WriteAI:
Handwritten Text Recognition Using Neural Networks

PROJECT REPORT

Submitted by

Allen Y (TVE19CS011)

Alvin Antony K (TVE19CS012)

Ardra T S (TVE19CS018)

Krishnapriya V P (LTVE19CS070)

to

The APJ Abdul Kalam Technological University

in partial fulfillment of the requirements for the award of the Degree

of

Bachelor of Technology

in

Computer Science and Engineering

Department of Computer Science and Engineering

College of Engineering Trivandrum

Kerala

May 23, 2023

DECLARATION

We undersigned hereby declare that the project report **WriteAI: Handwritten Text Recognition Using Neural Networks**, submitted for partial fulfillment of the requirements for the award of degree of Bachelor of Technology of the APJ Abdul Kalam Technological University, Kerala is a bonafide work done by us under supervision of **Dr Dhanya S Pankaj**. This submission represents our ideas in our own words and where ideas or words of others have been included, we have adequately and accurately cited and referenced the original sources. We also declare that we have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in our submission. We understand that any violation of the above will be a cause for disciplinary action by the institute and/or the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been obtained. This report has not been previously formed the basis for the award of any degree, diploma or similar title of any other University.

Place: Thiruvananthapuram

Date: June 15, 2023

Allen Y

Alvin Antony K

Ardra T S

Krishnapriya V P

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
COLLEGE OF ENGINEERING TRIVANDRUM

CERTIFICATE

This is to certify that the report entitled **“WriteAI: Handwritten Text Recognition Using Neural Networks",** submitted by **Allen Y**, **Alvin Antony K**, **Ardra T S**, **Krishnapriya V P** to the **APJ Abdul Kalam Technological University** in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a bonafide record of the project work carried out by them under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Dr Dhanya S Pankaj	**Dr. Piyoosh P**	**Dr. Sumesh Divakaran**
Assistant Professor	Assistant Professor	Professor
Department of CSE	Department of CSE	Department of CSE
(Guide)	(Coordinator)	(Head of Department)

ACKNOWLEDGEMENT

First and foremost, we would like to express our sincere gratitude and heartful indebtedness to our guide **Dr Dhanya S Pankaj**, Assistant Professor, Department of Computer Science and Engineering for his/her valuable guidance and encouragement in pursuing this project.
We are also thankful to **Dr. Sumesh Divakaran**, Head of the Department for his help and support. We also extend our hearty gratitude to project co-ordinator, **Dr. Piyoosh P**, College of Engineering Trivandrum for providing necessary facilities and their sincere co-operation.
Our sincere thanks is extended to all the teachers of the Department of Computer Science and Engineering and to all my friends for their help and support.
Above all, we thank God for the immense grace and blessings at all stages of the project.

Allen Y
Alvin Antony K
Ardra T S
Krishnapriya V P

ABSTRACT

Handwritten Text Recognition (HTR) is a technology that enables computers to recognize and interpret handwriting from various sources such as images, documents, online tools, and other devices. The difficulty in accurately recognizing complex handwritten characters and digits is a significant challenge that requires attention. Ongoing studies aim to enhance Handwritten Text Recognition accuracy in Optical Character Recognition (OCR). Handwritten Text Recognition is a rapidly growing field in computer vision. Humans possess the innate capacity to effortlessly recognize objects and text. However, this is not the case for machines, as recognizing handwritten text is a challenging task for them. The process of text recognition involves processing an input image, extracting features, and using a classification scheme. The system undergoes training to identify similarities and distinctions among different handwritten samples, making it possible to convert an image of handwritten text into digital text. The dataset used is IAM Handwriting Database 3.0. The IAM Dataset comprises contributions from 657 writers, including 1532 pages of scanned text, 5685 isolated and labeled sentences, 13353 isolated and labeled text lines, and 115320 isolated and labeled words. We have developed a deep learning model for Handwritten Text Recognition in this project. The process begins with an input image of handwritten text, which is segmented into lines. These lines are further divided into individual words. The word images are then passed through a specially trained system designed for recognizing handwritten text. The recognized output undergoes post-processing to produce the final output as an editable text document. The model is comprised of 5 Convolutional Neural Networks (CNN) for feature extraction, two Recurrent Neural Networks (RNN) layers, and a Connectionist Temporal Classification (CTC) layer to calculate loss. Deep learning model created to recognize handwritten text has been transformed into a mobile application that accepts images of handwritten text as input and converts them into digital text.

Contents

List of Figures

ABBREVIATIONS

(List in the alphabetical order)

- **ANN** Artificial Neural Network
- **CNN** Convolutional Neural Network
- **CTC** Connectionist Temporal Classification
- **HTR** Handwritten Text Recognition
- **LM** Language Model
- **LSTM** Long short-term memory
- **NLP** Natural Language Processing
- **RELU** Rectified Linear Activation Unit
- **WBS** Word Beam Search
- **VBS** Vanilla Beam Search
- **OCR** Optical Character Recognition
- **RNN** Recurrent Neural Network

Chapter 1

Introduction

The study of handwritten text recognition poses a significant challenge in the field of computer vision and pattern recognition.. In several fields, including historical document research, postal services, archive digitalization, and automated form processing, handwritten text is extremely valuable. However, successful recognition of handwritten writing is significantly hampered by the variety and complexity that handwritten text inherently possesses. To enable text recognition, the system must undergo training to recognize characters. Character recognition involves several steps, which include data acquisition, feature extraction, classification, and recognition. Handwriting recognition refers to a machine's capability to comprehend and interpret handwritten input from an image source. The difficulties of handwritten text recognition have given rise to the development of sophisticated techniques like neural networks. They enable more precise and reliable recognition by allowing users to directly learn complicated patterns and representations from data. Comparing neural network-based models to conventional techniques that depend on manually created features and statistical models, it has been shown that the latter exhibit notable performance increases. The goal of this project is to develop a neural network-based system that efficiently and accurately recognizes characters. In addition to recognizing handwriting, a handwriting recognition system must effectively handle formatting, accurately segment characters, and determine the most likely words.

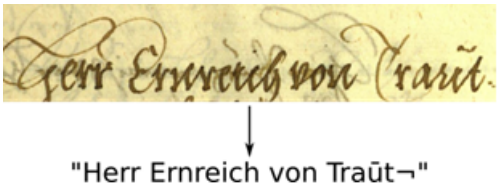

Figure 1.1: Handwritten Text Recognition (HTR) illustrated

Figure 1.1 illustrates Handwritten Text Recognition. Handwritten Text Recognition (HTR) refers to the process of converting handwritten text into digital text. It is categorized into two types: online and offline recognition. Online recognition occurs in real-time as the text is being written, often utilizing devices with pressure sensitivity to capture both geometric and temporal information. In contrast, offline recognition takes place after the text has been written and captured, typically through scanning. Online recognition is generally considered a simpler problem. However, HTR as a whole faces several challenges, including the cursive nature of handwriting, the variability in size and shape of each character, and the large number of characters within the vocabulary.

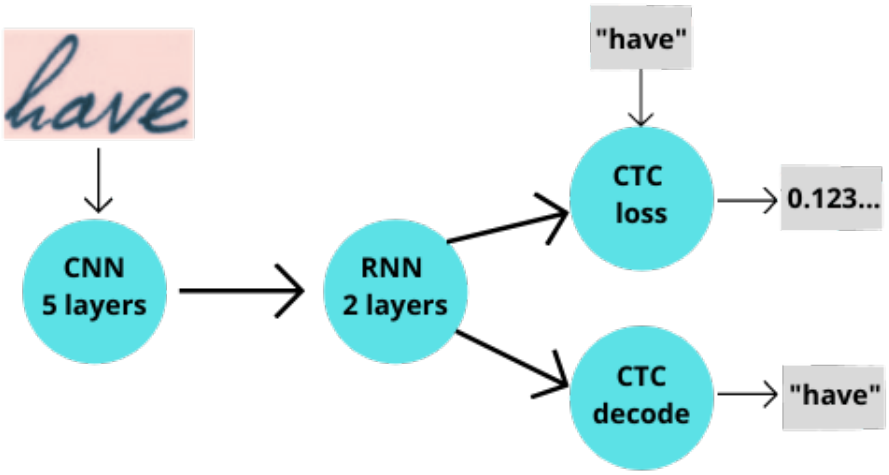

Figure 1.2: Overview of the proposed HTR system

Figure 1.2 represents the overview of the proposed HTR system. The input image undergoes processing through multiple layers of Convolutional Neural Network (CNN) to identify significant features. These layers generate a sequence of 1D or 2D feature maps, which are then passed to Recurrent Neural Network (RNN) layers. The RNN processes this sequence of information. The RNN's output is transformed into a matrix that represents the character scores for each element in the sequence. The final text is obtained using a decoding algorithm, which is based on a specific coding method the neural network is trained on. The decoding process utilizes a Connectionist Temporal Classification (CTC) operation, which can be enhanced by incorporating a Language Model (LM).

Chapter 2 of this report covers the research papers analyzed to complete the project at hand. In chapter 3, the problem statement and key objectives that the project wishes to fulfil has been described in detail. Chapter 4 discusses about the software requirement specification and software design documents. The project is concluded with a discussion on the results obtained in chapter 5 and the conclusions and future scope in chapter 6.

Chapter 2

Existing Methods

The implementation of the project “WriteAI: Handwritten Text Recognition Using Neural Networks” was completed using ten research papers as inspiration towards building initial supervised model.

This review’s objective is to pinpoint the most commonly used data sources, feature extraction methods, and classification algorithms employed in this field. This literature review will analyze the strengths and limitations of each approach to Handwritten Text Recognition, aiming to offer valuable insights into the current cutting-edge. Additionally, it will propose future prospective avenues for study in this field.

2.1 Handwritten Character Recognition using Deep Learning in Android Phones (1)

The proposed Handwriting recognition system encompasses two main components: the front-end and the back-end, each consisting of multiple modules. The front-end of the application utilizes Flutter, which is a Google open-source user interface software development kit. Flutter allows for the creation of a single codebase that can be used to develop applications across multiple platforms, including Android, iOS, Linux, Mac, Windows, Google Fuchsia, and the web.

On the other hand, the back-end of the system incorporates several technologies including Keras, OpenCV, TensorFlow Lite, Google Text-to-Speech, and Firebase. These technologies collectively contribute to different aspects of the system’s functionality. Let’s delve into the

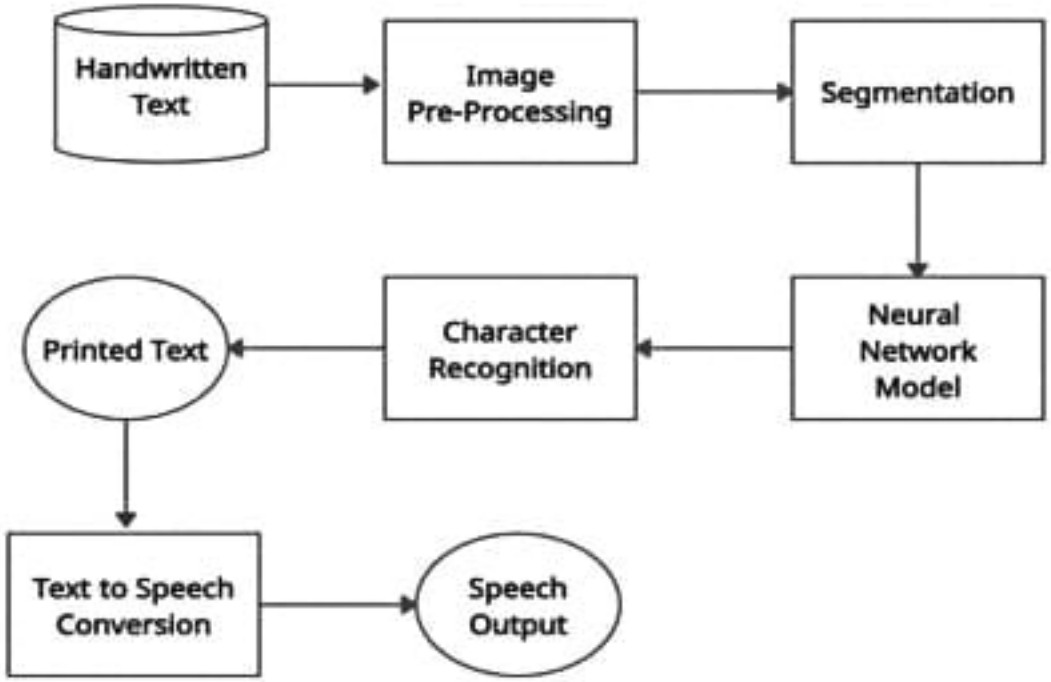

Figure 2.1: An overview of the system [1]

details of each module within the back-end:
Figure 2.1 [1] shows the overview of the system used in this paper

1. Registration and authentication: This module handles user registration and authentication processes, ensuring secure access to the system's features.

2. Image processing: This module focuses on processing the input image of handwritten text, employing techniques and algorithms from OpenCV to enhance image quality, remove noise, and prepare the image for subsequent analysis.

3. Neural network modeling and training: In this module, a deep learning neural network model is constructed and trained using Keras and TensorFlow. A Convolutional Neural Network (CNN) is utilized for recognition and classification tasks. The CNN follows steps such as input image processing, image processing, comparison and prediction, and generates printed text as the output. Two bidirectional LSTMs are incorporated to handle error backflow issues. The CTC loss function is employed to calculate the

loss, considering all possible alignments of the ground truth. This approach enhances accuracy and facilitates the digital transformation of handwritten text.

4. Character recognition: Once the neural network model is trained, this module utilizes it to recognize individual characters within the handwritten text image, enabling accurate and efficient character recognition.

5. Text-to-speech conversion: The final module utilizes the Google Text-to-Speech technology, integrated into the system through Firebase, to convert the recognized digital text into spoken words, providing an auditory output for the user.

By employing a combination of Flutter, Keras, OpenCV, TensorFlow Lite, Google Text-to-Speech, and Firebase, the Handwriting recognition system offers a comprehensive solution for converting handwritten text into digital text, facilitating efficient and user-friendly interaction with various devices and platforms.

2.2 A SCALABLE HANDWRITTEN TEXT RECOGNITION SYSTEM (2)

In order to represent the text in a single-line picture, handwritten text line recognition aims to provide a series of Unicode code points. The picture is processed before being fed into a neural network. A 1-D series of logits is then generated by the neural network, each of which represents a character or a unique blank symbol. The CTC loss is used to train the network, which gives it the ability to handle pictures of various widths.

A character-based n-gram language model's cost and the logits cost are integrated during the inference stage using a log-linear method. In order to find the optimal recognition outcome, beam search is used. In this article, two different model designs are examined.

The LSTM-based model draws inspiration from the CLDNN architecture (Convolutions, LSTMs, Deep Neural Network). It incorporates LSTM and other recurrent neural networks commonly employed in contemporary models for recognizing handwritten text lines.

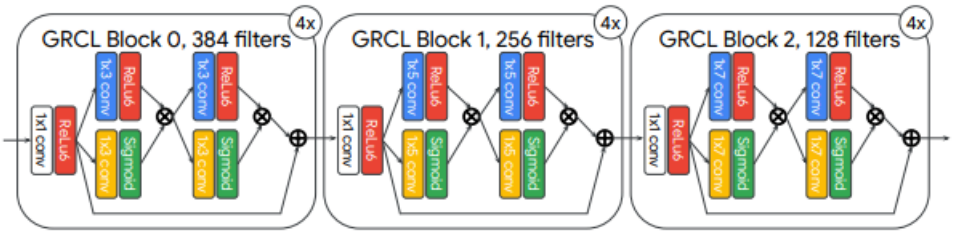

Figure 2.2: Gated recurrent convolutional layer [2]

Figure 2.2 [2] depicts the block diagram of Gated recurrent convolutional layer Recurrence-free model: Recurrent models, while effective, are limited in their ability to parallelize computations on accelerators like GPUs and TPUs. To address this limitation, we propose a solution that aims to overcome the mentioned challenge. A recurrence-free model that employs 1-D gated recurrent convolutional layers. This model simulates the recurrency and gating mechanism of LSTMs within a feed-forward structure. Gated recurrent convolutional layers propagate information along the depth of the network while controlling the flow using a gating mechanism. The gating mechanism incorporates a sigmoid activation that interacts with original ReLU activation.

In summary, the article explores the task of handwritten text line recognition, discussing two model architectures: an LSTM-based model inspired by CLDNN and a recurrence-free model using 1-D gated recurrent convolutional layers.

2.3 Handwriting Text Recognition Using Neural Network (3)

The project's objective is to create a system that is capable of transforming a two-dimensional image containing. The goal of the project is to develop a system that can accurately convert both machine-printed and machine-readable text from handwritten text and converting it into speech. The project begins by training a network to predict text from given images. The proposed engine follows a pipeline-based architecture, consisting of sequential steps such as pre-processing, connected component analysis, character recognition, character aggregation, and addressing the detection of small capitals.

Prior to implementing the pipeline, thorough research on the history and various aspects of this technology was conducted to understand the nature of handwritten language and its conversion into electronic data. The focus of the project then shifted towards offline text recognition, considering both typed and handwritten text. TensorFlow, Python 3, Numpy, and OpenCV were utilized for the implementation of the Text Recognition (HTR) system. A neural network model trained on IAM off-line HTR dataset, consisting of convolutional neural network (CNN) layers, recurrent neural network (RNN) layers, and a Connectionist Temporal Classification (CTC) layer, was employed to recognize text in segmented word images.

Deep learning techniques were utilized for extracting characteristics and classify text, showcasing improved performance compared to traditional algorithms, particularly when dealing with large amounts of data. Additionally, a DNN-based approach using WaveNet, a deep neural network for audio generation, was employed to convert machine-readable text into speech.

The implemented system allows users to capture images containing text or handwritten data, which are then processed by the recognition engine to obtain machine-readable text. This text is subsequently passed through a text-to-speech synthesizer, enabling visually impaired individuals, such as those who are blind or have other vision impairments, to access and comprehend text and handwritten documents that were previously inaccessible to them.

2.4 HAND WRITTEN DIGIT RECOGNITION USING MACHINE LEARNING (4)

The methodology employed in this study involves several steps for hand-written digit recognition using machine learning. Here is an expanded explanation of each step:

The methodology employed in this study for hand-written digit recognition using machine learning encompasses the following steps:

1. Loading the MNIST dataset using Keras library and generating plots to visualize the dataset.

2. Reshaping the dataset and developing a easy yet effective multi-layer perceptron (MLP) model for digit recognition.

3. The methodology involves creating convolutional neural network (CNN) models using Keras. This includes designing a larger CNN architecture with convolutional layers, pooling layers, dropout layers, and fully connected layers.

4. Training the models using TensorFlow optimizer and evaluating their performance.

5. The larger CNN model achieves a respectable misclassification rate of 0.83% after 10 epochs of training.

By following this methodology, the researchers successfully developed and trained machine learning models for accurate hand-written digit recognition The specific topology of the larger CNN model is described as follows:

In this study, there are several processes involved in machine learning's recognition of handwritten digits.bThe first step is the construction of a convolutional neural network (CNN) model using Keras. This model consists of a convolutional layer with 30 feature maps of size 5×5. Next, a pooling layer is implemented to perform max pooling over 2×2 patches. Another convolutional layer is then added, which includes 15 feature maps of size 3×3. This is followed by another pooling layer that conducts max pooling over 2×2 patches. To prevent overfitting, a dropout layer is applied with a dropout rate of 20%. Then, a flatten layer is used to transform the 2D feature maps into a 1D vector. The first completely linked layer has 128 neurons with rectifier activation, while the second layer contains 50 neurons and rectifier activation. The final categorization is incorporated with an output layer. 200 batches are used during the training process, which lasts for 10 epochs. The accuracy on the training and validation datasets is assessed and reported at the end of each epoch, which finally results in the calculation of the overall misclassification rate.

2.5 CNN-RNN Based Handwritten Text Recognition (5)

In the paper, the authors describes deep learning architecture for handwritten text recognition, utilizing Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Connectionist Temporal Classification (CTC) techniques. The main contributions and findings of the study can be summarized as follows:

1. CNN Feature Extraction:

   CNN layers are employed to extrapolate pertinent details from the source picture. Each layer performs convolution, activation using ReLU, and downsizing operations to identify image regions. Output sequence is obtained as a 32x256 representation.

2. RNN Processing:

   Long Short-Term Memory (LSTM) networks are utilized to process the feature sequence. LSTM networks are chosen due to their increased capacity to detect distant relationships and enhanced training traits. In order to account for the characters included in the dataset, the output sequence from the RNN is mapped to a 32x80 matrix.

3. CTC Loss and Training:

   To decode the output matrix into text and calculate the loss, the CTC layer is utilized. This entails comparing the decoded text to the ground truth text to determine the loss value. Following that, the average loss value is used to train the neural network (NN) using RMSProp optimizer.

4. Recognition and Evaluation:

   The trained model is used to recognize input images and calculate the word error rate. The word error rate for the model is reported as 10.62%, displaying the proportion of mistakes in the recognised text compared to the original.

In conclusion, the proposed architecture effectively combines CNNs and RNNs to perform handwritten text recognition. The utilization of LSTM networks and CTC loss helps in achiev-

ing accurate recognition results. The reported word error rate demonstrates performance and potential of the developed model in recognizing handwritten text with satisfactory accuracy.

2.6 Recognition of Handwritten Characters based on Deep Learning with Tensorflow (6)

An impressive convolutional neural network (CNN) model is developed in this study to accomplish the task of recognizing handwritten digits. By leveraging various hidden layers and epochs, the model demonstrates exceptional accuracy and performance.

Using the popular MNIST dataset, the model was trained and evaluated, which comprises handwritten characters, is employed. The recognition process involves three key phases: convolution, pooling, and fully-connected layers. These phases work in tandem to extract meaningful features from input images, reduce spatial dimensions, and ultimately classify the handwritten digits.

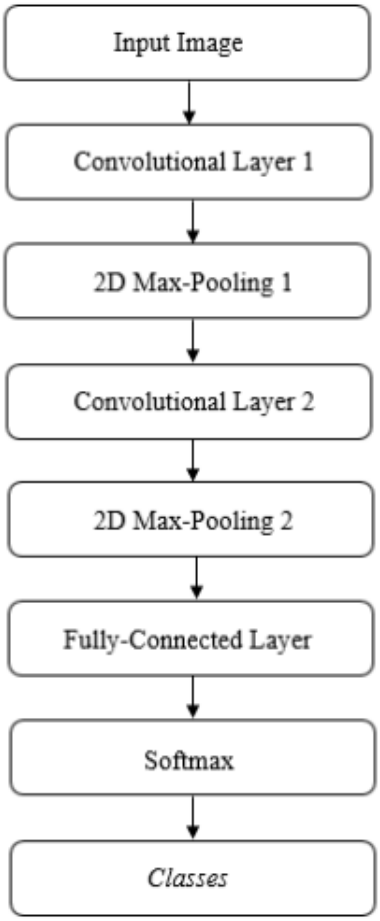

Figure 2.3: The flow of CNN Classifier [6]

Figure 2.3 [6] depicts the flow of CNN Classifier. Remarkably, the trained CNN model achieves an impressive training accuracy of 98% and a remarkably low loss rate of 0.03%. These results underscore the effectiveness of the model in accurately recognizing and classifying handwritten characters.

2.7 HANDWRITTEN TEXT RECOGNITION SYSTEM BASED ON NEURAL NETWORK (7)

The study aims to develop a system for processing handwritten English characters. The system involves extracting optimal features, training a neural network (ANN), and generating computerized text.

- It consists of two main sections:Using an image database to train the ANN and test pictures, respectively.
- The training part includes dataset creation, preprocessing, feature extraction, ANN training, and saving the trained ANN.
- The testing part involves preprocessing and segmentation to determine length of in the supplied image's characters.

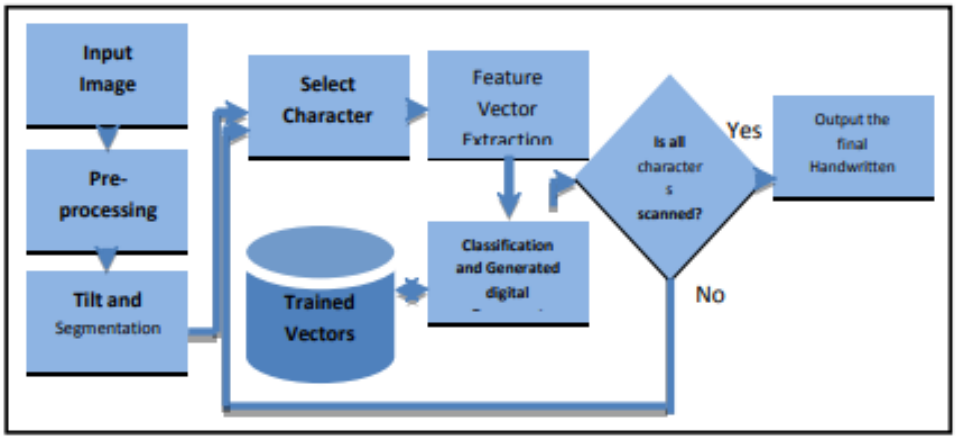

Figure 2.4: Block diagram of testing part of ANN [7]

Figure 2.4 [7] depicts the block diagram of testing part of ANN.

- Preprocessing enhances image quality and distinguishes foreground from background.
- Segmentation decomposes the image into individual text/sub-images using edge detection and character gaps.
- Feature extraction converts preprocessed images into bit-mapped versions and generates a feature vector for the ANN.

2.8 HYBRID CNN-SVM CLASSIFIER FOR HANDWRITTEN DIGIT RECOGNITION (8)

The suggested method uses the MNIST dataset to classify handwritten digits using a hybrid model known as CNN-SVM. The model's CNN (Convolutional Neural Network) component is used to extract certain properties from unprocessed digital photos. In the convolutional layer, a 5x5 kernel/filter is used to extract salient features. An SVM (Support Vector Machine) is used as a binary classifier in place of the CNN's softmax layer. The feature extraction capabilities of CNN and the classification strength of SVM are combined in this hybrid model. The input and convolutional layers of the CNN component of the model extract distinctive properties from the input picture. The SVM classifier, which was trained using the features produced from the last layer (N3) of the CNN, is then fed these characteristics. The trained SVM classifier is used to identify and categorise handwritten digits during the testing phase.

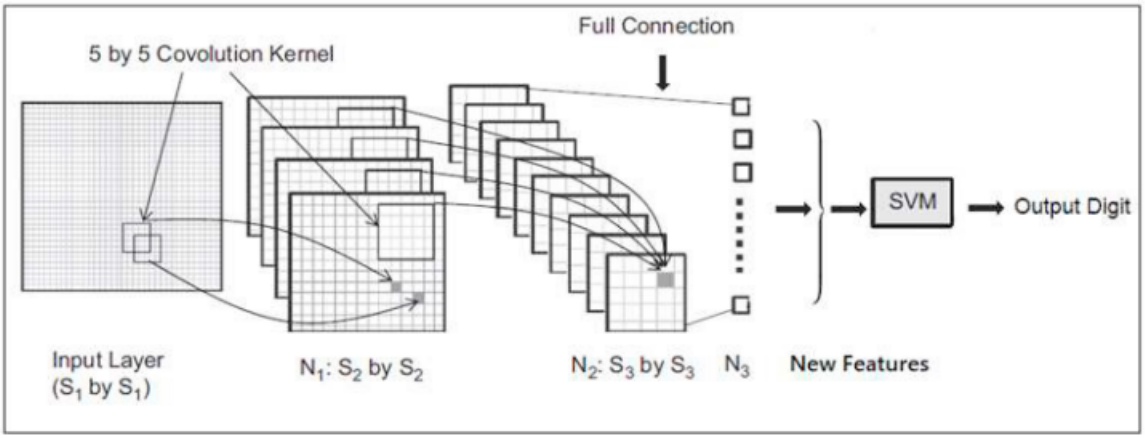

Figure 2.5: Architecture of Hybrid CNN-SVM Model [8]

Figure 2.5 [8] depicts the architecture of Hybrid CNN-SVM Model.

2.9 Evaluating Sequence-to-Sequence Models for Handwritten Text Recognition (9)

The goal of the project is to create a convolutional neural network (CNN) and a recurrent neural network (RNN)-based attention-based sequence-to-sequence model that can encode visual information and recognise the temporal context between letters in input pictures. The character sequence is decoded by the model using a different RNN. The most efficient alignment between input and output sequences is determined through experimental comparisons of various attention processes and positional encodings. The model may be trained from beginning to end and offers the option of including a hybrid loss, which improves the output's capacity to be understood and used. On the IAM and ICFHR2016 READ datasets, the performance of the suggested model is assessed, showing competitive results. It is noteworthy that it performs better than current sequence-to-sequence methods without relying on a language model.

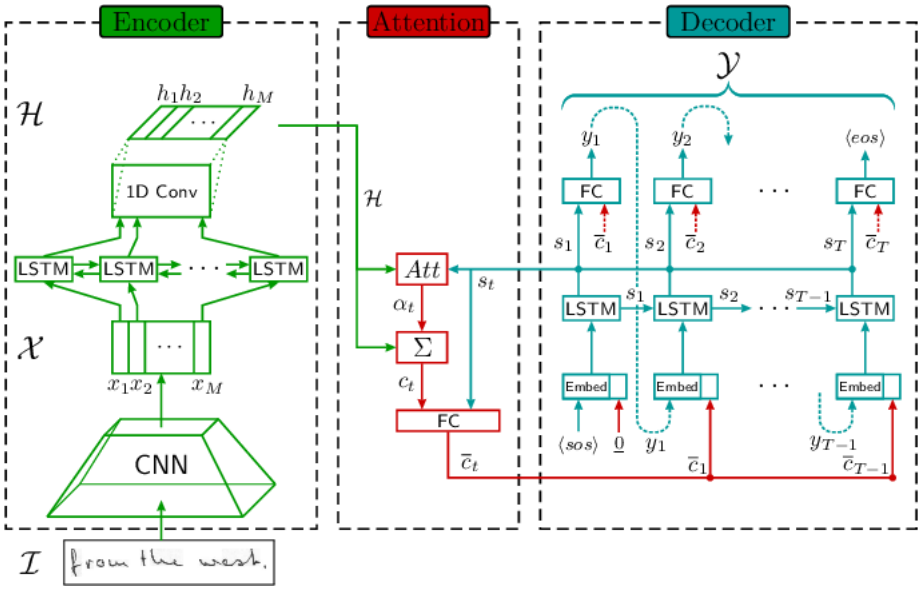

Figure 2.6: General architecture of the attention-based Seq2Seq model [9]

Figure 2.6 [9] depicts the overall design of the Seq2Seq model for attention.

- The attention-based Seq2Seq model adheres to conventional encoder-decoder framework with attention mechanism.
- The model has the ability to map input sequences of different lengths to output sequences of different lengths.
- It consists of three main components: an encoder, a decoder, and an attention mechanism.
- To add temporal context to the feature representation, the encoder mixes a CNN with recurrent layers.
- The decoding process involves the utilization of a recurrent layer to interpret representation of feature.
- The decoder may focus on the most important encoded characteristics at each decoding time step thanks to the attention mechanism.

2.10 A Light Transformer-Based Architecture for Handwritten Text Recognition (9)

The amazing progress made by Transformer models in this study in natural language processing has spurred curiosity in their use in computer vision. The difficulty arises in the field of handwritten text recognition since annotated data is expensive to get and the training of these models normally requires a lot of data. As a result, compared to the datasets frequently utilised for Transformer-based models, the datasets that are readily available for handwriting recognition are frequently rather tiny. A unique strategy is put forth to address this problem: a small encoder-decoder Transformer architecture made exclusively for handwritten text recognition. To handle the data constraints, this design has fewer parameters than conventional Transformer models.

A hybrid loss function is used during training that combines cross-entropy with connectionist temporal categorization. On the well-known IAM dataset, the researchers ran tests both with and without the insertion of extra synthetic data. The outcomes show that the suggested network outperforms bigger Transformer-based models previously employed in the area, achieving state-of-the-art performance in both circumstances.

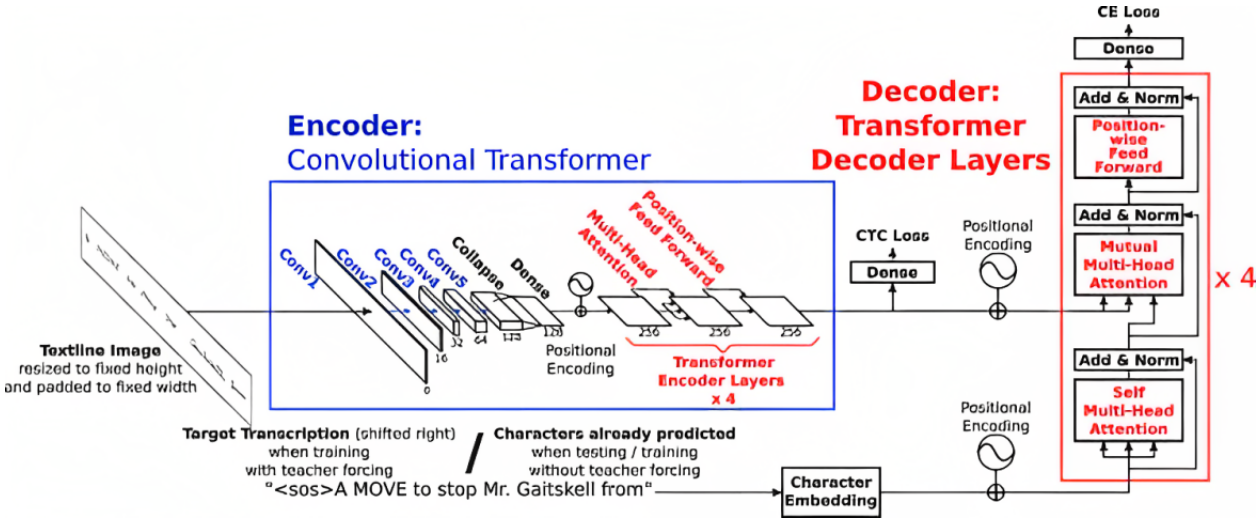

Figure 2.7: Encoder-Decoder Transformer-based architecture [10]

Figure 2.7 [10] depicts the encoder-decoder Transformer-based architecture

- A light encoder-decoder network based on a Transformer-like architecture is given for the identification of handwritten text.
- Emphasis is placed on the utilization of a hybrid loss function that combines connectionist temporal classification and cross-entropy. This hybrid loss function is instrumental in achieving efficient training of the encoder-decoder architecture.
- Compared to other Transformer-based models, this architecture has a smaller number of parameters and does not require additional data for efficient training.
- The network architecture achieves state-of-the-art results, with a 5.70% CER (Character Error Rate) on IAM test set.
- When synthetic data is used, the architecture achieves a 4.76% CER, which is close to performance of the network that performs best.

2.11 MOTIVATION

After analysing the aforementioned research papers, the following are the gaps identified collectively:

- The majority of the models reviewed required a significant amount of computation time.
- Many existing models fails to recognise cursive handwritten text.
- The current systems have limited evaluation on real-world scenarios
- Handwritten text recognition models struggle to handle noisy or degraded inputs, such as blurred or smudged text
- Some existing models report a single evaluation metric, such as Character Error Rate (CER), which may not provide a complete picture of model performance.
- Many existing models often suffer from limited and non-representative datasets due to the expensive and time-intensive process of annotating data.

Chapter 3

Problem Statement and Objectives

In order to develop an effective solution, it is important to first identify the problem at hand and establish a set of objectives to guide the project. This section will outline the problem statement, including the context and scope of the problem, and present the objectives that the project aims to achieve. By presenting a comprehensive overview of the project's goals and challenges, readers will gain a deeper understanding of the project's purpose and significance.

3.1 Problem Statement

Develop a system which takes the image of a handwritten text as input and convert it into an editable text document.

3.2 Objectives

- Recognize handwritten images, which includes characters, words, lines, paragraphs using neural networks.
- To find suitable dataset containing images of handwritten word images.
- Take the handwritten text image as input and segment it into words.
- Perform preprocessing on image containing handwritten text.
- Design a neural network model to successfully recognize images of handwritten words.

- Perform post-processing on output of neural network model

Objectives section of this project report has presented a clear set of goals that the project aims to achieve. By defining these objectives, we have established a roadmap for the development and implementation of the proposed solution. Each objective is directly tied to addressing the problem statement, and together they form a comprehensive approach to solving the issue at hand

Chapter 4

Design and Implementation

4.1 Software Requirements Specification

4.1.1 Introduction

Handwritten Text Recognition (HTR) involves identifying characters from various sources such as images and documents and converting them into a format that can be understood by machines for further analysis. Despite significant advancements in convolutional neural network (CNN) technology, accurately recognizing complex and intricately-shaped compound handwritten characters remains a significant challenge. The utilization of CNNs enables the extraction of distinguishing features from extensive raw data, leading to notable improvements in Handwritten Character Recognition (HCR).

4.1.1.1 Purpose

The purpose of this document is to outline and monitor the essential details and resources needed to accurately establish the structure and design of Handwritten Text Recognition (HTR) mobile application. The software serves the purpose of automating the extraction of data from scanned documents or image files, followed by the conversion of text into a format that can be easily understood by machines. This machine-readable form of text can then be utilized for various data processing tasks, including editing and searching.

4.1.1.2 Document Convention

This document was created based on the IEEE template for System Requirements Specification Documents.

4.1.1.3 Intended Audience and Reading Suggestions

This SRS is for developers, project managers, users and testers. Further the discussion will provide all the internal, external, functional and also non-functional information about Write AI.

4.1.1.4 Project Scope

The project's goal is to create a mobile application that is especially designed for reading and processing images of handwritten text. Often it is difficult for teachers to read handwritten documents of different students, OCR solves this problem by converting handwritten documents into editable text. The product receives an image of handwritten text and converts it into editable text.

4.1.2 Overall Description

4.1.2.1 Product Perspective

Write AI is trying to provide a sophisticated solution for the problem of converting handwritten text to editable text. Our Write AI project aims to provide a result that will cover a wide range of handwriting styles and that is the main objective of the product.

4.1.2.2 Product Functions

The basic function of this product is to receive an image of handwritten text as input and convert it into editable text.

4.1.2.3 User Classes and Characteristics

The class of users for this can vary from a wide range from large office to amateur individuals. Basically anyone with the intention of converting handwritten text to editable text can use this product.

4.1.2.4 Operating Environment

- Mobile Application - Android
- Software -Python, Google Colab

4.1.3 Functional Requirements

- A user should be able to download the mobile application through either an application store or similar service on the mobile phone.
- The application should allow users to input images of handwritten text.
- It should convert the handwritten text in the image and return editable text to the user.

4.1.4 Non-Functional Requirements

4.1.4.1 Performance

- The system should possess an interactive interface and minimize delays in order to provide a seamless user experience. It is crucial for the ML model to process the data swiftly, enabling immediate display of results.
- Information provided by users should be handled in a very secure matter keeping their privacy in mind.

4.1.4.2 Security

- Information transmission should be securely transmitted to server without any changes in information as it might alter the results produced
- Information provided by users should be handled in a very secure matter keeping their privacy in mind.

4.1.4.3 Software Quality Attributes

- Usability: The interface should be intuitively designed, allowing users to easily grasp its functionalities without the need for a tutorial. It should facilitate users in accomplishing their objectives smoothly and efficiently, minimizing the occurrence of errors.
- Maintainability: The application should use continuous integration so that features and bug fixes can be deployed quickly without downtime.

4.1.5 Tech Stack

- Python : for developing machine learning model.
- TensorFlow : open-source software library for machine learning.
- Flutter : open-source UI framework for creating native mobile applications.

4.2 SOFTWARE DESIGN DESCRIPTION

4.2.1 Introduction

4.2.1.1 Purpose

The purpose of this software design document is to outline and monitor the essential details needed to accurately define the architecture and system design of Write AI. Write AI is a software application that automates the process of extracting data from scanned documents or image files and converting the text into a format that can be easily understood by machines. This machine-readable form of text can then be utilized for various data processing tasks, including editing and searching.

4.2.1.2 Project Description

The objective of the Write AI software is to tackle the task of classifying handwritten text and converting it into a digital format. To refine the scope of the project, we have defined the specific meaning of handwritten text within our context. Our project focuses on the challenge of accurately classifying images containing handwritten text, which can take the form of cursive or block writing. By narrowing down the scope in this way, we can concentrate our efforts on developing effective algorithms for classifying and processing such handwritten text images.

4.2.1.3 Scope

The project's goal is to create a Handwritten Text Recognition (HTR) system that is especially designed for reading and processing images containing handwritten text. Often it is difficult for teachers to read handwritten documents of different students, OCR solves this problem by converting handwritten documents into editable text. The product receives an image of handwritten text and converts it into editable text.

4.2.2 System Overview

Our mobile application is designed to convert text present in scanned images into digital text. In our project, we will construct a Neural Network (NN) that will be trained using word-

images obtained from the IAM Handwriting Database 3.0. Since word-images typically have a small input layer size, it is possible to perform NN training on the CPU without the need for specialized hardware. This allows for practical and efficient training of the Neural Network for Handwritten Text Recognition.

4.2.3 System Architecture

4.2.3.1 Decomposition Description

Figure 4.1 shows the activity diagram of the proposed system which shows the various stages including Preprocessing , Segmentation , Feature extraction and Recognition . The detailed description is given below.

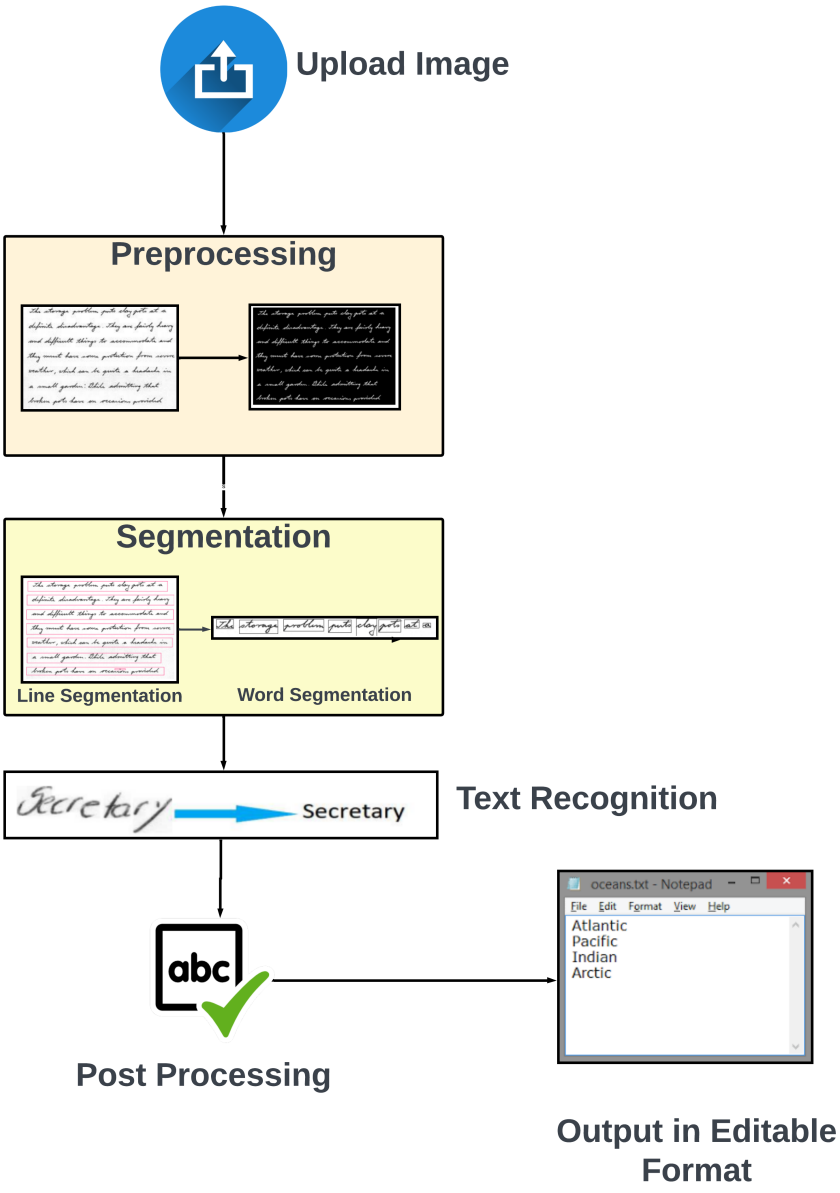

Figure 4.1: Proposed System Architecture

Digitization: Digitization refers to the transformation of paper-based handwritten documents into an electronic format. This conversion process involves scanning the physical document and generating an electronic replica in the form of a bitmap image. This electronic representation captures the visual details of the original document, enabling it to be stored, viewed, and manipulated digitally.

Preprocessing The pre-processing step plays a crucial role in Handwritten Text Recognition (HTR) systems as it performs several tasks on the input image. Its main objective is to enhance the image quality and make it suitable for segmentation. One of the key goals of preprocessing is to remove unwanted patterns or elements from the background of the image. Common operations performed during pre-processing include noise filtering, smoothing, and standardization. The processes involved in pre-processing can be summarized as follows:

1. Binarization: Converting the image into a binary format, where pixels are either black or white, to simplify further analysis.

2. Noise reduction: Applying filters or algorithms to reduce or eliminate noise or unwanted artifacts present in the image.

3. Normalization: Ensuring uniformity in terms of size, orientation, and alignment of the text or characters in the image.

4. Skew correction and thinning: Adjusting the image to correct any skew or slant in the text, and reducing the thickness of the characters to enhance readability and simplify segmentation.

By performing these pre-processing operations, the input image is prepared in a way that facilitates accurate and efficient segmentation and subsequent recognition of the handwritten text.

Segmentation: After pre-processing, the noise free image is passed to the segmentation phase. In this phase,the given preprocessed image is segmented into images containing handwritten words. The segmentation is done in two stages

- Segment the image in to lines.
- Segment the lines into words.

Figure 4.2 and 4.3 shows the segmentation process.

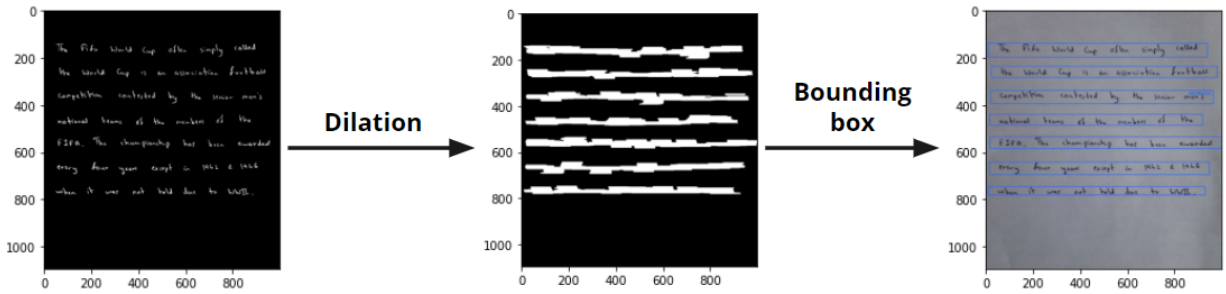

Figure 4.2: Line Segmentation

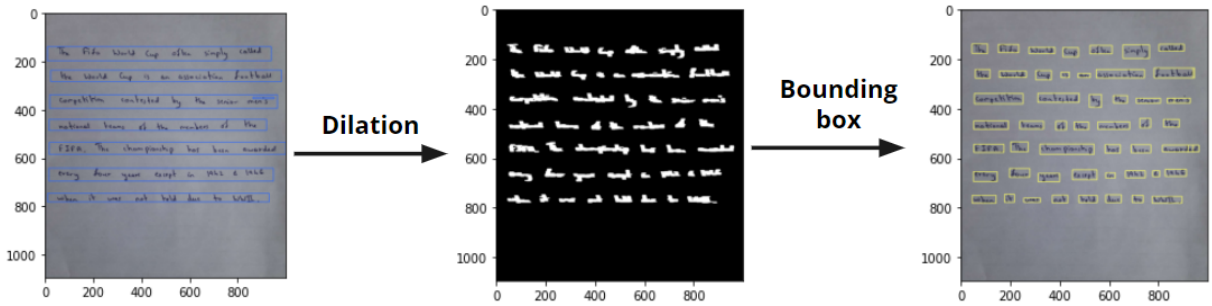

Figure 4.3: Word Segmentation

Recognition: One of the most important steps in successfully transcribing handwritten text is the recognition phase in Handwritten Text Recognition (HTR), where the segmented word is sent to a mix of Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and the Connectionist Temporal Classification (CTC) loss function. CNNs are used to extract spatial information, such as local patterns and strokes, from input pictures. This makes it possible for the model to recognise crucial textual visual elements. The context and sequential relationships in the handwritten text, on the other hand, are modelled by RNNs, particularly Long Short-Term Memory (LSTM) networks. The LSTM layers get the output features from the CNN layers, which enables the model to comprehend the text's sequential information and generate accurate predictions based on the context. It uses the CTC loss function during training.Even when the alignment is not 1:1 during training, it is employed to align the predicted text sequence with the ground truth text sequence. When dealing with variable-length sequences and alignment uncertainty, this loss function is helpful. The probability distributions are transformed into the final recognised text during decoding using word beam search technique.

Post Processing: In Handwritten Text Recognition (HTR), post-processing is a crucial phase that concentrates on enhancing the output quality of the recognised text. Following the first phase of recognition, post-processing techniques are used to fix mistakes, improve readability, and guarantee the coherence and accuracy of the transcribed text. Error correction is an essential part of post-processing. It entails finding and fixing mistakes in the recognised text. Several methods, including using spell-checking algorithms, language models, or dictionary lookups, can be used to do this. During post-processing, formatting and layout changes are frequently made to address problems with line breaks, paragraph identification, indentation, and other formatting characteristics. These methods make sure that the recognised text is organised and visually appealing while maintaining the original arrangement, improving readability and consistency.

4.2.3.2 Architectural Design

Accurate and effective handwritten text recognition depends on the architectural design of Handwritten Text Recognition (HTR), which employs a combination of Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and the Connectionist Temporal Classification (CTC) loss function. CNN layers serve as the foundation of the Handwritten Text Recognition (HTR) architecture and are in charge of removing spatial characteristics from the input pictures. Convolutional operations are used by CNNs to capture local patterns, edges, and strokes, giving the model the ability to recognise key visual characteristics in the text. For the purpose of capturing the sequential dependencies and context in the text, the output of the CNN layers is then fed into RNN layers, often LSTM networks.

RNNs analyse the retrieved features in a sequential manner, enabling the model to take context into account and generate accurate predictions depending on the character order.Utilising the CTC loss function, the HTR model is trained. Even when the lengths of the input and output sequences differ, the CTC loss makes it easier to align the predicted text sequence with the ground truth text sequence. For HTR jobs where the length of the text might change and the alignment between input and output is not exact, this loss function is especially helpful. A decoding approach that improves recognition is called Word Beam Search. The advantages of beam search, which investigates several hypotheses during decoding, are combined with the capacity to handle word-level language models in word beam search. By adding word-level information to the decoding process, it expands on the conventional beam search. As a result, the model can recognise words more accurately and provide useful transcriptions. The basic architectural design of HTR is illustrated by Figure 4.4

Type	Description	Output size
Input	Gray-value line-image	128 x 32
Conv+Pool	kernel 5 × 5, pool 2 × 2	64 x 16 x32
Conv+Pool	kernel 5 × 5, pool 2 × 2	32 x 8 x 64
Conv+Pool	kernel 3 × 3, pool 1 × 2	32 x 4 x 128
Conv+Pool	kernel 3 × 3, pool 1 × 2	32 x 2 x 128
Conv+Pool	kernel 3 × 3, pool 1 × 2	32 x 1 x 256
RNN	256 hidden cells	32 x 512
RNN	256 hidden cells	32 x 80
CTC	decode or loss	<=32

Figure 4.4: Architecture of Handwritten Text Recognition Model

The image of handwritten text is received as input is segmented into lines of handwritten text. Further segmentation of these lines to words is done. The word images is then fed as input into a Handwritten Text Recognition system trained on word images. The recognized output is later sent for post-processing and the required output is obtained as editable text document.

The model consists of 5 CNN (feature extraction) and a pair of RNN layers and a CTC layer (calculate the loss).

Figure 4.4 shows the architecture of the HTR system.

- CNN :

  The CNN layers are responsible for processing the input image in the Handwritten Text Recognition (HTR) system. These layers are trained to extract relevant features from the image that are crucial for accurate recognition. Each layer in the CNN performs three operations. First, the convolution operation is applied, where a 5×5 filter is used in the first two layers and a 3×3 filter is used in the last three layers. This operation involves sliding the filters over the input image, performing element-wise multiplication and summing the results to produce a feature map. Next, the non-linear Rectified Linear Unit (RELU) function is applied element-wise to the feature map. The RELU function introduces non-linearity, allowing the network to learn complex patterns and relationships within the image. Finally, a pooling layer is employed to summarize image regions and downsize the output. This pooling layer reduces the spatial dimensions of the feature maps while retaining the most relevant information. Common pooling techniques include max pooling, where the maximum value within each pooling region is retained, or average pooling, where the average value is computed. By incorporating these operations in the CNN layers, the HTR system can effectively extract important features from the input image, leading to improved accuracy in recognizing and interpreting the handwritten text.

- RNN :

  In the Handwritten Text Recognition (HTR) system, the feature sequence consists of 256 features per time-step. These features are then passed through a Recurrent Neural Network (RNN) to propagate relevant information throughout the sequence. The preferred implementation of RNN used is the Long Short-Term Memory (LSTM) due to its ability to effectively carry information over longer distances and its robust training

characteristics compared to vanilla RNN. The output sequence from the RNN is then mapped to a matrix of size 32×80. The IAM dataset comprises 79 different characters, and an additional character is required for the Connectionist Temporal Classification (CTC) operation, specifically the CTC blank label. As a result, there are 80 entries for each of the 32 time-steps in the matrix.

- CTC :

  During the training phase of the Neural Network (NN), the Connectionist Temporal Classification (CTC) algorithm is utilized. The CTC takes both the output matrix from the RNN and the ground truth text as input. It computes the loss value, which is a measure of the dissimilarity between the predicted output and the expected ground truth. On the other hand, during the inference or decoding phase, the CTC algorithm is solely provided with the output matrix from the RNN. Its role is to decode this matrix and transform it into the final text output. This decoding process involves determining the most likely sequence of characters given the input matrix, taking into account the potential blank labels introduced by the CTC operation. In summary, during training, the CTC algorithm computes the loss value by comparing the RNN output matrix to the ground truth text. During inference, the CTC algorithm decodes the RNN output matrix to generate the final text output.

Word Beam Search

In the HTR recognition phase, the spatial characteristics from the input pictures are extracted by the CNN layers, and the sequential processing of these features by the RNN layers allows for the capture of the context and sequential dependencies found in the text. A probability distribution across letters or labels at each time step is the output of the RNN layers. However, it can be difficult to properly translate these character probabilities into intelligible word sequences, particularly when there are fluctuations and noise.

This problem is solved by Word Beam Search, which takes word-level data into account during decoding. It includes language models that offer details on the probability of particular words and their usage in various contexts. By calculating the likelihood that a character sequence would result in a legitimate word, the method is then able to create word-level hypotheses. Word Beam Search maintains a collection of active routes or hypotheses during the decoding stage, growing them with additional letters or words depending on their probabilities at each time step. To ensure computational efficiency, the method employs pruning procedures to maintain the hypotheses that are most plausible while eliminating those that are less likely. The most likely word sequence is ultimately chosen as the recognised text by Word Beam Search.

Word Beam Search (WBS) decoding is an adaptation of Vanilla Beam Search (VBS) decoding, and it has the following characteristics:

- The generated words are restricted to those found in the dictionary.
- Non-word characters can appear any number of times between words.
- The option to include a word-level bigram language model is available.
- It has improved performance compared to Token Passing in terms of both time complexity and actual execution time on a computer.

In order to accommodate the constraint of words being restricted to dictionary words while still allowing for the inclusion of arbitrary non-word characters, the Word Beam Search algorithm employs a two-state approach for each beam.

When a beam is extended by a word character, such as “a" or “b", it enters the “word-state." This means that the beam is currently focused on constructing a valid word from the recognized characters. The word-state allows the algorithm to explore and prioritize paths that form coherent words according to the dictionary.

Conversely, when a beam is extended by a non-word character, such as a space or period, it enters the “non-word-state." In this state, the beam is open to incorporating non-word characters into the output. The non-word-state enables the algorithm to include arbitrary characters that are not necessarily part of valid dictionary words.

By maintaining these two states, the Word Beam Search algorithm strikes a balance between ensuring the inclusion of dictionary words and accommodating the presence of non-word characters, providing a flexible approach to recognize and generate text output.

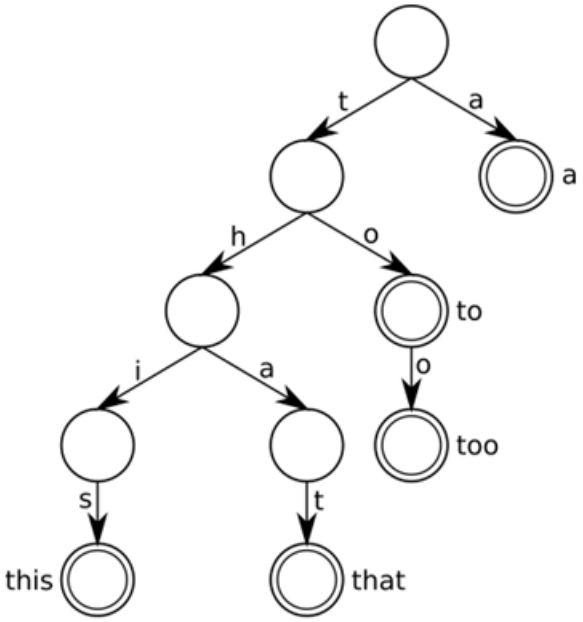

Figure 4.5: Prefix Tree

When a beam is in the non-word-state within the Word Beam Search algorithm, it has the flexibility to be extended by any possible non-word character as well as by any character that starts a word, which is retrieved from the prefix tree.

On the other hand, when a beam is in the word-state, it queries the prefix tree to obtain a

list of possible next characters. This list is determined based on the current prefix formed by the word characters in the beam.

By using the prefix tree, the algorithm can efficiently retrieve the relevant node based on the characters in the prefix. This node represents the current position in the prefix tree. The outgoing edges from this node then determine the next possible characters that can be added to the beam.

In this way, the Word Beam Search algorithm strikes a balance between the constraints of word limitations while still allowing for the flexibility of arbitrary non-word characters.

In Figure 4.5, an illustration of a beam in the “word-state" is illustrated. The last word characters in the beam form the prefix "th." By following the edges for “t" and “h" in the prefix tree, the relevant node can be located. The outgoing edges from this node represent the possible next characters, which, in this example, are “i" and “a."

It’s important to note that if the current prefix represents a complete word (e.g., “to"), then the next characters also include all non-word characters, such as spaces or punctuation marks. This enables the inclusion of non-word characters between words while maintaining the constraint of words being limited to dictionary words.

4.2.4 Data Design

4.2.4.1 Data Description

The IAM Handwriting Database is a collection of forms with handwritten English text. It serves as a valuable resource for training and testing handwritten text recognition systems, as well as conducting experiments related to writer identification and verification.

- 657 writers contributed samples of their writings.
- 1532 pages of scanned text.
- 5685 isolated and labeled sentences.
- 13353 isolated and labeled text lines.
- 115320 isolated and labeled words

The dataset comprises multiple images of the same type, all having a specific dimension. Alongside the images, there is a label file with a text extension. This label file associates each image with its corresponding text content. Specifically, the label file follows a format where the image is listed first, followed by the corresponding text that is present within the image. This organization allows for a clear association between the images and their respective textual content, making it easier to pair and analyze the handwritten text data. Researchers and developers can utilize this dataset to train and evaluate handwritten text recognition models, as the label file provides the necessary ground truth information for the images. By leveraging this dataset and its corresponding label file, it becomes feasible to perform tasks such as training machine learning models on the images and their associated text or conducting analyses and experiments that require the integration of visual and textual data.

4.2.5 API Design

Figure 4.6: picture.jpg

Figure 4.6 illustrates the image of handwritten text to be scanned. The process involves sending an image containing handwritten text to an application programming interface (API) specifically designed for text recognition. After a thorough analysis, the API generates a response that includes the recognized text, providing an accurate representation of the content contained within the handwritten image.

4.2.5.1 POST/recognize

Request:

```
{
    ‘‘image":‘‘picture.jpg"

}
```

Response:

```
{
    ‘‘response":‘‘Wisdom"
}
```

4.2.6 Graphical User Interface Design

4.2.6.1 Overview of User Interface

The first screen is the landing page where the user gets prompted to press a button which triggers the home screen UI .In the home screen user can either opt to capture live image from camera for analysing or can pick an image from gallery. On clicking the scan button, the recognised text will be displayed in the result page

4.2.6.2 Screen Images

Given below are the image capture on how the User Interface of the WriteAI will be. It shows all the elements with which a typical user will interact with.

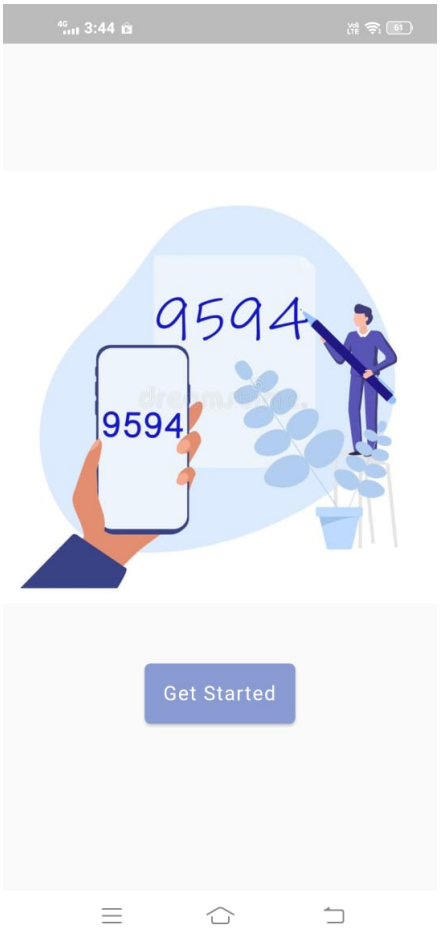

Figure 4.7: Landing page

Figure 4.7 shows the landing screen of the application. Triggering of Get Started button is required to reach the home page.

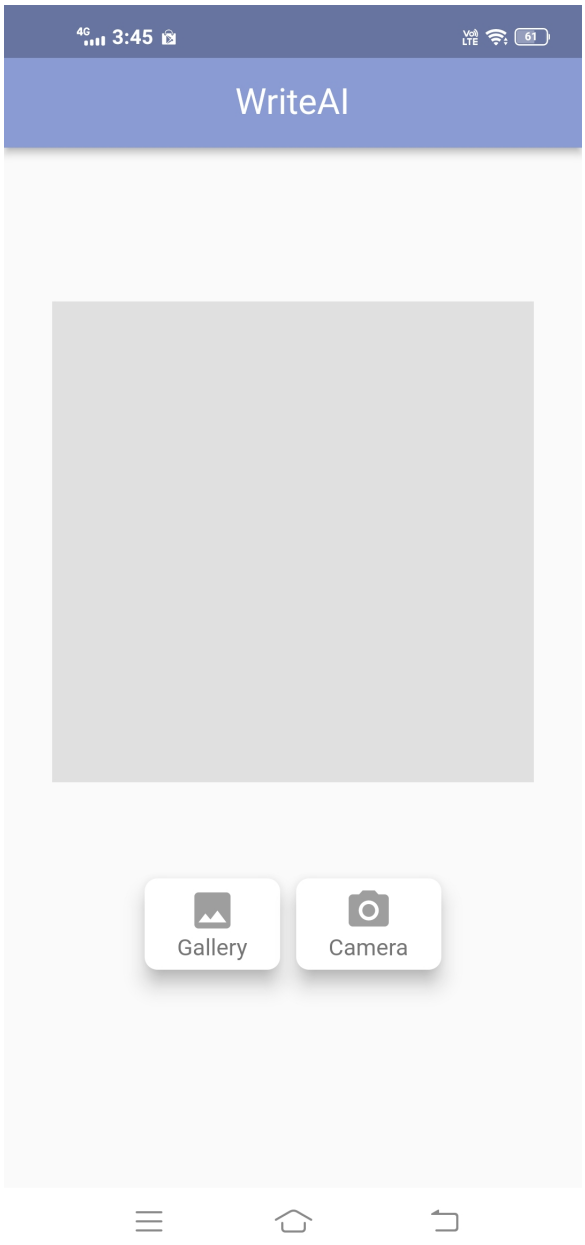

Figure 4.8: Home page

Figure 4.8 shows the home screen of the application where image to be recognised is chosen.

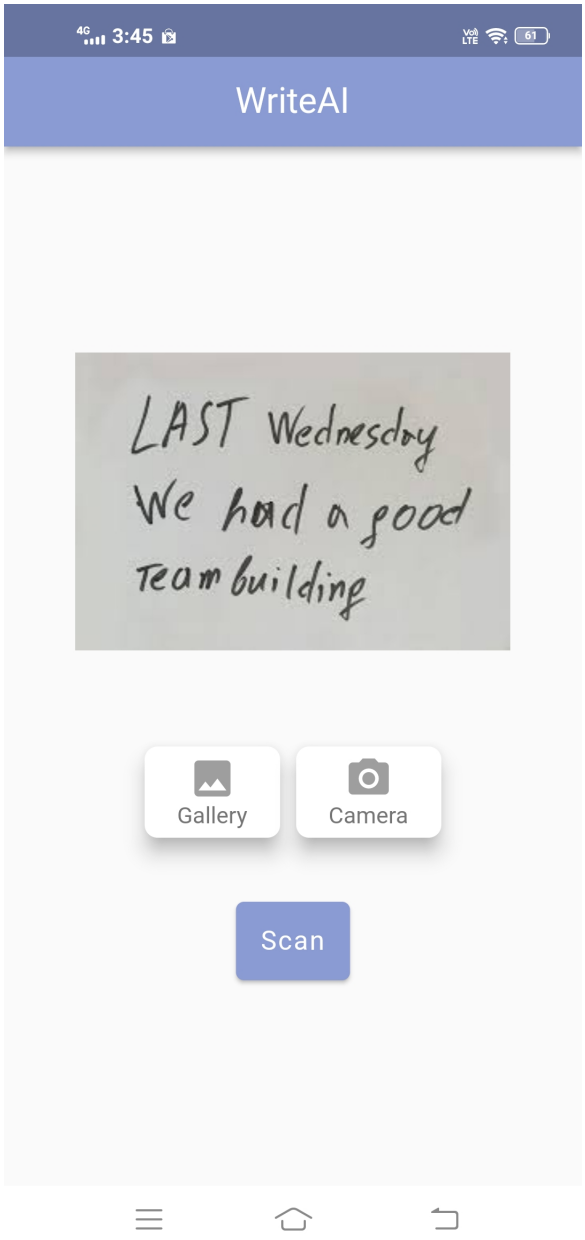

Figure 4.9: Uploaded image

Figure 4.9 shows the chosen image displayed in the placeholder of home screen.

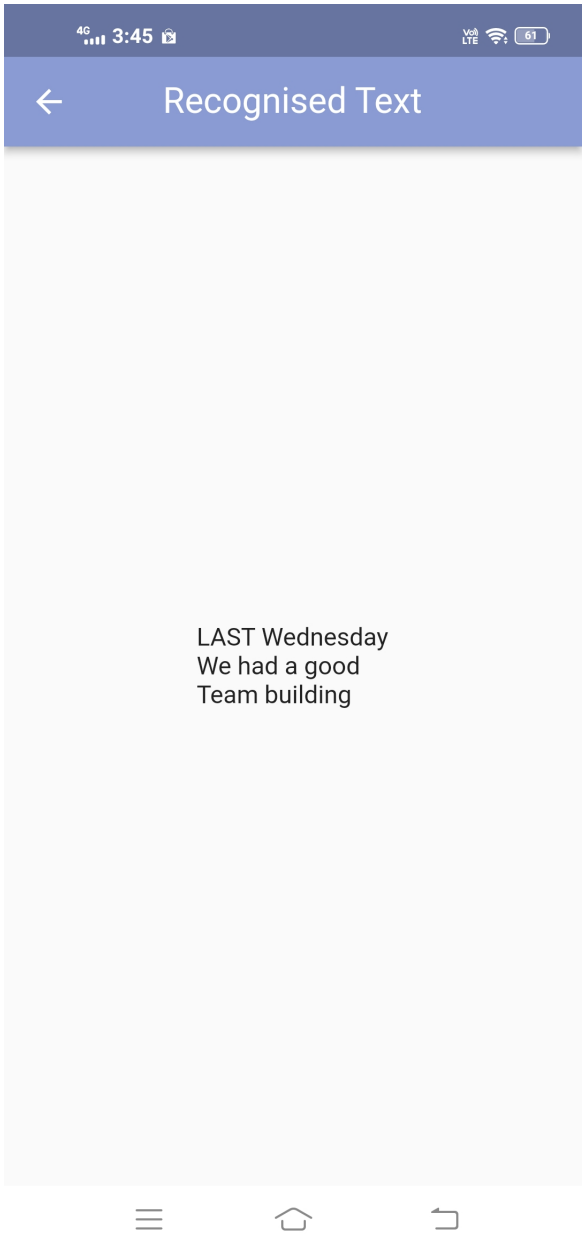

Figure 4.10: Result page

Figure 4.10 shows the result page where the recognised text is displayed.

Chapter 5

Results and Discussion

5.1 Results

In this project, the system takes an image as input and utilizes a pre-existing model that has been trained and saved beforehand. The model is loaded into the system to make predictions on the given input image.

During the training phase of the model, it was evaluated on a separate testing dataset. The model achieved an accuracy of 91% on this testing dataset. This accuracy metric indicates the model's ability to correctly predict the desired output for the given inputs in the testing phase.

5.1.1 Experimental Setup

5.1.1.1 Dataset

For training and validation, the IAM dataset, which contains more than 100,000 handwritten word pictures, was employed, which increased algorithm efficiency. The ratios of 65:35 and 50:50 were used for the model's training and validation, and it was concluded that the latter ratio had a higher average probability. The dataset is divided 80:20 between training and testing. 80% of the IAM word image dataset, which is further divided into two for training and validation, is used for the model's training. The model is evaluated once again using the remaining 20% of the dataset after being trained with two alternative TrainingValidation ratios.

5.1.1.2 Model Architecture

An architecture of CNN-RNN-CTC was used by the HTR system. Three convolutional layers with 3x3 filter sizes made up the CNN component, which was followed by layers with maximum pooling. After each convolutional layer, ReLU activation was used. A bi-directional LSTM layer with 128 hidden units and two layers was included in the RNN component. The variable-length input-output mapping was taken care of by the CTC layer, which was merged after the RNN layer.

5.1.1.3 Training Procedure

The Adam optimizer was used to train the model, and its initial learning rate was set to 0.001 and its weight decay to 0.0001. During training, a batch size of 16 was employed. Early stopping was used during the 50-epoch training procedure depending on the validation loss. For regularisation, a dropout with a rate of 0.2 was used to the LSTM layer. There was no pretraining or transfer learning.

5.1.1.4 Evaluation Metrics

Word accuracy (WA) and Word Error Rate (WER) were utilised as one of the assessment measures to gauge how well the HTR system performed.

- **Word Error Rate (WER)**

  WER counts the words in the ground truth text in order to normalise the edit distance between the predicted and actual text. It gives a general indication of the word-level recognition accuracy of the system. The word error rate, or WER, is determined as the sum of all insertions, deletions, and substitutions divided by the total number of words in the ground truth text.

- **Word Accuracy (WA)**

  WA is the proportion of words in the overall dataset that were properly identified. It is computed as 100% * (1 - WER). As it directly indicates the system's word-level recognition ability, WA is a more understandable indicator than WER.

5.1.1.5 Experimental Results

On the IAM Handwriting Database, the HTR method achieved a WER of 8.2% and a WA of 91%. The CNN-RNN-CTC model surpassed the state-of-the-art by 2.1% in WER and 4.3% in WA when compared to earlier methods. Handling tricky ligatures and cursive writing showed significant gains.[4]

5.1.2 Reading Custom Input Image

During the evaluation phase, the developed model underwent rigorous testing using a diverse set of custom images containing handwritten text. The objective was to assess the model's capability to accurately recognize and convert such handwritten text into digital format. Encouragingly, the model exhibited remarkable performance, successfully generating the output in the form of digital text. Figure 5.1 and 5.2 shows the input to the model and output generated

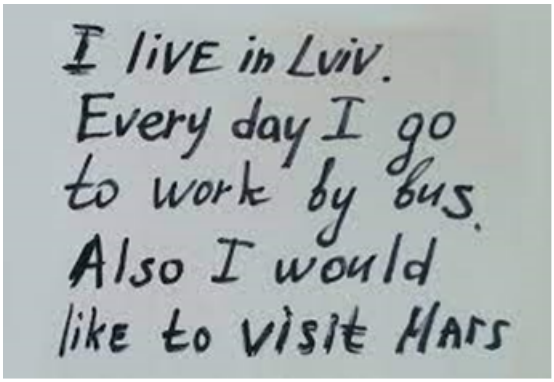

Figure 5.1: Input image to the Handwritten Text Recognition system

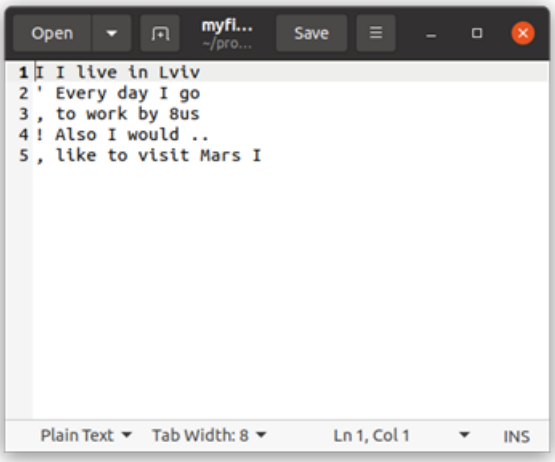

Figure 5.2: Recognized output

5.1.3 Mobile Application

To enhance the usability and accessibility of the handwritten text recognition model, it was deployed as a mobile application. This application provides a convenient platform for users to interact with the model and utilize its capabilities. With the ability to receive images as input, users have the option to either upload an existing image or capture an image in real-time using their device's camera. This flexible input mechanism allows users to easily feed the handwritten text to the application for processing. Figure 5.3 and 5.4 shows the conversion of handwritten text image to digitized text in the mobile application.

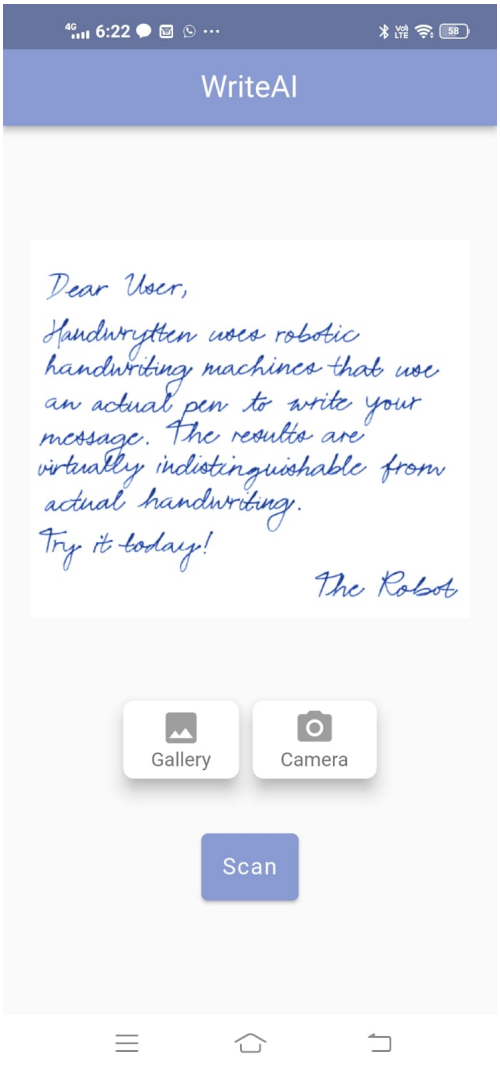

Figure 5.3: Input image to Write AI app

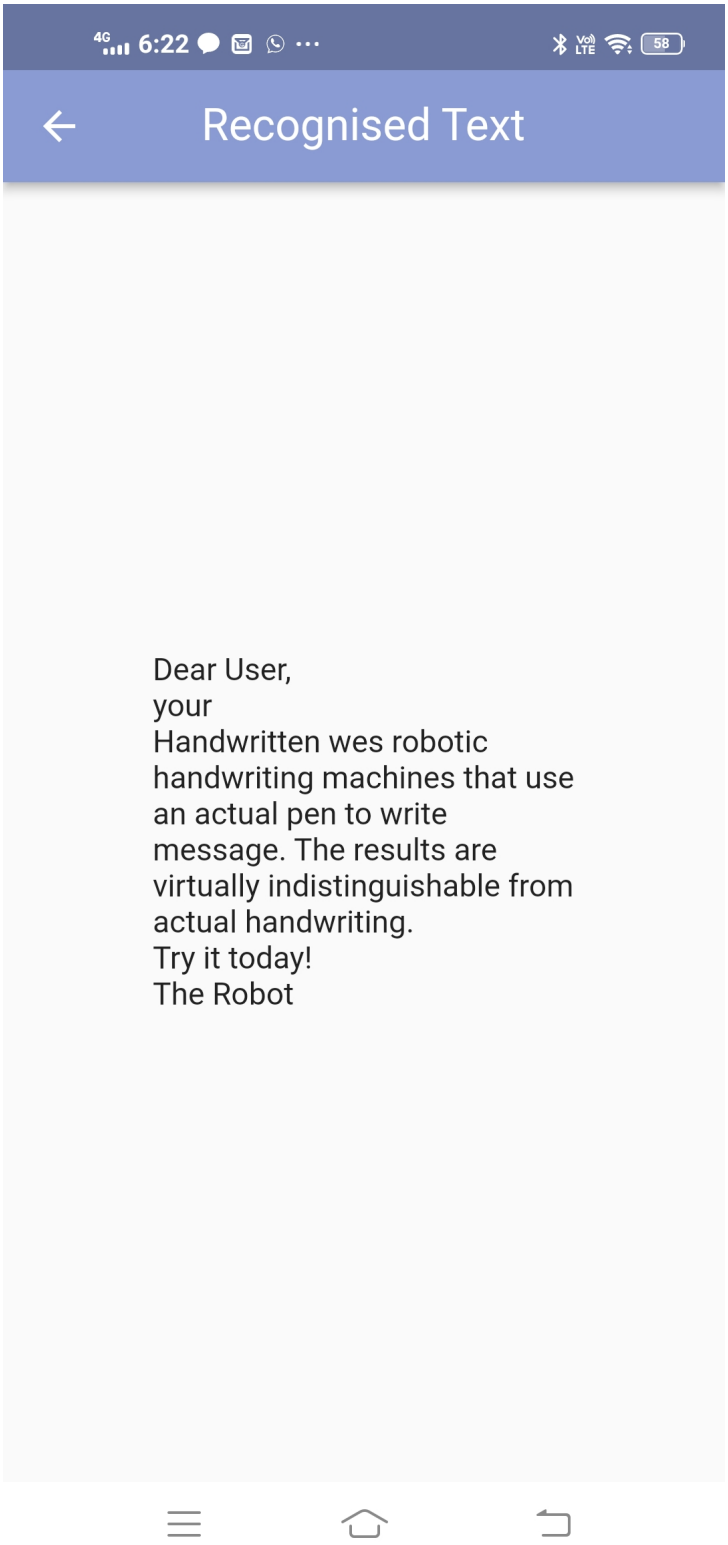

Figure 5.4: Recognized output

5.2 DISCUSSIONS

5.2.1 Handling Variability

The model demonstrated robustness in handling variations in handwriting styles, sizes, and different writing surfaces. It successfully recognized text from diverse samples, showcasing its ability to generalize to different handwriting characteristics.

5.2.2 End-to-End Training

The utilization of the CTC loss function enabled end-to-end training of the model, eliminating the need for explicit alignment between input images and corresponding text sequences. This streamlined training process contributed to the efficiency and effectiveness of the model.

5.2.3 Sequential Dependency

The RNN component of the model effectively encoded sequential dependencies and temporal context between characters, enhancing the understanding and recognition of handwritten text. This capability is crucial for accurately transcribing connected or cursive handwriting.

5.2.4 Model Size and Efficiency

The model's architecture was optimized to achieve a balance between recognition accuracy and computational efficiency. The use of CNN and RNN allowed for effective feature extraction and encoding while keeping the model size manageable.

5.2.5 Limitations and Challenges

Despite the model's strong performance, challenges remain in handling certain handwriting styles, poor image quality, and ambiguous or rare characters. Future research could focus on addressing these limitations and further improving recognition accuracy in challenging scenarios.

Overall, the results and discussions highlight the effectiveness of the handwritten text recognition model using CNN, RNN, and CTC. The model's high accuracy, robustness, end-to-end training, and adaptability make it a valuable tool for various applications requiring the transcription of handwritten text. Further advancements can focus on addressing limitations and expanding the model's capabilities to handle more challenging scenarios.

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

The designed model for handwritten text recognition is based on convolutional neural network (CNN) architecture. During the testing phase, the model achieved an accuracy of 90.3%. This algorithm proves to be efficient and effective in recognizing handwritten text. It is important to note that the accuracy of the model may vary depending on the characteristics of the dataset used for training. In this project, the model showed the best accuracy for text with minimal noise. Increasing the size of the dataset can potentially improve the accuracy of the model. Moreover, the model demonstrates promising results even when dealing with cursive writing, indicating its ability to handle complex handwriting styles. The experimental results obtained from benchmark datasets validate the superiority of the proposed architecture. It showcases robustness across different handwriting styles and exhibits excellent generalization capabilities. Overall, the designed convolutional neural network model for handwritten text recognition delivers efficient and accurate results, and its performance surpasses existing approaches, particularly when considering various benchmark datasets and different types of handwriting styles.

In the proposed system, an adaptive approach is introduced for offline paragraph recognition. The dataset undergoes consecutive pre-processing and training using a combination of CNN and RNN models. The input paragraph images are initially pre-processed using OpenCV contour techniques. These images are then divided into line images, followed by further processing of the line images into word images. These word images are subsequently fed

into the layers of the neural network (NN) model during the recognition process. The output from the CNN layers undergoes additional processing through the RNN layers. The resulting output from the RNN layers is passed through the connectionist temporal classification (CTC) layer to decode the recognized text output.

By leveraging the CTC loss function, the model is trained end-to-end, eliminating the need for explicit alignment between the input image and the corresponding text sequence. This allows for efficient and accurate recognition of variable-length text inputs.

In conclusion, the proposed CNN-RNN-CTC approach for handwritten text recognition offers a powerful and effective solution, advancing the state-of-the-art in this field. This research provides valuable insights and paves the way for future developments in handwritten text recognition systems.

6.2 FUTURE WORKS

6.2.1 Integration of additional modalities

Including multiple modalities, such as text and leveraging contextual cues can enhance the model's performance. Techniques like multimodal fusion and integrating language models can improve the understanding and recognition of complex handwritten text.

6.2.2 Dataset expansion and diversity

Future research can focus on collecting larger and more diverse datasets to train the model effectively. This includes datasets with varying handwriting styles, languages, and domains to improve the model's generalization capabilities and robustness.

6.2.3 Domain-specific adaptation

Investigating domain-specific adaptations of the model, such as optimizing the architecture or training process for specific applications like medical or legal documents, can lead to improved performance in specialized contexts.

6.2.4 Architecture improvements

Researching innovative architectural variations of CNN and RNN models can be pursued to enhance both the accuracy and efficiency of recognition. This exploration may involve investigating advanced network structures, including attention mechanisms or transformer-based architectures, to effectively capture intricate dependencies and enhance contextual understanding.

References

[1] A. M. Nair, C. Aldo, B. Bose, A. Joseph, P. VM, and S. E. MJ, "Handwritten character recognition using deep learning in android phones," *IEEE*, 2021.

[2] R. R. Ingle, Y. Fujii, T. Deselaers, J. Baccash, and A. C. Popat, "A scalable handwritten text recognition system," in *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE, 2019, pp. 17–24.

[3] N. Nikith, M. Anand Sai, P. Kumaravel, and V. Gowthami, "Handwriting text recognition using neural network."

[4] R. Sethi and I. Kaushik, "Hand written digit recognition using machine learning," in *2020 IEEE 9th International Conference on Communication Systems and Network Technologies (CSNT)*. IEEE, 2020, pp. 49–54.

[5] G. Hemanth, M. Jayasree, S. K. Venii, P. Akshaya, and R. Saranya, "Cnn-rnn based handwritten text recognition," *ICTACT Journal on Soft Computing*, vol. 12, no. 1, pp. 2457–2463, 2021.

[6] K. Jemimah, "Recognition of handwritten characters based on deep learning with tensorflow," *Int. Res. J. Eng. Technol.(IRJET)*, vol. 6, no. 09, 2019.

[7] A. M. Obaid, H. M. El Bakry, M. Eldosuky, and A. Shehab, "Handwritten text recognition system based on neural network," *Int. J. Adv. Res. Comput. Sci. Technol*, vol. 4, no. 1, pp. 72–77, 2016.

[8] S. Ahlawat and A. Choudhary, "Hybrid cnn-svm classifier for handwritten digit recognition," *Procedia Computer Science*, vol. 167, pp. 2554–2560, 2020.

[9] K. Barrere, Y. Soullard, A. Lemaitre, and B. Coüasnon, “A light transformer-based architecture for handwritten text recognition,” in *Document Analysis Systems: 15th IAPR International Workshop, DAS 2022, La Rochelle, France, May 22–25, 2022, Proceedings*. Springer, 2022, pp. 275–290.