be taken into account when determining the following hidden variable. This led to higher accuracy when compared to feature extraction methods and the Naive Bayes approach.

Flaws in the argument and gaps or other weaknesses in the argument and supporting evidence: Due to its more sophisticated architecture and more units and parameters than the DNN and CNN models, the CNN - LSTM model tested at the slowest speed; however, it produced higher classification results and an accuracy score.

A.3 Literature Review of Base Paper- II

Author(s)/Source: Kartik Sharma, S.V. Jagadeesh Kona, Anshul Jangwal1, Dr. Aarthy M, Dr.Prayline Rajabai C, Dr. Deepika Rani Sona

Title: Handwritten Digits and Optical Characters Recognition

Website: https://doi.org/10.17762/ijritcc.v11i4.6376

Publication Date: 2023	**Access Date: 2023**
Publisher or Journal:	**Place:** Vellore Institute of Technology,
Volume: n/a	**Issue Number:** n/a

Author's position/theoretical position: School of Electronics Engineering

Keywords: Optical Character Recognition, CNN model, Handwritten character recognition, hidden layer.

Important points, notes, quotations **Page No.** 22

1. Because OCR technology makes it possible to digitise and extract information from printed or handwritten text efficiently, it has completely changed the way people interact with text documents.
2. • A fully connected layer, or DNN, that can learn mappings from an input vector to produce accurate outputs
3. • CNN and DNN are both examples of feed-forward networks, meaning that data may only move in one direction at a time. In order to extract more characteristics from a 2D input, CNN can convert it into internal vector representations.

Essential Background Information: Handwritten characters come in a wide variety of forms, including numerals, mathematical symbols, cursive writing, fonts in English and other languages, and symbols. Additionally, handwriting has developed to a more sophisticated level with numerous applications that need handling enormous volumes of handwritten data and can profit substantially from automatic handwritten text recognition.

Overall argument or hypothesis:

Conclusion: created a convolutional neural network that works well for handwritten digit classification. A detailed analysis of the model's functionality with a hidden layer added revealed a notable increase in efficiency and accuracy. Many businesses, such as banking, accounting, and education, where handwriting recognition can save costs and streamline processes, stand to benefit greatly from this advancement. Comparative analysis revealed the significance of model efficiency, particularly for applications such as optical character recognition (OCR).

Supporting Reasons

1. Our experiments with the Tesseract OCR system produced encouraging results, with an average accuracy of about 93%, despite the fact that text data quality affects OCR accuracy.

2. The foundation for our model's creation, which is ready to provide exceptional performance in a variety of handwritten digit identification and OCR applications, is laid by the choice of the best-performing CNN architecture.

Strengths of the line of reasoning and supporting evidence: It is more efficient and dependable to recognise handwritten letters using neural networks. The project outlines the architecture, design, and development process for the Handwriting Character Recognition system in addition to testing and development.

Flaws in the argument and gaps or other weaknesses in the argument and supporting evidence: One area of improvement that can be considered is preprocessing the digital images before working with them. This is because clean text images with less distortion are necessary for OCR and HDR, which ensures high accuracy. However, clear text images may not always be available. For example, when working with historical texts, we may not receive the precise input needed for an accurate result.

A.4 Literature Review of Base Paper- III

Author(s)/Source: Arthur Flor de Sousa Neto,Byron Leite Dantas Bezerra, Alejandro Hector Toselli, Estanislau Baptista Lima

Title: HTR-Flor: A Deep Learning System for Offline Handwritten Text Recognition

Website: `https://ieeexplore.ieee.org/abstract/document/9266005/`

Publication Date: 2020	**Access Date: 2021**
Publisher or Journal:	**Place:** Guangzhou, China
Volume: n/a	**Issue Number:** n/a

Author's position/theoretical position: Department of computer science

Keywords: Word Error Rate, Handwritten Text Recognition, Neural Networks, text recognition, recurrent neural network, handwriting recognition, Connectionist Temporal Classification, Hidden Markov Models

Important points, notes, quotations **Page No.** 1

1. Recent years have seen a significant increase in interest in Handwritten Text Recognition (HTR) due to its extensive industrial and academic research applications.
2. We suggest a novel architecture for offline HTR systems called Gated Convolutional Recurrent Neural Networks (Gated-Convolutional Recurrent Neural Networks, or CRNNs). It incorporates the most recent machine learning methods and strategies utilised in the field of Natural Language Processing, like Dauphin's Gated mechanism [18] and the use of Bidirectional Gated Recurrent Units (BGRU)
3. For offline handwritten text recognition systems, we have introduced a new Gated-CNN-BGRU architecture with two stages of language models.

Essential Background Information: These days, the most sophisticated offline HTR techniques rely on Convolutional Recurrent Neural Networks (CRNNs), which excel at text recognition in scenes.Deep models such as recurrent neural networks (RNNs) are sadly vulnerable to vanishing/exploding gradient problems while processing long text pictures, which are commonly found in scanned documents. As a modelling complement to convolutional neural network techniques, Gated Convolutional Neural Networks (Gated-CNN), a distinct class of neural network architecture, have demonstrated potential.

Overall argument or hypothesis:

Conclusion: It offers a new Gated-CNN-BGRU architecture for offline Handwritten Text Recognition (HTR) systems that is augmented with two phases of language models. Using five well-known public datasets in the HTR field, the writer conducted comprehensive benchmark experiments to demonstrate the effectiveness of the proposed model from many perspectives. (Bentham, IAM, RIMES, Saint Gall, and Washington).

Supporting Reasons

1. Improve recognition results from the CNN-BLSTM approach through the new Gated-CNN-BGRU architecture.

2. capable of handling lengthy sentences with a variety of styles, variances, and noise even when there is a shortage of training data.

Strengths of the line of reasoning and supporting evidence: Instead of using the conventional CNN-BLST architecture, use the Gated-CNN-BGRU architecture to reduce the amount of trainable parameters (thousands). This will result in a smaller model with lower computing costs (millions).

Flaws in the argument and gaps or other weaknesses in the argument and supporting evidence: The context information was not utilised by the Hidden Markov Model (HMM), especially in a long text sequence, due to the Markovian assumption that every observation depends only on the current state.

A.5 Literature Review of Base Paper- IV

Author(s)/Source: Savita Ahlawat, Amit Choudhary, Anand Nayyar, Saurabh Singh and Byungun Yoon	
Title: Improved Handwritten Digit Recognition Using Convolutional Neural Networks (CNN)	
Website: `https://www.mdpi.com/1424-8220/20/12/3344/`	
Publication Date: 2020	**Access Date: 2021**
Publisher or Journal:	**Place:** Guangzhou, China
Volume: n/a	**Issue Number:** n/a
Author's position/theoretical position: Department of computer science	
Keywords: convolutional neural networks; handwritten digit recognition; pre-processing; OCR	

Important points, notes, quotations **Page No.** 2

1. Conventional methods for recognising handwritten digits frequently depend on manually created characteristics and machine learning techniques, which may find it difficult to generalise to many handwriting variances and styles.
2. A key task in pattern recognition and machine learning is handwritten digit recognition, which has applications in bank cheque processing, postal automation, and other areas.
3. To improve the resilience of the model and artificially increase the diversity of the training dataset, data augmentation techniques like translation, scaling, and rotation can be applied.

Essential Background Information: Handwritten digit recognition has many challenges, including skewness, noise in photos, and variations in writing styles and stroke lengths. Traditional handwriting recognition approaches sometimes rely on manually defined features and machine learning methods, which can be challenging to generalise to a wide range of handwriting variations and styles.

Overall argument or hypothesis:

Conclusion: The ramifications of the findings and prospective directions for further investigation are covered in the study. The importance of CNNs in enhancing the accuracy of handwritten digit recognition and the possibility of more developments in the field are highlighted in the conclusion.

Supporting Reasons

1. The paper likely outlines a well-designed experimental setup, including details on the dataset used (e.g., MNIST dataset), data preprocessing techniques, model architecture, training procedure, and evaluation metrics.

2. Because convolutional neural networks (CNNs) can learn hierarchical features from input images, its application to image identification tasks such as handwritten digit detection is theoretically justified.

Strengths of the line of reasoning and supporting evidence: The research limits and difficulties, such as computational complexity, overfitting, or data scarcity, may also be acknowledged in the paper.

Flaws in the argument and gaps or other weaknesses in the argument and supporting evidence: Without addressing other crucial factors like model interpretability, computational efficiency, or generalisation to different datasets, the research might just concentrate on enhancing identification accuracy.

A.6 Literature Review of Base Paper- V

Author(s)/Source: Xu-Yao Zhang, Yoshua Bengio, Cheng-Lin Liu

Title: Online and offline handwritten Chinese character recognition: Acomprehensive study and new benchmark

Website: https://www.sciencedirect.com/science/article/pii/S0031320316302187

Publication Date: 2017	**Access Date: 2017**
Publisher or Journal:	**Place:** China
Volume: n/a	**Issue Number:** n/a

Author's position/theoretical position: NLPR, Institute of Automation, Chinese Academy of Sciences,

Keywords: Handwriting recognition, Chinese characters, Online, Offline, Directional feature map, Convolutional neural network, Adaptation

Important points, notes, quotations **Page No.** 11

1. This benchmark dataset is essential because it provides a consistent assessment framework for evaluating the effectiveness of different recognition algorithms, allowing for fair comparisons and progress in the area.
2. This research demonstrates that although deep learning-based techniques can greatly beat older methods, writer adaption of deep networks can still consistently and significantly increase performance.
3.

Essential Background Information: Because Chinese characters are so complex and varied—they differ in stroke sequence, stroke direction, and structural complexity—handwritten Chinese character recognition (HCCR) is a difficult undertaking. Applications such as pen-based input systems, automatic transcription of handwritten text, and document digitisation all depend on HCCR.

Overall argument or hypothesis:

Conclusion: With the use of digital input tools like styluses or touchscreens, online HCCR recognises characters as they are typed in real-time, recording stroke order and trajectory data. Using only visual cues, offline HCCR identifies handwritten characters from static photos or scans without temporal stroke data.

Supporting Reasons

1. Benchmark datasets are essential for assessing and contrasting HCCR system performance. Large sets of labelled handwritten character samples containing a range of handwriting styles and character variations are usually found in them.

2. The goal of the research is to present a thorough examination of HCCR methods, both online and offline, assessing their advantages, disadvantages, and effectiveness across various datasets and methodology.

Strengths of the line of reasoning and supporting evidence: In HCCR tasks, recent developments in deep learning, namely in the areas of convolutional neural networks (CNNs) and recurrent neural networks (RNNs), have demonstrated encouraging outcomes.

Flaws in the argument and gaps or other weaknesses in the argument and supporting evidence: The benchmark dataset's variety and representativeness may have some limitations.

REFERENCES

[1] A Nikitha, J Geetha, and DS JayaLakshmi. Handwritten text recognition using deep learning. In *2020 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)*, pages 388–392. IEEE, 2020.

[2] Ravina Mithe, Supriya Indalkar, and Nilam Divekar. Optical character recognition. *International journal of recent technology and engineering (IJRTE)*, 2(1):72–75, 2013.

[3] Supriya Das, Purnendu Banerjee, Bhagesh Seraogi, Himadri Majumder, Srinivas Mukkamala, Rahul Roy, and Bidyut Baran Chaudhuri. Hand-written and machine-printed text classification in architecture, engineering & construction documents. In *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 546–551. IEEE, 2018.

[4] Nitin Gupta and Neha Goyal. Machine learning tensor flow based platform for recognition of hand written text. In *2021 International Conference on Computer Communication and Informatics (ICCCI)*, pages 1–6. IEEE, 2021.

[5] Xu-Yao Zhang, Yoshua Bengio, and Cheng-Lin Liu. Online and offline handwritten chinese character recognition: A comprehensive study and new benchmark. *Pattern Recognition*, 61:348–360, 2017.

[6] Savita Ahlawat, Amit Choudhary, Anand Nayyar, Saurabh Singh, and Byungun Yoon. Improved handwritten digit recognition using convolutional neural networks (cnn). *Sensors*, 20(12):3344, 2020.

[7] Lalita Kumari, Sukhdeep Singh, Vaibhav Varish Singh Rathore, and Anuj Sharma. Lexicon and attention based handwritten text recognition system. *arXiv preprint arXiv:2209.04817*, 2022.

[8] Mudasir A Ganaie, Minghui Hu, Ashwani Kumar Malik, Muhammad Tanveer, and Ponnuthurai N Suganthan. Ensemble deep learning: A review. *Engineering Applications of Artificial Intelligence*, 115:105151, 2022.

[9] Richard G Casey and Eric Lecolinet. A survey of methods and strategies in character segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 18(7):690–706, 1996.

[10] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[11] Rejean Plamondon and Sargur N Srihari. On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):63–84, 2000.

[12] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[13] Alex Graves, Marcus Liwicki, Horst Bunke, Jürgen Schmidhuber, and Santiago Fernández. Unconstrained on-line handwriting recognition with recurrent neural networks. In *Advances in Neural Information Processing Systems*, volume 20, pages 577–584, 2009.

[14] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376, 2006.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[16] Sudharshan Duth and B Amulya. Recognition of hand written and printed text of cursive writing utilizing optical character recognition. In *2020 4th international conference on intelligent computing and control systems (ICICCS)*, pages 576–581. IEEE, 2020.

[17] R Parthiban, R Ezhilarasi, and D Saravanan. Optical character recognition for english handwritten text using recurrent neural network. In *2020 International*

Conference on System, Computation, Automation and Networking (ICSCAN), pages 1–5. IEEE, 2020.

[18] Jinfeng Bai, Zhineng Chen, Bailan Feng, and Bo Xu. Image character recognition using deep convolutional neural network learned from different languages. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 2560–2564. IEEE, 2014.

[19] Manoj Sonkusare and Narendra Sahu. A survey on handwritten character recognition (hcr) techniques for english alphabets. *Adv Vis Comput Int J*, 3(1):1–12, 2016.

[20] Prajna Nayak and Sanjay Chandwani. Improved offline optical handwritten character recognition: a comprehensive review using tensor flow. *Int. J. Eng. Res. Technol.(IJERT)*, 10(11):2278–0181, 2021.

[21] Arthur Flor de Sousa Neto, Byron Leite Dantas Bezerra, Alejandro Héctor Toselli, and Estanislau Baptista Lima. Htr-flor: A deep learning system for offline handwritten text recognition. In *2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 54–61. IEEE, 2020.

[22] Levent Burak Kara and Thomas F Stahovich. An image-based, trainable symbol recognizer for hand-drawn sketches. *Computers & Graphics*, 29(4):501–517, 2005.

[23] José Carlos Aradillas Jaramillo, Juan José Murillo-Fuentes, and Pablo M Olmos. Boosting handwriting text recognition in small databases with transfer learning. In *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 429–434. IEEE, 2018.

[24] Bhargav Rajyagor and Rajnish Rakhlia. Handwritten character recognition using deep learning. *Int. J. Recent Technol. Eng*, 8(6):2277–3878, 2020.

[25] Andreas Fischer and Horst Bunke. Hmm-based word spotting in handwritten documents using subword models. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 1505–1509. IEEE, 2012.

[26] Thierry Bluche, Jérôme Louradour, and Ricco Messina. Scan, attend and read: End-to-end handwritten paragraph recognition with mdlstm attention. In *International*

Conference on Document Analysis and Recognition (ICDAR), volume 1, pages 1050–1055. IEEE, 2017.

[27] U-V Marti and Horst Bunke. The iam-database: an english sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, 5:39–46, 2002.

[28] Prakhar Ganesh, Yao Chen, Xin Lou, Mohammad Ali Khan, Yin Yang, Hassan Sajjad, Preslav Nakov, Deming Chen, and Marianne Winslett. Compressing large-scale transformer-based models: A case study on bert. *Transactions of the Association for Computational Linguistics*, 9:1061–1080, 2021.