

Automated Handwritten Answer Sheet Evaluation System Using Deep Learning Methods

by

Khushali Ratanghayara
202111049

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of

MASTER OF TECHNOLOGY
in
INFORMATION AND COMMUNICATION TECHNOLOGY
to

DHIRUBHAI AMBANI INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGY

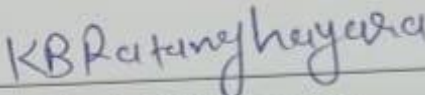


May, 2023

Declaration

I hereby declare that

- i) the thesis comprises of my original work towards the degree of Master of Technology in Information and Communication Technology at Dhirubhai Ambani Institute of Information and Communication Technology and has not been submitted elsewhere for a degree,
- ii) due acknowledgment has been made in the text to all the reference material used.


Khushali Ratanghayara

Certificate

This is to certify that the thesis work entitled Handwritten Answersheet Evaluation Using Deep Learning has been carried out by Khushali Ratanghayara for the degree of Master of Technology in Information and Communication Technology at *Dhirubhai Ambani Institute of Information and Communication Technology* under my/our supervision.



Prof. Anil Roy
Thesis Supervisor



Prof. Pritam Anand
Thesis Co-Supervisor

Acknowledgments

It is widely acknowledged that continuous learning is crucial. Throughout my two-year journey pursuing my MTech degree, I acquired valuable skills in addressing real-world challenges within the field of computer science. I am immensely grateful to the Dhirubhai Ambani Institute of Information and Communication Technology for granting me this wonderful opportunity and an exceptional experience.

First and foremost, I would like to extend my heartfelt appreciation to Prof. Anil Roy, Prof. Saurish Dasgupta and Prof. Pritam Anand, who has consistently supported and motivated me. Their extensive expertise in research has been instrumental in guiding me throughout my work. I could not have asked for a better advisor who displayed remarkable patience while developing my MTech thesis.

I am deeply grateful to Mr Tushar Gadhiya for his invaluable guidance and mentorship during my academic journey as a student specializing in software systems. His advice to pursue online courses in Machine Learning during the summer break has been instrumental in expanding my knowledge and skills in this field. Inspired by his guidance, I chose to explore the captivating domain of handwritten answer sheet evaluation using deep learning techniques as the topic for my thesis. I owe a great deal to Mr Tushar for his encouragement and support in nurturing my interest and helping me shape my academic path.

I am also thankful to my fellow research colleague and dear friend, Harshal Vora, for being a reliable source of support. They have always encouraged me to refine my drafts and offered assistance whenever I faced challenges.

Lastly, I would like to thank my family, professors, and friends for investing their valuable time in shaping me into a better individual. I sincerely appreciate the guidance and support each of you has provided me in various situations. Your teachings have had a lasting impact on me, and I will always cherish and remember them.

Contents

Abstract	v
List of Principal Symbols and Acronyms	vi
List of Tables	vii
List of Figures	viii
1 Introduction	1
1.1 Background	1
1.2 Limitations of the traditional evaluation process	2
1.3 Motivation	3
1.4 Objective	3
1.5 Literature Review	5
1.5.1 Overview of text segmentation models	6
1.5.2 Overview of text recognition methods	7
1.5.3 Literature based on text recognition	9
1.6 Plan of the thesis	12
2 All about datasets	14
2.1 Benchmark data	14
2.2 Real-life data collection	15
2.3 Dataset Annotation	19
2.3.1 Dataset Structure	20
3 Classification approach	23
3.1 YOLO Architecture	24
3.2 Experiment details	25
3.3 Results	27
3.4 Discussion	28
3.4.1 Effectiveness of approach	29

3.4.2	Limitations of the Approach	29
3.5	Conclusion	30
3.5.1	Future Research Direction	30
4	Recognition Approach	32
4.1	Proposed Pipeline	33
4.2	Pre-Processing	34
4.3	The Segmentation model	38
4.4	The recognition model	40
4.4.1	Convolutional Neural Network based model	40
4.4.2	Convolutional neural networks and long short-term mem- ory networks (CNN+LSTM)	41
4.4.3	Convolutional neural network with a bidirectional long short- term memory layer (CNN+BiLSTM)	42
4.4.4	Convolutional neural network with bidirectional gated re- current units (CNN+BiGRU)	44
4.5	Results	45
4.5.1	Visual Results	45
4.5.2	Quantitative Analysis of Results	50
4.6	Discussion	54
4.7	Structure of question paper and answer sheet	54
4.8	Scope and limitations of the system	55
4.9	Conclusion and Future work	56
	References	58
	Appendix A APPENDIX: Initial Attempts for Segmentation models	61
A.1	Dataset	61
A.2	Architecture	63
A.3	Discussion and Result	63
	Appendix B Background Knowledge and Model design	66
B.1	Neural Network	66
B.2	Computer Vision And Object Detection	66
B.3	Convolution Neural Network	67
B.4	Recurrent Neural Networks	68
B.4.1	Simple Recurrent Neural Network	69
B.4.2	Long Short Term Memory	70
B.4.3	Gated Recurrent Unit	71

Abstract

Automation has gained significant prominence in various technological domains, offering the potential to streamline processes and minimize human error. Even though it is believed that education and the process of teaching-learning require human-to-human interaction, automation may have tremendous promises here too, particularly in tasks such as student registration, class attendance, administrative duties, and answer sheets evaluation.

This thesis focuses on automating the evaluation of answer sheets submitted by students as a result of a conducted examination. As the scope of the thesis, we have considered answers to only two types of questions, namely, multiple choice questions for which one of the four alphabets: a, b, c or d is selected as the correct answer and objective type questions which have single word answers such as 'Yes' or 'No'; 'correct' or 'incorrect'; 'true' or 'false'. Therefore we need to 'read' these 10 answers to mark it a right answer, else it is a wrong answer. The right answer will fetch a positive mark and the wrong answer will attract a zero or a negative mark, whatever the case may be.

We took a two-pronged approach to achieve automation. At first, we tried the 'classification' approach, in which our system was trained to classify the answers in one of these 10 classes, i.e., a, b, c, d, Yes, No, Correct, Incorrect, True, False. We employed an object detection model, YOLO which was capable of classifying a fixed set of ten classes representing possible answers. This model achieved an impressive accuracy rate of 93% and demonstrated the potential to automate the evaluation of multiple-choice and one-word answer-type questions. Our system worked fine until it found an answer which was of a new class, for example, 'right'. The system tries to read it as one of the 10 classes and that is wrong. It reduces the efficiency of this automation system. To experiment with this approach we created our own handwritten dataset of these 10 classes. It has over 24200 data.

In order to address the limitations of the first approach, we attempted this

problem as a recognition problem. It leveraged text recognition models combining convolutional neural networks (CNN) and recurrent neural networks (RNNs). YOLO was used to recognize each of the 26 alphabets of English script. Then 5 deep learning models, such as CNN, CNN + RNN, CNN + LSTM, CNN + Bidirectional LSTM, and CNN + Bidirectional GRU were used to read the word. These models were capable of recognizing all words written in the answer sheets. We then proceeded to match the recognized words with a fixed set of answers. However, in cases where a match was not found, we considered those responses for further evaluation. Through a comparison of the outputs from each model, we achieved an impressive accuracy of 91%. This outcome underscores the effectiveness of employing diverse methods to automate the evaluation of answer sheets.

In order to achieve it, we used a benchmark IAM word dataset [17]. This dataset was used to train these deep-learning models for effective automation. Then we combine the self-generated dataset of handwritten samples in this approach to test the automatic answer sheet evaluation system.

In future, the remaining two types of questions, e.g., short answer type and long answer type, may also be included in this answer sheet evaluation automation system. This will require NLP based approach to evaluate the answers in a contextual approach.

List of Tables

- 4.1 Predicted and Ground Truth Text of CNN model 46
- 4.2 Predicted and Ground Truth Text of CNN+LSTM model 47
- 4.3 Predicted and Ground Truth Text of CNN+BiLSTM model 48
- 4.4 Predicted and Ground Truth Text of CNN+BiGRU model 49
- 4.5 Comparison of Handwritten Word Recognition Models 50

List of Figures

2.1	Raw Images of DA-IICT Students' Dataset	16
2.2	Scanned Images of DA-IICT Students' Dataset	17
2.3	Images Captured Paper with a background of DA-IICT students' Dataset	18
2.4	Interface of Roboflow tool [14]	20
2.5	Distribution of Data Labels in the Dataset	22
2.6	Heatmap of Annotations: Distribution of Handwritten Text in the Dataset	22
3.1	Data annotation to generate ground thruth	26
3.2	Result of YOLO Model with predicted class confidence on detected bounding box on ruled page	27
3.3	Result of YOLO Model with predicted class confidence on detected bounding box on blank page	27
3.4	Result graphs of loss, precision and recall for training data and validation data	28
3.5	Confusion Matrix for all the predicted class	28
4.1	HTR Pipeline	33
4.2	Pre-Processing Steps for Segmentation Model	36
4.3	Pre-Processing Steps for Recognition Model	37
4.4	Output of Segmentation Model: Segmented image produced by the segmentation model applied to an input image	39
4.5	Flow diagram of convolution neural network based model	40
4.6	Flow diagram of convolutional neural networks and long short-term memory networks (CNN+LSTM) model	42
4.7	Flow diagram of convolutional neural network with a bidirectional long short-term memory layer (CNN+BiLSTM) model	43
4.8	Flow diagram of Convolutional neural network with bidirectional gated recurrent units (CNN+BiGRU) model	44

4.9	The training and validation loss per epoch in CNN	51
4.10	The training and validation loss per epoch in CNN+LSTM	52
4.11	The training and validation loss per epoch in CNN+BiLSTM	53
4.12	The training and validation loss per epoch in CNN+BiGRU	53
A.1	Schematic illustration of CRAFT network architecture[4]	63
A.2	Result of CRAFT model for all answers	64
A.3	Result of CRAFT model for a single character	65
B.1	Architecture of a Simple Recurrent Neural Network (RNN) [13]	69
B.2	Architecture of LSTM (Long Short-Term Memory) model [7]	71
B.3	Architecture of Gated Recurrent Unit (GRU) [7]	72

CHAPTER 1

Introduction

1.1 Background

Education is a critical aspect of a country's development, and India has been investing heavily in its education sector in recent years. Referring to the Economic Survey 2022-23, the Government of India allocated below 3% of the country's 2022 GDP. It includes national and state level budgets both [24]. According to the Unified District Information System for Education (UDISE) 2021-22 report [1], there are 15.1 lakh schools in India, with Gujarat having 54,629 schools. Over 26 crore students across 15.1 lakh schools are being taught by over 95 lakh teachers of pre-primary to higher secondary schools.

In Gujarat, the State Examination Board (SEB) conducts various exams such as the Secondary School Certificate (SSC), and Higher Secondary School Certificate (HSSC). Also, there is a regular system of conducting monthly tests for each class within the schools. The purpose of these continuous exams or grading systems is to provide the students with feedback on their performance. It helps them improve continuously. Most of these exams are pen-and-paper exams. The question papers of any of these pen-and-paper exams may comprise any or all of the four types of questions, which are explained in Section 1.4. Currently, the evaluation process for these exams is manual, where teachers check the answer sheets and grade them manually. This process is time-consuming, requires significant manpower, and can take several weeks or even months due to the large number of students and schools. Above all, there is a finite potential for errors in the evaluation due to human factors.

Instead, consider a situation in which a model or master answer key is created, which is fed into the computer and then the computer is tasked to evaluate the submitted answer sheets of students using an automated system.

In conclusion, automating the evaluation process of answer sheets can provide a significant benefit to the education sector in India, particularly in Gujarat, where the number of students and schools is quite large.

1.2 Limitations of the traditional evaluation process

The traditional evaluation process has several limitations, which affect the accuracy and efficiency of the evaluation process. Firstly, it is time-consuming, and the results may get delayed due to the large number of students and schools. The manual process of evaluating answer sheets is a time-consuming process that can take several weeks or even months to complete. This delay can cause significant problems, especially when announcing the results. Students may have to wait long to receive their grades, which can cause anxiety and stress.

Another major limitation of the manual evaluation process is the potential for errors. Human errors are inevitable, and fatigue and bias can affect the accuracy of the evaluation process. Different teachers may evaluate the same answers differently, even if a model answer is provided to all teachers involved in the evaluation. leading to inconsistencies and a lack of uniformity in the evaluation process. This lack of uniformity can affect the reliability of the results.

Furthermore, the traditional evaluation process requires a significant amount of manpower. The process of collecting answer sheets from various exam centres and distributing them among teachers for evaluation requires a large workforce. This not only makes the process expensive but also creates a burden on the teachers, who have to spend a considerable amount of time evaluating the answer sheets.

The above can be clubbed in the following three major issues of the current evaluation process i.e.

1. Incorrect evaluation due to human error.
2. Time-consuming in collection and distribution of answer sheets for evaluation
3. Time-consuming due to evaluation by humans.

To overcome these limitations, automating the evaluation process can help to reduce the burden on teachers, minimize the time taken to evaluate the answers,

and improve the accuracy of the evaluation process. Hence, there is a need to develop an automated system that can evaluate the answer sheets and provide accurate and uniform results.

1.3 Motivation

Developing an automated system for evaluating answer sheets can address these limitations and may entice transparency too. First, automation can significantly reduce the time and resources required for the evaluation process. Automated systems can process answer sheets quickly, accurately, and consistently. With automated systems, teachers can focus on other aspects of their teaching responsibilities, such as lesson planning, student engagement, and feedback provision.

Automation can improve accuracy and ensures consistency in the evaluation process. The automated system is not prone to personal biases that may creep in due to human evaluation. It can provide a uniform evaluation for all answer sheets, ensuring that each student gets the same marks for similar answers.

The automated system can also shrink the time taken in result announcement and to provide feedback to students, It can help them identify their strengths and weaknesses and improve their performance in subsequent exams in a timely manner. This feedback will be crucial in enabling students to learn and grow, leading to better teaching-learning outcomes in our education system.

In summary, the motivation for developing an automated system for evaluating answer sheets is to address the challenges of the traditional manual evaluation process and provide several benefits, including improved efficiency, accuracy, and consistency of the evaluation process, reduced workload for teachers, and enhanced learning outcomes for students.

1.4 Objective

The assessment of student knowledge and understanding plays a crucial role in the education system. A key component of this evaluation is the administration of question papers, which typically encompass four different question types:

1. Multiple-choice questions (MCQ): These require students to select the correct answer from a given set of options, typically represented by letters such

as A, B, C, or D. Right now we are considering that at primary and secondary school levels these MCQs do not have multiple right answers.

2. Objective-type questions: These questions call for single-word responses, such as “yes” or “no”; “correct” or “incorrect”; “true” or “false”.
3. Short answer type questions: Students are asked to write concise answers within the range of 2 to 3 lines or at the most not exceeding a paragraph.
4. Descriptive answer type questions: These may also be called essay-type questions. Answers to these questions may even exceed a page depending upon the nature of the questions.

Assessing student knowledge and skills across different subject areas is made comprehensive through the use of diverse question types. However, each question type presents unique challenges when it comes to evaluation and grading.

Multiple-choice questions offer a straightforward evaluation process since the correct answer is predefined. Objective-type questions also have fixed answers, requiring students to provide single-word responses. These question types can be evaluated using text segmentation and recognition techniques.

On the other hand, short-answer-type and descriptive-answer-type questions pose additional complexities as they lack predetermined correct answers. Evaluation of such questions requires an automated system to understand the context of the answers to determine their correctness. This goes beyond simple pattern matching and calls for the incorporation of natural language processing (NLP) techniques. NLP allows for the analysis of the meaning and relevance of the written answers, enabling a more comprehensive evaluation.

This thesis specifically focuses on designing an automated system for evaluating the first two types of questions, i.e., MCQs and one-word answer-type questions. The objective nature of these question types, which do not depend on subjective interpretation, makes them suitable for automated evaluation. Additionally, the Gujarat Government Primary School exams, which allocate significant weightage to multiple-choice questions, is a potential customer for an automated system to streamline the evaluation process.

To achieve our objective, we propose a comprehensive system replacing human evaluators’ key task of the manual evaluation process. The system will encompass the following steps:

1. **Answer Sheet Scanning and Line Recognition:** The system will scan the answer sheets and identify the pages containing the answers. It will then proceed to recognize the lines of text where the answers are written.
2. **Text Segmentation:** To accurately identify the location of each word, a text segmentation model will be employed. This model will segment the text regions into individual units for further analysis.
3. **Text Recognition:** Once the text regions are segmented, a text recognition model will be utilized to identify the characters and words written within each segment. This model will enable the system to recognize and extract the answer from the answer sheet accurately.
4. **Matching Recognized Words with Fixed Answers:** The system will compare the recognized words with the fixed set of answers. If a match is found, the corresponding label will be assigned. Otherwise, the system will classify the word as an "other" category, as the system is not accepting any labels except the fixed set of answers.
5. **Evaluation and Result Generation:** Finally, the system will compare the recognized answers with the correct answers and generate the evaluated results accordingly.

The proposed system has the potential to be scalable and adaptable to different types of answers and evaluations. This means that it can be used across different levels of education, from primary to higher education, and can be customized for different types of evaluations, such as subjective questions or essay-type answers.

Overall, the significance of this research lies in its potential to revolutionize the current evaluation process and create a more efficient, accurate, and standardized system that benefits both teachers and students alike.

1.5 Literature Review

Handwriting Recognition has vast practical applications. Some of them are document digitalization, Automatic transcription, signature verification, and education and assessment. It is a challenging task because of the variability in handwriting across regions. These reasons made handwritten text recognition tasks more difficult and gained significant attention in recent years. Our work focuses on the automation of Answer sheet evaluation, which requires knowledge of text

segmentation and text recognition methods. This literature review is structured into three sections: An overview of text segmentation models, an Overview of text recognition methods and a Review of recent studies using deep learning and text recognition for handwritten answer sheet evaluation. These sections provide an overview of the state-of-the-art techniques and their performance in relevant research studies.

1.5.1 Overview of text segmentation models

Text segmentation involves identifying the boundaries of individual characters, words, or lines in a text document. Text segmentation is challenging due to different script styles, variations in the size, style, and orientation of characters and the presence of noise and distortion in the document.

Several text segmentation techniques have been proposed in the literature, including both traditional and deep learning-based methods. This literature review will focus on some of the state-of-the-art text segmentation techniques.

The Efficient and Accurate Scene Text (EAST) detection algorithm is a deep learning-based method for text detection and segmentation [32]. It is the combination of feature extraction and geometry prediction. The feature extraction network is based on a fully convolutional neural network (FCN) that extracts features from the input image. The geometry prediction network predicts the geometry of the text regions, including the score map, the vertical coordinate map, and the horizontal coordinate map.

The EAST model has been shown to achieve state-of-the-art performance on benchmark datasets; ICDAR 2015[19], MSRA-TD500[30], and COCO-Text[28]. The model is efficient, and accurate, and can process multiple scales of images.

Another deep learning-based text segmentation method is the Character Region Awareness for Text Detection (CRAFT) model[4]. The CRAFT model is designed to detect and segment characters in scene text. The model consists of two stages: detection and recognition. In the detection stage, the CRAFT model first detects the character regions in the input image using a fully convolutional network (FCN). Then, the model refines the detected regions using a character region refinement network. In the recognition stage, the CRAFT model uses a character recognition network to recognize the characters in the segmented regions.

The CRAFT model has demonstrated state-of-the-art performance on ICDAR 2013[26], ICDAR 2015[19], and Total-Text[8] benchmark datasets. The model is robust to variations in text size, orientation, and style and can handle curved text.

While deep learning-based text segmentation methods have shown remarkable performance, traditional techniques, such as connected component analysis (CCA), are still widely used. CCA is a simple and effective technique for text segmentation that involves identifying connected regions of pixels in an image.

Basu et al. [16] used CCA for line text segmentation in ICDAR2009. The authors demonstrated that their method works outstanding in English, German, Greek, and French language handwritten text with an accuracy of 93%.

In summary, text segmentation is an important task in document analysis, scene text analysis and OCR. While traditional techniques such as CCA are still widely used, deep learning-based methods such as EAST and CRAFT have demonstrated state-of-the-art performance on benchmark datasets. The EAST and CRAFT models are efficient, accurate, and robust to variations in text size, orientation, and style. However, there is still scope for further research to improve the performance of text segmentation methods, especially for handling noise and distortion in the document.

1.5.2 Overview of text recognition methods

Over the years, researchers have developed a wide range of text recognition methods, from traditional rule-based to deep learning-based approaches. This literature review will explore some of automated answer sheet evaluation systems' most promising text recognition techniques. Most important of them are:

1. Rule-based methods: Rule-based methods are the earliest text recognition techniques used in OCR systems [10]. These methods typically involve defining rules or heuristics to analyze text characteristics and identify characters. Rule-based methods can be highly accurate when dealing with simple fonts and layouts but are limited in handling complex or variable fonts and layouts.
2. Template matching: Template matching is a common text recognition technique that involves comparing a portion of an image with a pre-defined template to identify characters [10]. This approach works well for consistent text in a known font but can struggle with font, size, and style variabil-

ity. Furthermore, the need for predefined templates can significantly limit template-matching methods.

3. Feature-based methods: Feature-based methods involve extracting specific features from the text, such as edges, corners, or lines, and using these features to identify characters [3]. These methods can be effective when dealing with variable font and size but can struggle with complex layouts and styles.
4. Machine learning-based methods: In recent years, machine learning-based methods have emerged as some of the most promising text recognition techniques. These methods use algorithms to learn from examples and improve their ability to recognize text. Some of the most commonly used machine learning-based methods in text recognition include CNNs and RNNs.
 - Convolutional Neural Networks (CNNs): CNNs have shown great promise in text recognition due to their ability to learn and extract features from images automatically. These networks can be trained on large datasets of annotated text images to improve their ability to recognize characters accurately [9, 15, 21, 29, 32].
 - Recurrent Neural Networks (RNNs): RNNs are particularly useful in recognizing text sequences, such as words and sentences, due to their ability to capture the contextual information of each character. These networks can be trained on large text datasets to learn the relationships between characters in a sequence and improve recognition accuracy [29].
 - Long Short-Term Memory (LSTM) Networks: LSTM networks are a type of RNN that can better handle long-term dependencies in text sequences. They can be particularly useful in recognizing cursive or joined handwriting, where the characters are often linked [6, 9].
 - Encoder-Decoder Networks: Encoder-Decoder networks consist of two parts: an encoder that maps the input image to a fixed-length representation and a decoder that maps the representation to a sequence of characters. These networks have shown great promise in recognizing variable-length text sequences, such as paragraphs or entire documents [29].
 - Hybrid approaches: Hybrid approaches combine multiple text recognition techniques to improve accuracy and handle complex text layouts.

For example, a hybrid approach may use a rule-based method to identify text regions in an image, followed by a machine learning-based method to recognize the characters in those regions [6, 21, 29].

In conclusion, text recognition is a critical component of automated answer sheet evaluation systems, enabling machines to accurately and efficiently interpret student responses. While rule-based and template-matching methods have been used for many years, the emergence of machine learning-based methods has shown great promise in improving accuracy and handling complex text layouts. As the field of automated answer sheet evaluation continues to evolve, hybrid approaches combining multiple text recognition techniques will likely become increasingly prevalent.

1.5.3 Literature based on text recognition

This section covers papers that utilize the methods mentioned in Section 1.5.2 and provides details about the models used and their performance.

Bharadia et al. [5] proposed a system for the automatic evaluation of short answer questions using machine learning techniques. The authors propose a system that uses natural language processing techniques to address this issue to evaluate short-answer questions. The system pre-processes the input answer to remove irrelevant information and then applies a machine-learning model to evaluate the answer. The authors use the Support Vector Machine (SVM) algorithm for classification and achieve an accuracy of 75-87.5%. The authors evaluate their proposed system on a dataset of 800 answers and compare it with the manual evaluation system. They show that their proposed system is 300% more time-efficient and can evaluate 1100% more answers than the manual system.

A method for the automatic evaluation of handwritten answer sheets using deep learning techniques was presented by Rahaman et al. [21]. The proposed algorithm extracts the text from the answer sheet and uses a convolutional neural network (CNN) to recognize handwriting. The system then assesses the response using natural language processing (NLP) methods to determine the text's meaning. On a dataset of 400 answer sheets, the proposed system was tested, and it obtained an accuracy of 85%.

Zimmerman et al. [33] proposed an automatic word segmentation scheme for offline cursive handwriting. The segmentation approach consists of two steps; In the initial step, they used the Viterbi decoder. It uses normalized text lines and

their transcription for the best word boundaries of the Hidden Markov Model-based Recognition system in forced alignment mode. The next step is the word boundaries to separate words and punctuation characters. The proposed approach resulted in accurate bounding box information and matching word labels for about 25000 handwritten words in the IAM database. On an IAM dataset, they were able to segment words correctly with a rate of 98 %.

Bluche et al. [6] presented an attention-based model for end-to-end handwriting recognition. The paragraph-level dataset images from the RIMES [11] and IAM datasets were used in this paper. The authors introduced a novel joint line segmentation approach, combining attention mechanisms with a multi-dimensional Long-Short-Term Memory Recurrent Neural Network (LSTM). They modified the standard LSTM architecture by incorporating an attention mechanism into the collapse layer. This modification enables a segmentation-free method for both text segmentation and transcription. The weights assigned by the attention mechanism help identify the position of the input in the paragraph image iteratively. All the images of the dataset have 300 dpi resolution in the experiment; the method achieved a CER of 4.9% on the IAM dataset and 2.5% for the RIMES datasets.

Krishnappa et al. [15] proposed a general CNN-based approach to analyze the handwritten document in Kannada. The Chars74K dataset, consisting of more than 657 classes, each containing 25 handwritten characters, was utilised during the experiment. Various data augmentation techniques were employed to increase the dataset's size, including denoising, binarization, grayscale, segmentation and contrast normalization. On the Chars74K dataset, it achieved an accuracy of 99% . Also tested on separate custom datasets, the model achieved an accuracy of 96%.

Chung et al. [9] extended the work presented by Bluche for the attention-based model. It introduced a full-page offline handwritten text recognition framework. The pipeline of this framework includes the location of handwritten text with object detection and the recognition of handwritten text with feature extraction. The CNNs were used with fully feed-forward LSTM models. The character error rate (CER) of the proposed method is 8.5%. The main advantage is the reduction in computation costs compared to existing methods.

Wukunnan et al. [29] introduced a method for detecting and recognizing handwritten text and lines. They proposed a novel approach called Multi-level Convolutional and Recurrent Network (MLC-CRNN), which integrates deep learn-

ing techniques such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and the Connectionist Temporal Classification (CTC) loss function. To train a handwriting text-line detector, the authors utilized the Connectionist Text Proposal Network (CTPN), a specific model designed for this purpose. This training process aimed to improve the accuracy of the text-line detection stage within the overall MLC-CRNN framework. The dataset was collected from 300 students, the training contains 3883 images, and the testing contains 297 images. The MLC-CRNN model combined with two Multi-level Convolutional modules achieved the best performance, achieving an accuracy of 91%.

Singh et al. [25] introduced a full-page handwriting document recognition model. The model utilized a Convolutional Neural Network (CNN) to extract relevant features from the document. The authors utilized several datasets, such as IAM, ANSWERS2, WIKITEXT and FREE FORM ANSWER, to train the model. These datasets provided a diverse range of handwritten documents for training the model which eventually lead to developing an evaluation system. The model achieved a CER of 7% on the FREE FORM ANSWER dataset, demonstrating its effectiveness in accurately recognizing and transcribing handwritten text within full-page documents.

We thus explained above the latest advancements in Handwritten Text Recognition (HTR) and their potential applications. Although the papers reviewed were not directly focused on automated answer sheet evaluation for MCQs and objective type questions, they helped us understand the benefits of the deep learning techniques in accurately recognizing handwritten text. These findings were crucial for the foundation of developing an effective and efficient automated system for recognizing answers in answer sheets.

The referred papers reveal a variety of approaches and techniques, such as attention-based models, CNNs, RNNs and CTC loss functions, that have shown promising results in HTR. Some of the methods, such as the CCA and MLC-CRNN models, have successfully detected and recognized handwritten text and lines, indicating the potential for automated evaluation pipeline.

This literature survey provides valuable insights and a strong foundation for developing an automated answer sheet evaluation system using deep learning techniques. By utilizing the techniques and approaches discussed in this section, it is possible to design an accurate and efficient system for recognizing answers for MCQs and objective type questions. We believe that it would ultimately save

time and efforts of teachers, educators and examiners.

1.6 Plan of the thesis

Before we discuss about how we collected data, we need to understand how data would be generated. For the purpose of the development of an automated answer sheet evaluation system, we assume that there would be a class room examination and a teacher would be there as an invigilator. The question paper has two types of questions described in the previous chapter, i.e., MCQs and objective type questions. The students are allowed to write any of the 10 letters or words (let's call them classes) as answer of each question. Once the time of the exam is over, students submit their answer sheets to the teacher. The teacher is then supposed to take photo of the page of the answer of each student bearing his or her unique ID (numeric roll number) and the answers using his/her smart phone. These images are then sent to the central server using wifi and internet. Please note that the network security and encrypted data transmission are out-of-the-scope of this thesis.

Our automated answer sheet evaluation system is deployed at the central server. This server receives all the answer sheets in form of uniquely identifiable images. Again, this part of unique identification of answer sheets is kept out-of-the-scope of the thesis. So we assume that the network is highly secured and these images have unique identification with each of the students.

Broadly speaking this automated system has to take these three major actions serially:

1. Pre-processing of the image, explained in Section 4.2
2. Text segmentation, explained in Section 4.3
3. Text recognition including matching with the answer key for evaluation, explained in Section 4.

The implementation of these three actions is conducted using two different approaches. Initially, we treated each answer as a class and attempted to detect and classify the text in a single go. This approach is called object detection and classification, described in detail in Section 3. This approach had a limitation. If a student wrote an answer that was not included in the predefined set of answers, the system would still attempt to classify it from the given fixed set, which could

lead to incorrect predictions. To overcome this problem, we have implemented a text recognition approach explained in Section 4. It identifies each character of the extracted answer and generates the corresponding word. This approach has the scope of post-processing and has the capability to identify words that may be not part of the predefined set of 10 answers. By recognizing each character and forming words, this approach shows greater flexibility and adaptability in handling a wider range of possible answers too.

CHAPTER 2

All about datasets

Having established the boundary and scope of this thesis, let's now understand how we planned to achieve this. Any deep learning based model or system requires a benchmark data for training and testing. It means we need to have a dataset with ground truth. The system self-learns and continuously improves its efficiency depending upon the size of the data in the dataset. After developing a system on such benchmark data, if we give the system some real-life data, it predicts the result. Hence the accuracy and effectiveness of such developed system largely depends upon the robustness of the benchmark data and appropriately chosen real-life data.

This chapter, therefore, has four parts about how we dealt with data while working for the development of a robust automated answer sheet evaluation system:

1. Benchmark data
2. Real-life data collection
3. Data annotation
4. Dataset Structure

2.1 Benchmark data

We have used two types of benchmark datasets in our experiments; For the classification approach described in Section 3, a custom-generated dataset of DA-IICT students is used and for the recognition model discussed in Section 4, two datasets are used first, is IAM word dataset [17] and the second is DA-IICT students' dataset [22].

2.2 Real-life data collection

The goal was to extract the answers from the answer sheet and recognize these answers from a fixed set of answers. To achieve this, we needed a dataset of handwritten English words, including all the answers.

We decided to create our dataset by collecting data from students at the DA-IICT Institute. The dataset we required was for 10 specific answers, MCQs and SAQs, namely "a", "b", "c", "d", "true", "false", "correct", "incorrect", "yes" and "no".

The data collection process involved providing the students with blank and ruled pages and asking them to write each word five times in a single row. The dataset was divided into 50 instances per page, with each word being written five times in both capital and small letters shown in Fig. 2.1. This ensured that we had enough data for training and testing our models.

We ensured that the students wrote the words legibly and that the handwriting was diverse, covering a range of handwriting styles and variations. We also provided the students with the option to capture the image of the page on their mobile phones or provide us with the answer paper, which we scanned on a scanner. Fig. 2.1 shows the images captured by mobile phones. some of the data samples cover the paper and background both illustrated in Fig. 2.3. Fig. 2.2 shows the images scanned by the scanner. We made sure that the images were of high quality and that the text was legible.

However, we must ensure the dataset is free of noise or anomalies that may negatively impact the model's performance. To achieve this, we will perform data cleaning and preprocessing before using the dataset for training and testing our models.

A	a	B	b	C
C	D	d	A	a
B	b	C	C	D
d	A	a	B	b
*True	false	TRUE	FALSE	TRUE
TRUE	FALSE	true	False	TRUE
correct	incorrect	correct	incorrect	correct
CORRECT	INCORRECT	correct	incorrect	correct
yes	no	YES	NO	YES
YES	NO	yes	no	yes

a	B	c	d	A
A	B	C	D	A
a	b	c	d	d
A	b	C	D	c
TRUE	TRUE	true	True	true
FALSE	FALSE	False	False	FALSE
correct	incorrect	correct	incorrect	correct
CORRECT	INCORRECT	correct	incorrect	correct
yes	Yes	yes	yes	YES
no	NO	no	No	NO

a a a a a
b b b b b
c c c c c
d d d d d

True true True true true
false false False false false
Correct Correct Correct Correct
Yes yes yes Yes Yes
No no no No No

a	b	c	d	A
B	A	d	c	B
C	C	D	A	B
D	c	a	B	C
TRUE	FALSE	true	True	true
FALSE	TRUE	False	False	FALSE
correct	incorrect	correct	incorrect	correct
INCORRECT	correct	incorrect	correct	incorrect
YES	yes	no	YES	NO
NO	no	yes	NO	NO

D	A	B	C	b
a	a	b	C	a
d	d	c	b	d
a	c	c	a	B
A	B	D	B	C
True	True	False	False	True
*True	True	true	False	False
False	Correct	Correct	Incorrect	Correct
Incorrect	Incorrect	Correct	Correct	Incorrect
Yes	No	yes	no	Yes
No	Yes	No	No	No

A	c	D	d	a
b	a	a	B	c
C	c	D	d	a
a	A	b	B	c
True	true	False	FALSE	TRUE
TRUE	true	FALSE	FALSE	true
Correct	correct	INCORRECT	Incorrect	INCORRECT
CORRECT	correct	Incorrect	Incorrect	Incorrect
Yes	no	NO	YES	yes
No	no	Yes	yes	YES

Figure 2.1: Raw Images of DA-IICT Students' Dataset

1. a b a d c
 2. c a b d a
 3. b c a d c
 4. b a c d c
 5. d c b a a

1. True False True True False
 2. False False True False True

1. Correct Incorrect Incorrect Correct Correct
 2. Correct Correct Incorrect Correct Correct

1. Yes No Yes Yes No
 2. No Yes Yes No No

a b c d e
 b c d e c
 c d e a b
 d e a b c
 e a b c d

True false False True True
 False False True True false
 Correct Correct Incorrect Correct Incorrect
 Incorrect Incorrect Correct Incorrect Correct
 YES No No No YES
 Yes NO YES NO YES

b c b c a
 a d c d b
 c c a c b
 d A B D e
 True True TRUE true false
 false True false FALSE false
 Correct Incorrect correct INCORRECT Correct
 Incorrect Correct correct Incorrect incorrect
 Yes No Yes no NO
 yes no NO YES NO

a b c d a
 b c a b d
 a d c d c
 d c d c d
 a c d b a

True False True True True
 False False False False False
 Correct Incorrect Incorrect Incorrect Incorrect
 Correct Correct Correct Correct Correct
 Yes Yes Yes No Yes
 No No Yes No Yes

a a A a A
 b b B b B
 c c C c C
 d d D d D
 true true TRUE True True
 false false FALSE False False
 correct correct CORRECT Correct Correct
 incorrect incorrect INCORRECT Incorrect Incorrect
 yes yes YES Yes Yes
 no no NO No No

a a A a A
 b b B b B
 c c C c C
 d d D d D
 true true TRUE True true
 false false FALSE False false
 correct Correct CORRECT Correct correct
 incorrect incorrect INCORRECT Incorrect incorrect

Figure 2.2: Scanned Images of DA-IICT Students' Dataset

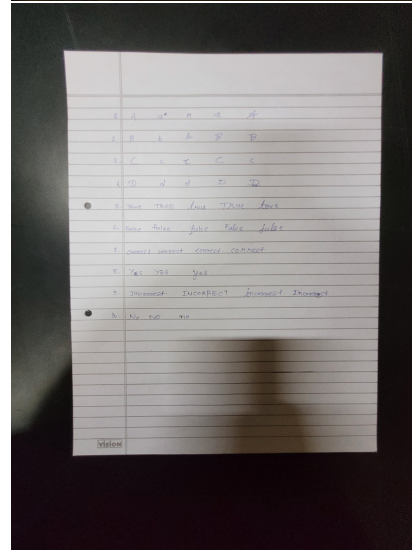
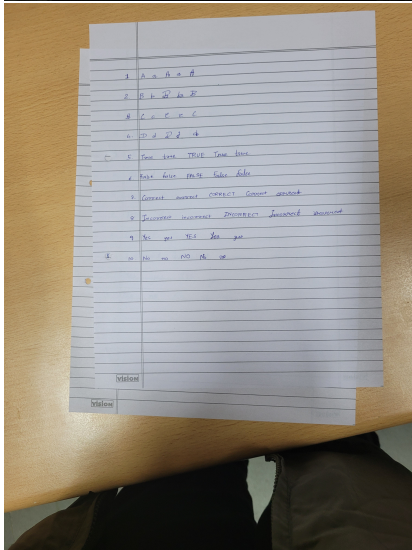
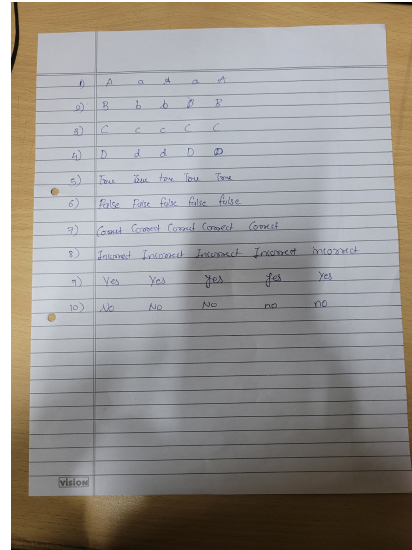
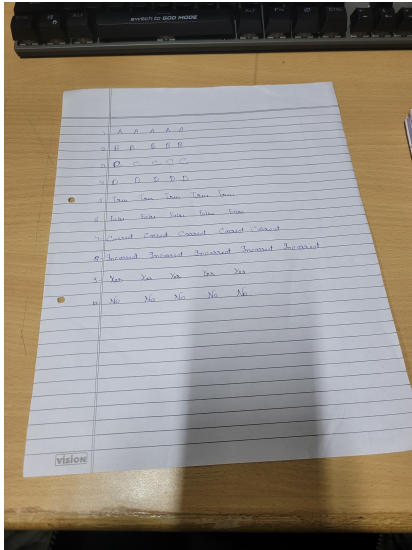


Figure 2.3: Images Captured Paper with a background of DA-IICT students' Dataset

2.3 Dataset Annotation

This section contains an explanation of dataset annotation in the context of using Roboflow.

We have used the Roboflow tool for the task of annotation. Roboflow is a platform that provides tools to help with the annotation and preprocessing of image datasets, among other things. To annotate the dataset of handwritten English words collected from students, we created a project in Roboflow and invited annotators to label the dataset.

The first step in annotating the dataset is to upload the images to Roboflow. Once the images are uploaded, Invited Annotators created bounding boxes around the text regions in the images and label the text within the boxes.

To create a bounding box, the annotator selected the region of the image that contains the text using a mouse shown in Fig. 2.4. The annotator then drew a box around the selected region and provided the label for the text within the box. For example, if the selected region contains the word "true", the annotator would label the box as "true".

If you have fixed the format of the dataset then Roboflow provides several tools to make the annotation process more efficient. For example, it can automatically divide the images into individual text regions and provide suggestions for labels based on previously labelled images. This reduces the amount of manual effort required to label each image.

Once the annotation is completed, the dataset is exported in a suitable format for training and testing text segmentation and recognition models. In this exported dataset, all the labels are replaced with integers. In our set of answers, we sorted the answers alphabetically in ascending order and replaced them with integers. the mapping looks like { "a": 0, "b": 1, "c": 2, "correct": 3, "d": 4, "false": 5, "incorrect": 6, "no": 7, "true": 8, "yes": 9 }. This transformation is necessary as deep learning models work with numeric data, and categorical data needs to be encoded as integers for ease of training and testing. The exported dataset includes the following components:

1. **Original Images:** The images from the dataset that were annotated for text regions.
2. **Annotated Bounding Boxes:** The bounding boxes outline the text regions

in the images.

3. **Integer Labels:** Each text region in the dataset is assigned a corresponding integer label, representing the specific category or class to which it belongs.

The models use the annotated data to learn and improve their ability to identify and recognize text in images. Additionally, the annotated data can be used to evaluate the performance of the models by comparing the predicted text regions and labels with the ground truth annotations.

In summary, dataset annotation involves creating bounding boxes around text regions in images and labeling the text within the boxes. It is necessary to use tools like Roboflow to make the annotation process more efficient and can export the annotated dataset in a format suitable for training and testing text segmentation and recognition models. The annotated dataset is essential for training and evaluating text segmentation and recognition models.

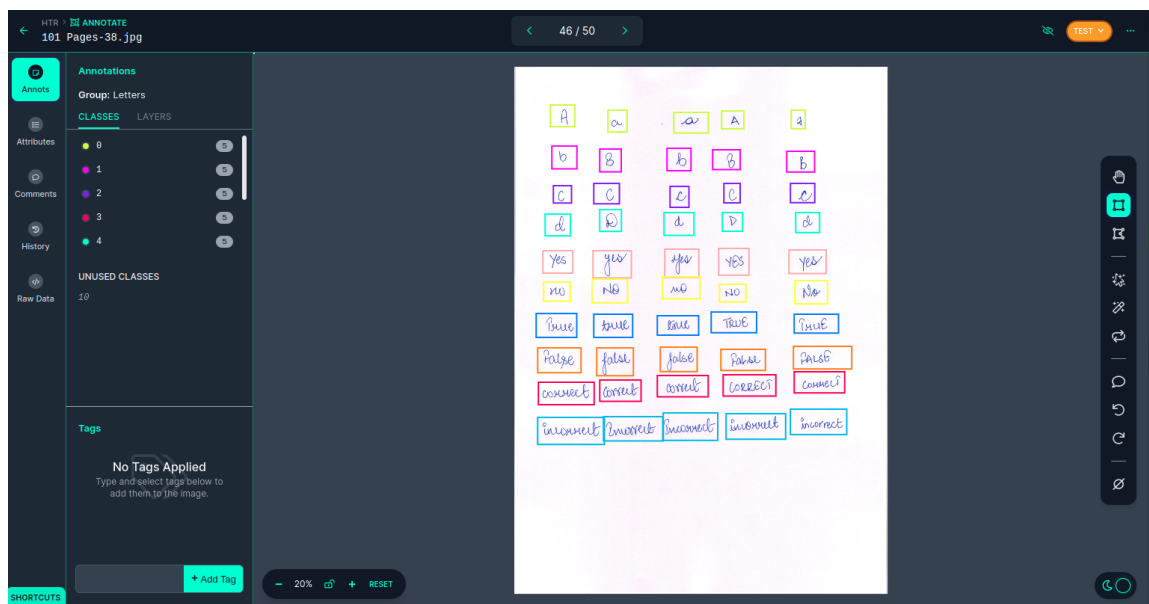


Figure 2.4: Interface of Roboflow tool [14]

2.3.1 Dataset Structure

In the automated answer sheet evaluation system, each model considers a different dataset structure for the input to the model. In the implementation, the dataset used for the text segmentation model YOLOv5 consisted of scanned dataset images in JPEG format and a corresponding text file containing details of annotations. Each row of the text file contained the path to the image, the height and

width of the bounding box, the centre x and y coordinates of the bounding box, and the label specifying the text written in the bounding box.

The dataset was divided into three parts: the training set, validation set, and test set. The training set contained 70% of the data, the validation set contained 15%, and the test set contained 15%. The dataset was shuffled before partitioning to ensure that the distribution of data in each set was similar.

For text recognition, a different dataset format was used. The dataset for text recognition consisted of images of extracted words in PNG format and a corresponding annotation file in a CSV format. Each row of the CSV file contained the path to the image, the text written in the extracted word images.

The text recognition dataset, like the segmentation dataset, was divided into three parts: the training set, validation set, and test set. This division allows for effective training, fine-tuning, and evaluation of the model's performance. The partitioning was performed using the same technique as for the segmentation dataset.

To streamline the data loading process during training, a YAML file was created for each dataset, specifying the location of the data, the file format, the column names in the annotation file, and the number of classes in the dataset. This YAML file was then used in the code to load the dataset, making changing the dataset. It makes the process of loading the dataset more manageable and efficient.

Figure 2.5 illustrates the distribution of instances for each data label in the entire dataset, revealing an approximate count of 2200 to 2500 instances per label. These counts refer to the clean data, where only properly written words are considered from the annotations while discarding other labels not written properly.

The median dimensions of the images in the dataset are (512,512). Out of the total 505 images, 355 have dimensions below (1024,1024), indicating smaller sizes, while 150 images are larger in size. Figure 2.6 displays a heatmap of the annotations, illustrating the density distribution of the dataset. This visualization assists in understanding the prominent regions where text is likely to be found.

Distribution of Labels

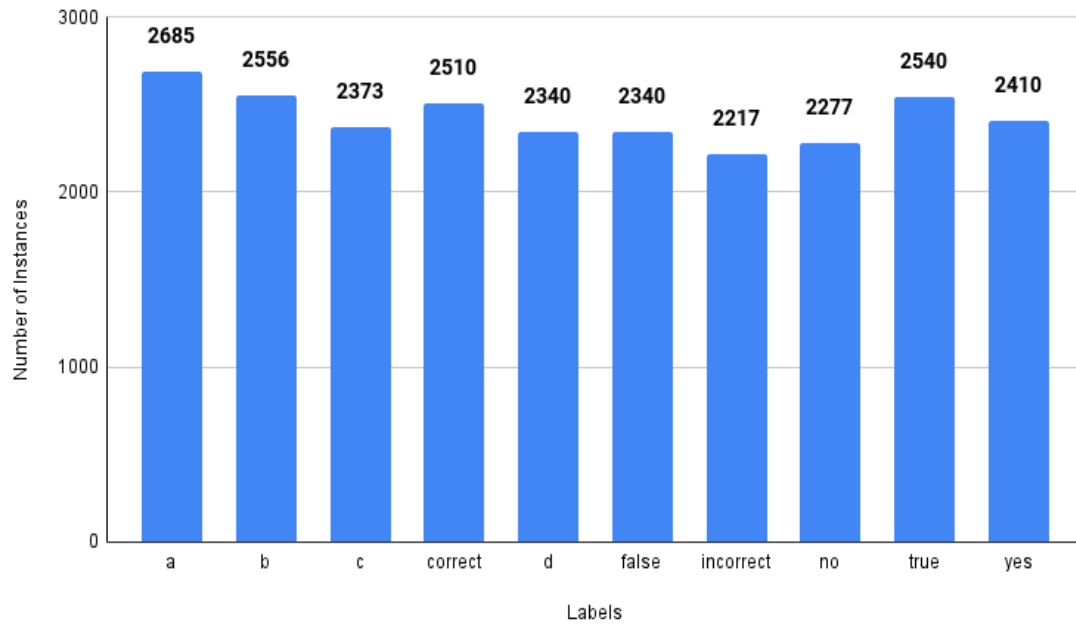


Figure 2.5: Distribution of Data Labels in the Dataset

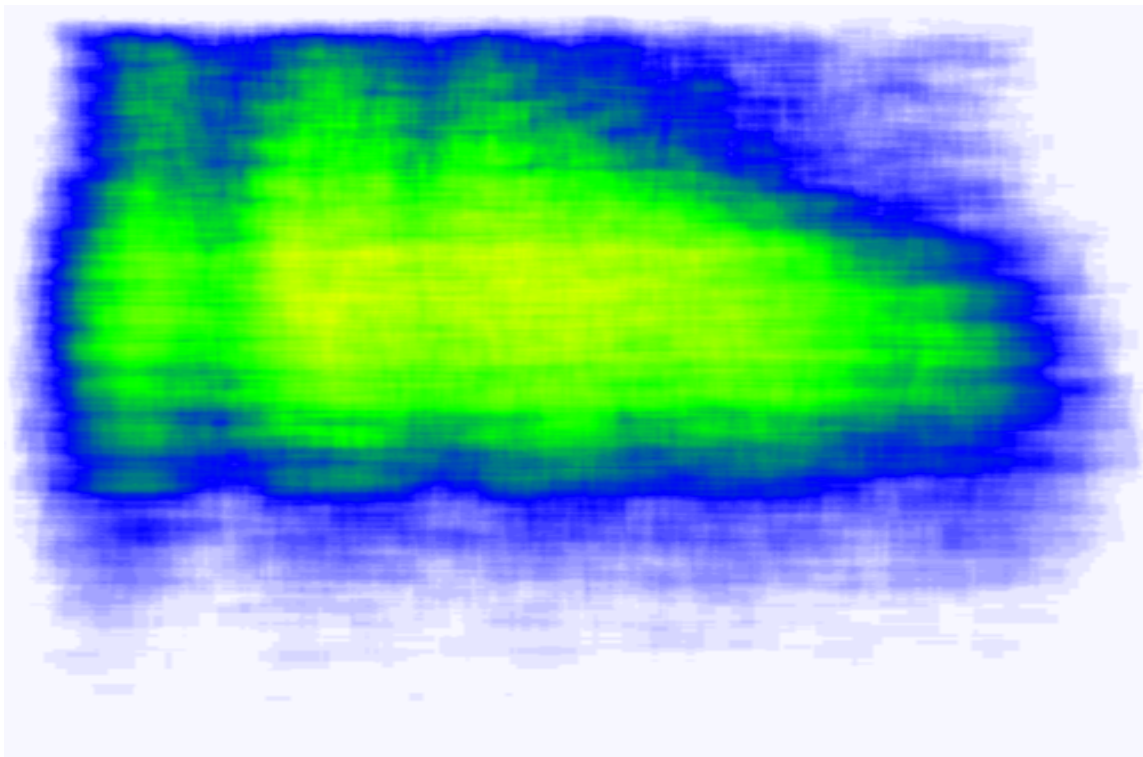


Figure 2.6: Heatmap of Annotations: Distribution of Handwritten Text in the Dataset

CHAPTER 3

Classification approach

We implement an object detection and recognition method to resolve the problem of identifying handwritten text and corresponding labels in an image of an answer sheet. Mathematically, we can define this problem as follows:

Given an image x containing handwritten English language answers, the goal is to predict the corresponding label for each text region. Let the set of possible labels be $L = \{ a, b, c, d, \text{true}, \text{false}, \text{correct}, \text{incorrect}, \text{yes}, \text{no} \}$. We can represent this problem as a function $f(x)$ that maps an input image x to a set of predicted labels l_1, l_2, \dots, l_n , where each label $l_i \in L$.

We can use the YOLO (You Only Look Once) object detection model to detect the text regions in the image and predict their corresponding labels. YOLO is a deep learning-based object detection model that can detect multiple objects in an image and classify them into different labels. It works by dividing the input image into a grid and predicting bounding boxes and class probabilities for each grid cell. Mathematically, we can define the YOLO model as mentioned in Equation (3.1).

$$f(x) \rightarrow (x_1, y_1, w_1, h_1, l_1), (x_2, y_2, w_2, h_2, l_2), \dots, (x_n, y_n, w_n, h_n, l_n) \quad (3.1)$$

In this case, the input feature x is the image containing handwritten text. The function $f(x)$ represents the YOLO object detection model that takes the input feature x and maps it to a set of tuples, where each tuple contains the coordinates (x_i, y_i) of the corner present at top-left direction, the width (w_i) and height (h_i) of the bounding box surrounding the text area, and the corresponding label (l_i) of the text area.

In multiclass classification, a softmax classifier is used to predict the label for each text region. The output of the YOLO model is given as an input to the softmax classifier which predicts the probability of each label for the text region. The

label with the highest probability is selected as the predicted label for that text region.

The goal is to use the YOLO model and softmax classifier [2] that produce the highest accuracy in detecting the text regions and predicting their corresponding labels in the image.

3.1 YOLO Architecture

This problem can be handled by two approaches, the single-shot recognition approach and the two-step recognition approach. We have used a first approach for which the YOLO model is used for the classification of ten answers. It uses a full convolution approach in which the network is able to find all objects within an image in one pass through the convent.

The accurate identification and categorization of Text Regions of Interest (ROI) are crucial to ensure high-quality input for the Handwritten Text Recognition (HTR) model. To accomplish this, the YOLOv5 algorithm is employed for the detection and classification of an object, which is already proven to be successful for the same. YOLOv5 works by partitioning the image into a grid system, with each object being identified within its corresponding grid cell. It is an improvised version of the YOLOv3 method suggested by Joseph and Ali [23]. Although there is no academic publication specifically for YOLOv5 [14], the conceptual framework of YOLOv3 is given because it forms the basis for the creation and improvement of YOLOv5.

In YOLOv3, the bounding box coordinates are predicted as l_x , l_y , l_w , and l_h . The subscripts x , y , w , and h in Equation (3.2), (3.3), (3.4), (3.5) are the x and y coordinates, width, and height of the bounding box. These values are used to determine the precise location and size of the detected object, here it is text area. The model is trained using the sum squared error. Additionally, the objectness score, the likelihood of being categorized into a specific object, is measured by logistic regression in the model. The feature extractor in YOLOv3, known as DarkNet-53, is composed of 53 layers. This architecture is designed to efficiently utilize the GPU, resulting in faster evaluation times during object detection. Three alternative scales are used by YOLOv3 to predict bounding boxes and extract characteristics from them. The output is a 3D tensor that contains the bounding box coordinates (a_x, a_y, a_w, a_h) , the objectness score, and the predicted classes. Equation (3.2) defines the x coordinate of the bounding box (a_x), Equation (3.3) defines

the y coordinate (a_y), Equation (3.4) defines the width (a_w), and Equation (3.5) defines the height (a_h). These equations are used to calculate the precise position and size of the predicted bounding boxes.

$$a_x = \sigma(l_x) + o_x, \quad (3.2)$$

$$a_y = \sigma(l_y) + o_y, \quad (3.3)$$

$$a_w = p_w \cdot e^{l_w}, \quad (3.4)$$

$$a_h = p_h \cdot e^{l_h}, \quad (3.5)$$

where p_w, p_h are the prior bounding box's width and height and o_x, o_y is the offset from the top left corner of an image. $\sigma()$ stands for the sigmoid function. On the foundation of YOLOv3's fundamentals, various YOLOv5 versions have been presented[12]. For example, YOLOv5n has the fewest parameters (1.9 million), YOLOv5s has the most parameters (86.7 million), YOLOv5m has 21.2 million parameters, YOLOv5l has 46.5 million parameters, and YOLOv5x likely refers to the same parameter count as YOLOv5n. These options provide flexibility in selecting a model based on specific requirements regarding model size, computational resources, and performance.

3.2 Experiment details

The System Specification required for all the experiments performed in the thesis should have a minimum of 8GB RAM and an octa-core CPU to run the model smoothly and efficiently.

The model execution procedure begins with implementing the YOLOv5 model for answer type classification. Specifically, the YOLOv5s model is chosen for this experiment. The annotated dataset is used for the YOLOv5 model. Section 2.3 has the information of annotation and Fig. 3.1 shows annotated bounding boxes and the assigned labels to it. The DA-IICT students dataset [22] is divided into ten different classes, and 355 images are used for training and validation, having a split ratio of 7:3. The model is then trained for 500 epochs, using a batch size of eight and an image dimension of (512,512). The best weights with the highest precision are kept for further evaluation and usage throughout the training process.

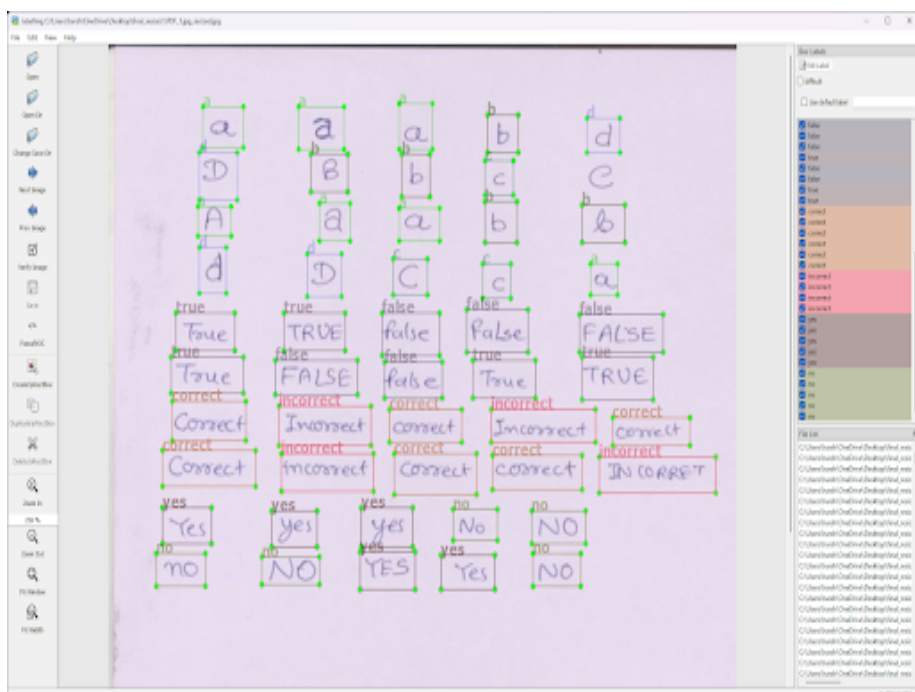


Figure 3.1: Data annotation to generate ground truth

3.3 Results

Figs. 3.2 and 3.3 illustrate the bounding box, where the first label indicates the class and the percentage value represents the confidence of the predicted label. To understand class identification consider the example, the label 'a' is mapped to 0; therefore, the bounding box with label 0 represents the class 'a'.



Figure 3.2: Result of YOLO Model with predicted class confidence on detected bounding box on ruled page

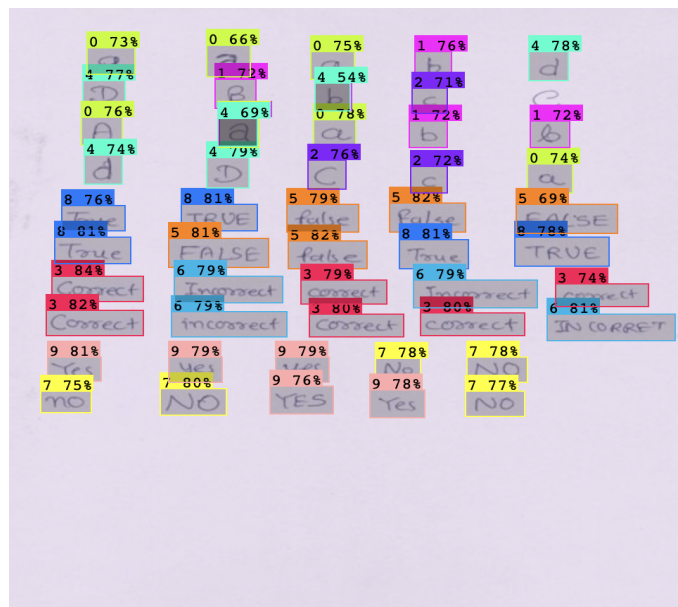


Figure 3.3: Result of YOLO Model with predicted class confidence on detected bounding box on blank page

The Region of Interest (ROI) for each of the ten labels has been successfully identified using the YOLOv5 model, with favourable findings. The model acquires an outstanding precision of 93.5% and a recall of 91.30%, with a mean average precision (mAP) of 94.8% at the 0.5 confidence threshold. As shown in Fig. 3.4, while training, the loss graphs for bounding box detection, object error, and classification loss consistently decreased resulting in good accuracy. In the confusion matrix, shown in Fig. 3.5, one may see that most of the classes have true positive values which are above 90%, except for classes 'b' and 'd'.

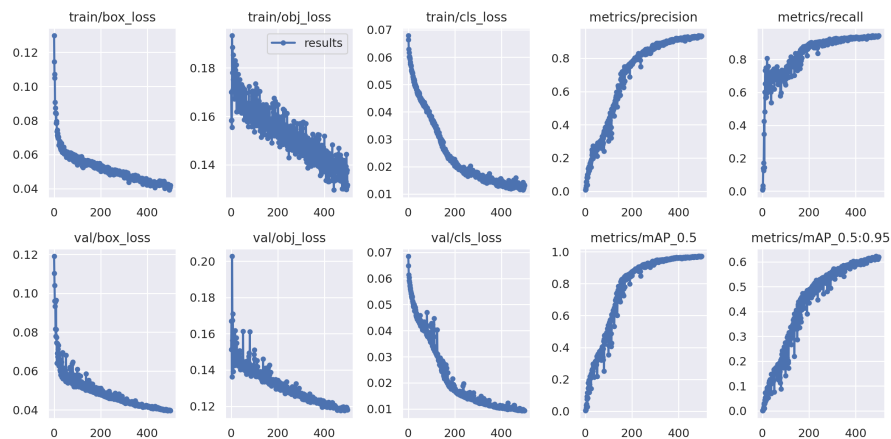


Figure 3.4: Result graphs of loss, precision and recall for training data and validation data

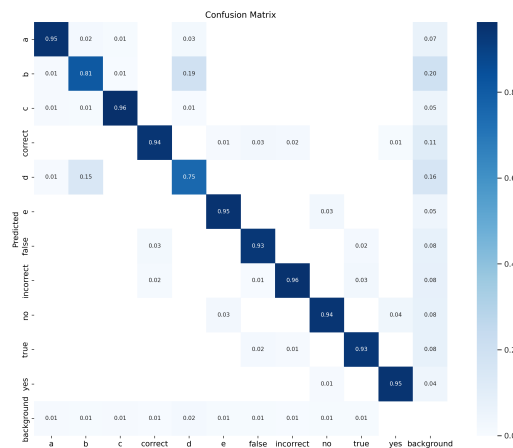


Figure 3.5: Confusion Matrix for all the predicted class

3.4 Discussion

This section explains the efficacy of the suggested approach and addresses its limitations. It also provides insights into future research directions.

3.4.1 Effectiveness of approach

To compare the effectiveness of our own approach with other state-of-the-art methods, we looked at previous research in the area of object detection and recognition for handwritten text. One commonly used benchmark dataset for such tasks is the ICDAR (International Conference on Document Analysis and Recognition) dataset, which contains various types of documents, including handwritten text.

One recent study used the ICDAR 2019 [31] dataset for handwritten text detection and recognition and achieved a mean average precision (map) of 0.75 and 0.70 for detection and recognition, respectively, using a combination of deep learning-based models (Faster R-CNN and CRAFT). Another study on the same dataset achieved a map of 0.82 for text detection using the EAST (Efficient and Accurate Scene Text) model.

Compared to these studies, our approach using the YOLOv5 model achieved a map of 94.8% for detecting the 10 labels on an answer sheet, which is more than the state-of-the-art result. However, it should be understood that our task is a little different, as we are specifically detecting and recognizing a limited set of labels on answer sheets, rather than general handwritten text on a document. Additionally, our model was trained on a smaller dataset specific to answer sheets, while the previous studies used the much larger ICDAR dataset.

3.4.2 Limitations of the Approach

While the proposed approach has shown promising results in accurately identifying the regions of interest (ROI) for the 10 pre-defined classes, it is important to acknowledge and address certain limitations.

1. The model may tend to predict the given input as one of the elements of the given set of classes, which limits the ability to detect new or unknown classes.
2. The model may not be scalable to accommodate new classes or changes in the types of answers in the answer sheet, as it relies on pre-defined classes. Adding a new class will lead to re-training of the entire model.
3. The model has no scope for post-processing the text that is detected, which can limit the accuracy of the classification and lead to errors in the evaluation of the answer sheet. For instance, an input such as "Thank you" may be predicted as "Incorrect", which can result in an incorrect evaluation.

4. Accuracy of the model may be affected by the quality of the input image, which can contain noise, blur, or other artefacts that can make it difficult to detect and classify the words accurately.

Overall, while the proposed approach has demonstrated promising results, these limitations need to be taken into account when applying the approach in practice. Future research can focus on addressing these limitations and further improving the accuracy and scalability of the approach.

3.5 Conclusion

A system of answer sheet evaluation requires text identification. Human handwritten text identification should be done by classification for a fixed set of answers. The suggested approach is made up of ROI localization and text data classification. YOLOv5 model is used to recognize handwritten 10 labels out of which 4 labels are for MCQs type answers and 6 labels are for one-word answer type questions. The proposed model was trained on self-collected data from the DA-IICT institute containing approx 2200-2500 instances per class, achieving a precision of 93%.

For future endeavours, it is recommended to collect additional training data to further improve the efficiency and accuracy of the model. This should include a diverse range of data samples encompassing different handwritten styles, including messier styles, to ensure successful identification.

The proposed approach can also be extended to identify both handwritten and printed text from answer sheets, followed by applying the Handwritten Text Recognition (HTR) pipeline to overcome the limitation mentioned in Subsection 3.4.2. HTR pipeline will recognize each character of extracted words and will provide the scope of post-processing.

Similarly, this approach can be applied to various domains such as clinical reports, insurance records, and industrial documents. Additionally, exploring other AI techniques and advancements can be considered for future work to enhance the overall performance of the model.

3.5.1 Future Research Direction

Future research can explore methods to enhance the quality of the input images, such as applying preprocessing techniques to reduce noise and artefacts that may

affect the model's accuracy. Additionally, to overcome the limitation of post-processing, future research can investigate the use of optical character recognition (OCR) instead of classification to fetch the text from the answer sheet. OCR can not only identify the text but can also check for spelling and grammar errors, and distinguish whether the text belongs to one of the ten fixed labels or some other word. This can improve the accuracy of the evaluation and reduce the potential for errors. Furthermore, future research can also explore the scalability of the model to accommodate new classes or changes in the types of answers in the answer sheet.

CHAPTER 4

Recognition Approach

The proposed approach for text recognition involves recognizing all answers to the answer sheet. Initially, location of words identified and then the word written by the student will be identified each alphabet character by character. The predicted word then matches with the 10 fixed labels. The set of all characters in the English alphabet is represented as A , while the set of 10 fixed labels is represented as L . The "other" label is represented as O .

A function f maps the set of characters A to a predicted word W , which can be represented as $W=f(A)$. After predicting the word W , we check whether it belongs to the fixed set of labels L . If $W \in L$, we assign the corresponding predicted word as its label. Otherwise, we assign it the label O and it will not be considered for further evaluation. This can be represented algorithmically as follows:

Algorithm 1 Label Assignment Algorithm

```
if  $W \in L$  then
  | label  $\leftarrow W$ 
else
  | label  $\leftarrow O$ 
end
```

The proposed approach is an end-to-end pipeline which includes preprocessing, segmentation, text recognition, and post-processing. The main goal of this approach is to accurately recognize the answers provided by the students in the answer sheet and assign the correct label to each answer. The proposed approach aims to overcome the limitations of the previous approach, such as the inability to handle new classes or changes in the types of answers and the lack of post-processing scope. By recognizing all characters and assigning the "other" label to words that do not belong to the set of fixed labels, the proposed approach aims to improve the accuracy of the classification and reduce errors in the evaluation of

the answer sheet.

4.1 Proposed Pipeline

The proposed Handwritten Text Recognition (HTR) pipeline, as depicted in Fig.1, consists of several stages that contribute to the recognition of words from input images. Explanation of each stage within the pipeline:

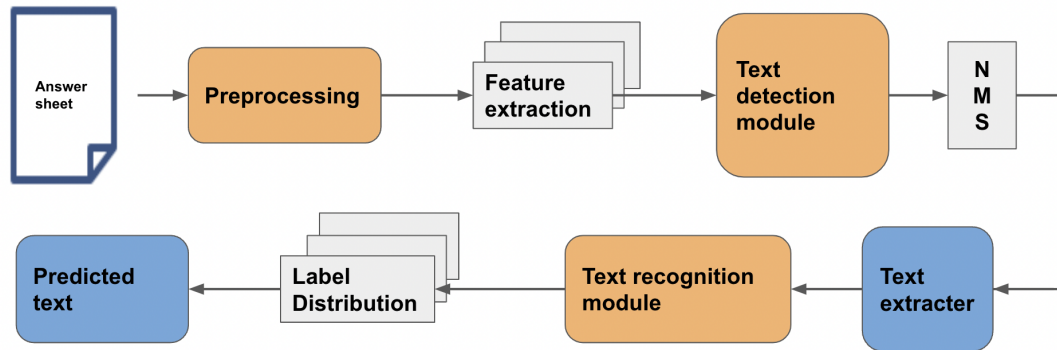


Figure 4.1: HTR Pipeline

Image Acquisition: In the proposed text recognition pipeline, the image acquisition stage involves capturing the student’s answer sheet image using a digital camera or scanner. The student or Exam Supervisor will capture and save the image in a digital format such as JPEG or PNG. It is important to capture high-quality images in good lighting conditions and at a suitable resolution for optimal pipeline performance.

Preprocessing: Before further processing, the acquired student answer sheet images undergo several preprocessing techniques to enhance the quality and reduce noise and distortion in the image.

Segmentation Model: The text segmentation breaks down the preprocessed answer sheet image into individual words. These words are stored as separate images. Accurate segmentation is necessary. Otherwise, words which are not detected will not be part of the recognition model.

Text Recognition: After segmentation, a deep learning model is applied to recognize the text within each segmented region. The model generates a label distribution for each time step or frame, and the predicted labels are decoded into a final text output.

Post-processing: The post-processing step involves combining techniques to correct errors in the predicted text, refine the output, and improve the system's accuracy. This step can include language modelling, spell-checking, and other error correction methods.

Overall, the proposed text recognition pipeline for Answer Sheets ensures the high-quality preprocessing of answer sheet images, accurate segmentation of individual words, and text recognition using a deep learning model. The post-processing step further enhances the accuracy and correctness of the final transcription output.

4.2 Pre-Processing

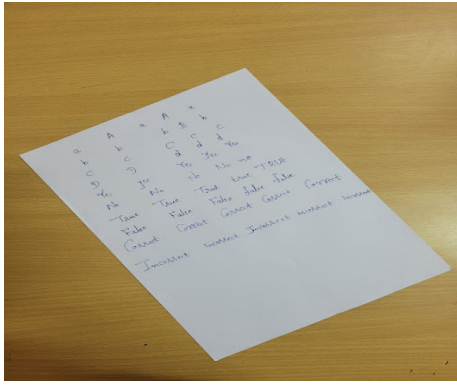
Prior to feeding the input images to the handwriting recognition model, several pre-processing steps were applied to enhance the quality and consistency of the data. The following pre-processing steps were applied:

1. Rescaling: All input images were rescaled to a consistent resolution of (500,500) pixels to ensure consistency in size as shown in Fig. 4.2a.
2. Contour detection: OpenCV's *findContours()* function was used to isolate the area of interest in the input images, which in this case, was the handwriting sample.
3. Canny edge detection: A Canny edge detection algorithm was applied to detect the boundaries of the answer sheet page, which helped in isolating the handwriting sample from the background. Fig. 4.2b shows edge detection on the collected data sample.
4. Perspective transformation: To correct any distortion caused by the camera's angle of view or position, a perspective transformation was applied to the input images. After applying perspective transformation on the data sample present in Fig. 4.2b, the output we got is shown in Fig. 4.2c.
5. Grayscale conversion: The input images were converted to grayscale to reduce computational complexity.
6. Image sharpening: An image sharpening algorithm was applied to improve the contrast and sharpness of the input images, which helped in enhancing

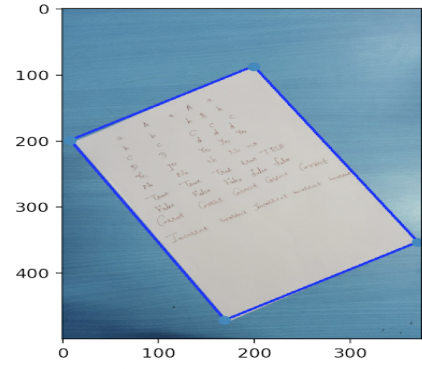
the visibility of the handwriting samples. Fig. 4.2d illustrates the masking of the input image.

7. Adaptive thresholding: Finally, an adaptive thresholding technique was applied to segment the input images into binary regions, which helped in further isolating the handwriting samples from the background shown in Fig. 4.2e.
8. Page Rotation: After completing all the aforementioned preprocessing steps, the images are opened, and the system prompts the user to perform rotations if necessary. This manual intervention is essential because some pre-processed images may have an answersheet that is upside down or scanned horizontally, which could affect the accuracy of subsequent processing.

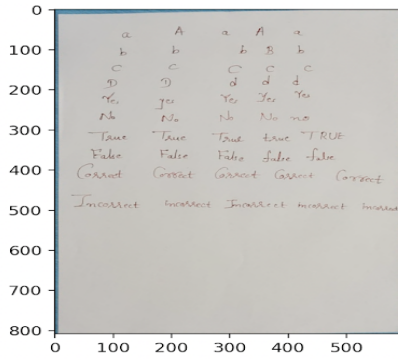
By applying these pre-processing steps, we were able to improve the quality and consistency of the input images, which in turn, helped in improving the accuracy of the handwriting recognition model. A few samples of input were given to the preprocessing pipeline, and the output images are shown in Fig. 4.3.



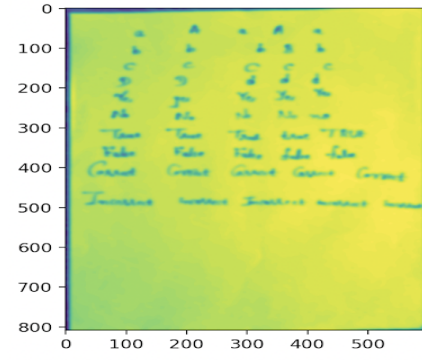
(a) Rescaled Input Image



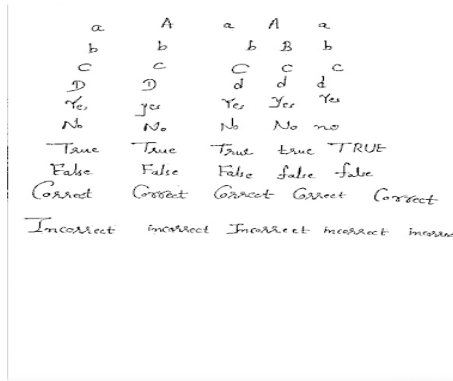
(b) Edge detection in Input Image



(c) Output of Perspective transformation

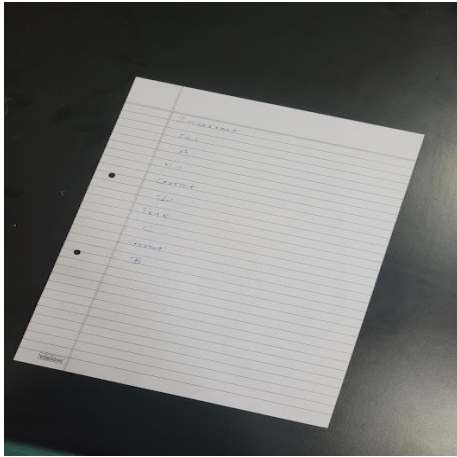


(d) Masked Image

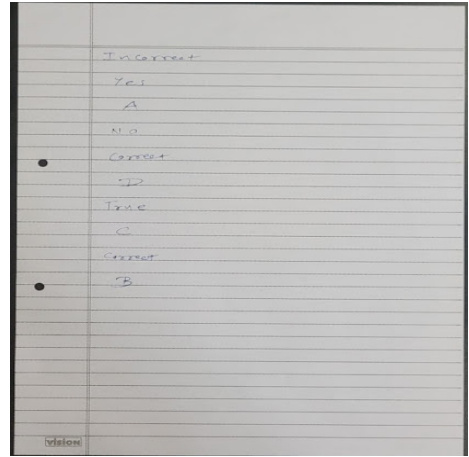


(e) Output of Adaptive threshold

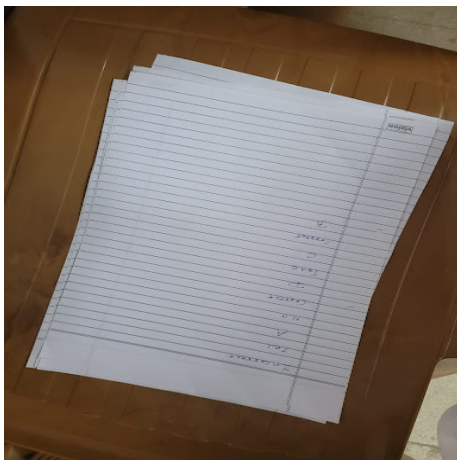
Figure 4.2: Pre-Processing Steps for Segmentation Model



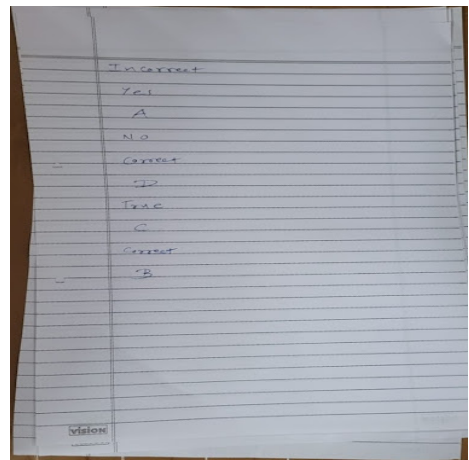
(a) Input Sample Image



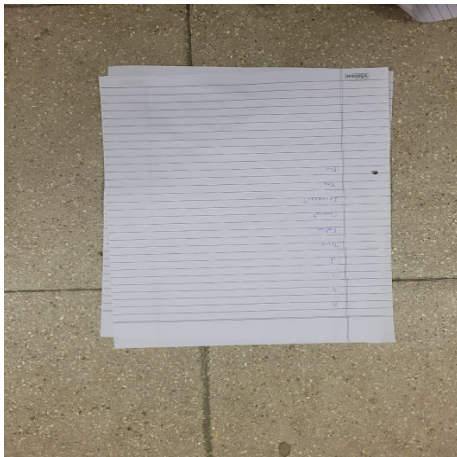
(b) Output of Pre-Processed Image



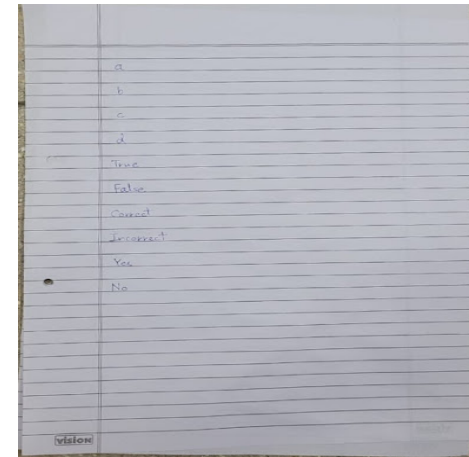
(c) Input Sample Image



(d) Output of Pre-Processed Image



(e) Input Sample Image



(f) Output of Pre-Processed Image

Figure 4.3: Pre-Processing Steps for Recognition Model

4.3 The Segmentation model

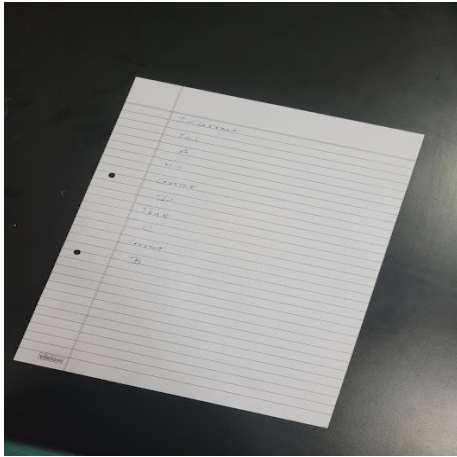
In the text recognition process, the first step is to identify the text region on the page. To achieve this, we used the YOLO (You Only Look Once) algorithm discussed in Section 3.1, which is an object detection system that divides the image into a grid and predicts the bounding boxes and class probabilities for each grid cell.

For this task, we used the DA-IICT students dataset [22], which consists of images of handwritten English words. The dataset was annotated with bounding boxes around the text regions using an annotation file that included the image path, centre x , centre y , height, and width of the bounding box.

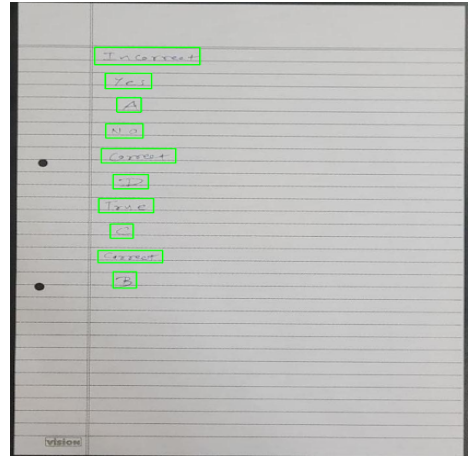
To train the segmentation model, we utilized 468 samples from the dataset. The model was designed to take preprocessed images as inputs and provide bounding boxes around text regions as outputs. We replaced all other labels with a single label named "text" to identify the text regions on the page.

The training process was conducted on images with a size of (640,640), and the dataset was split into an 8:2 training-testing ratio. The model was trained for 425 epochs to optimize its performance.

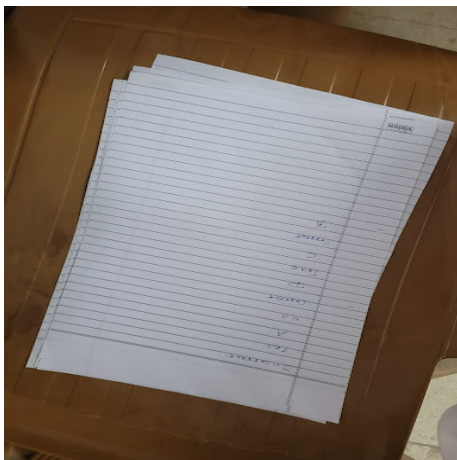
The performance and accuracy of the segmentation model are crucial as it plays a vital role in accurately identifying the text regions on a page. Without a reliable segmentation model, the subsequent text recognition process would be hindered, as the accurate identification of text regions would not be possible. The input and output of the segmentation model are provided in Fig. 4.4



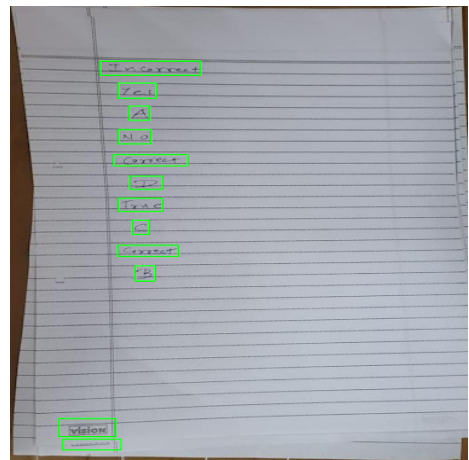
(a) Input Image



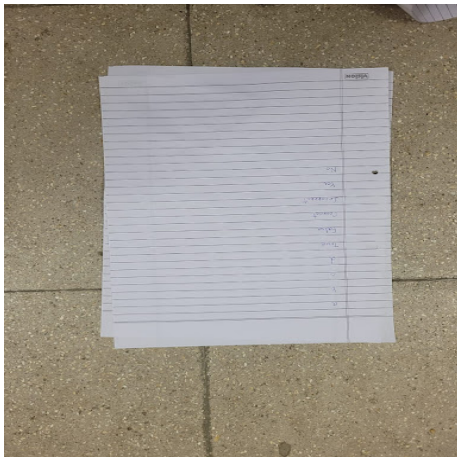
(b) Output of word Segmentation



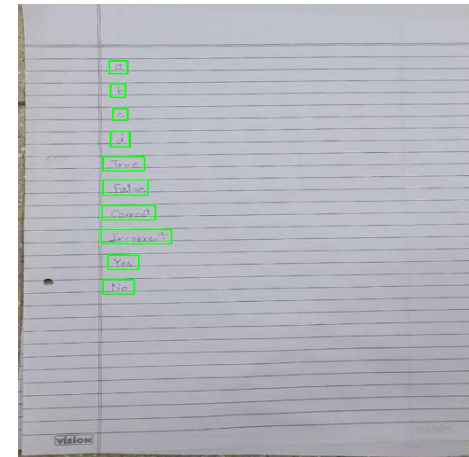
(c) Input Image



(d) Output of word Segmentation



(e) Input Image



(f) Output of word Segmentation

Figure 4.4: Output of Segmentation Model: Segmented image produced by the segmentation model applied to an input image

4.4 The recognition model

This section covers the implementation details of five text recognition models. The implementation process began with CNN and subsequently, additional layers of different RNNs were incorporated. These RNNs are one-to-many types of RNNs. They take the entire image as input and recognize the character sequence as the output. Refer to Section B to get more knowledge about object detection and deep learning models' architecture.

4.4.1 Convolutional Neural Network based model

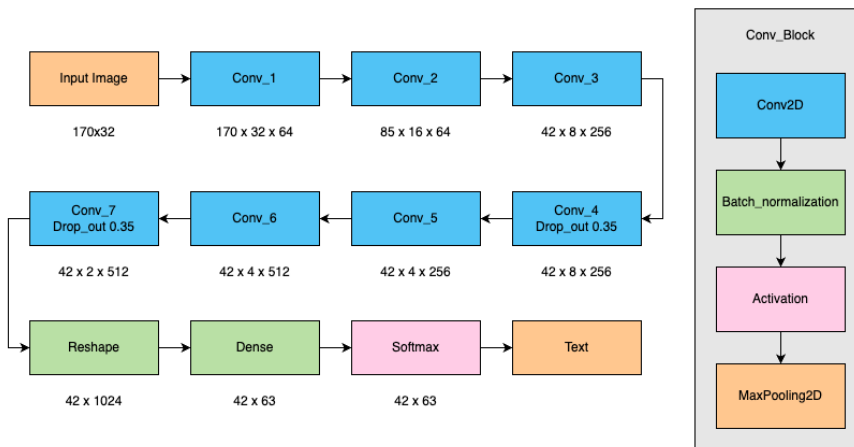


Figure 4.5: Flow diagram of convolution neural network based model

The proposed text recognition model is a Convolutional Neural Network (CNN) architecture based on the VGG-16 [18] model. CNNs are designed to automatically learn features from images using convolutional layers, which apply filters to the input image to extract relevant features.

The Proposed model consists of seven convolution layers, followed by batch normalisation, dropout, activation function and Maxpooling. The Input image has a dimension of (32,170). The input of the model contains the height of the images, the width of the images and the channel. In our implementation, we provided grayscale images representing a single channel in the input. In the conv block, we have used the batch normalisation technique for the input data. It subtracts the batch mean and divides the data with the standard deviation. This helps with the problem of covariate shift, where the distribution of input changes during training, leading to slow convergence.

Rectified Linear Unit (ReLU) activation function is applied to the weighted sum of inputs to a neuron in conv layer. The consecutive operation in the conv layer is Maxpooling; in our experiment, the pool size is (2,2). It extracts the maximum value from the (2,2) grid and stores that single value instead of the four values covered on that grid.

In the convolution layer, the conv filter of (3,3) moves around the image. The jump to the next grid is decided by stride. The stride is (2,2) for the initial two layers, and the next step is the stride. The filter size in the model varies from 64, 128, 256 and 512. To reduce the overfitting of the model dropout layer is used in the convolution layer. A total of 35% neurons were dropped during the training. Also, the early-stop technique involves monitoring validation loss; if validation loss stops improving and starts degrading the models' accuracy, it will stop the training process and store weights that performed best on validation data.

The output produced by the convolutional layer is then flattened, transformed into a one-dimensional vector, and fed into a fully connected layer. This fully connected layer utilizes a softmax activation function to generate the predicted characters as its output. The model uses a VGG-16-based architecture shown in Fig. 4.5, a deep convolutional neural network known for its effectiveness in image recognition tasks.

4.4.2 Convolutional neural networks and long short-term memory networks (CNN+LSTM)

In the third model, we have replaced the Simple RNN layers mentioned in Section ?? with LSTM layers. The LSTM overcomes the limitation of Simple RNNs, such as vanishing gradient and difficulty in capturing long-term dependency. The LSTM architecture has a memory cell with three gate mechanism which helps selectively remember and forget the information. Fig. 4.6 shows the flow of the CNN+LSTM model, which incorporates the combination of Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks. The combination of CNNs and LSTMs has shown good performance in image recognition tasks as mentioned in Section 1.5.

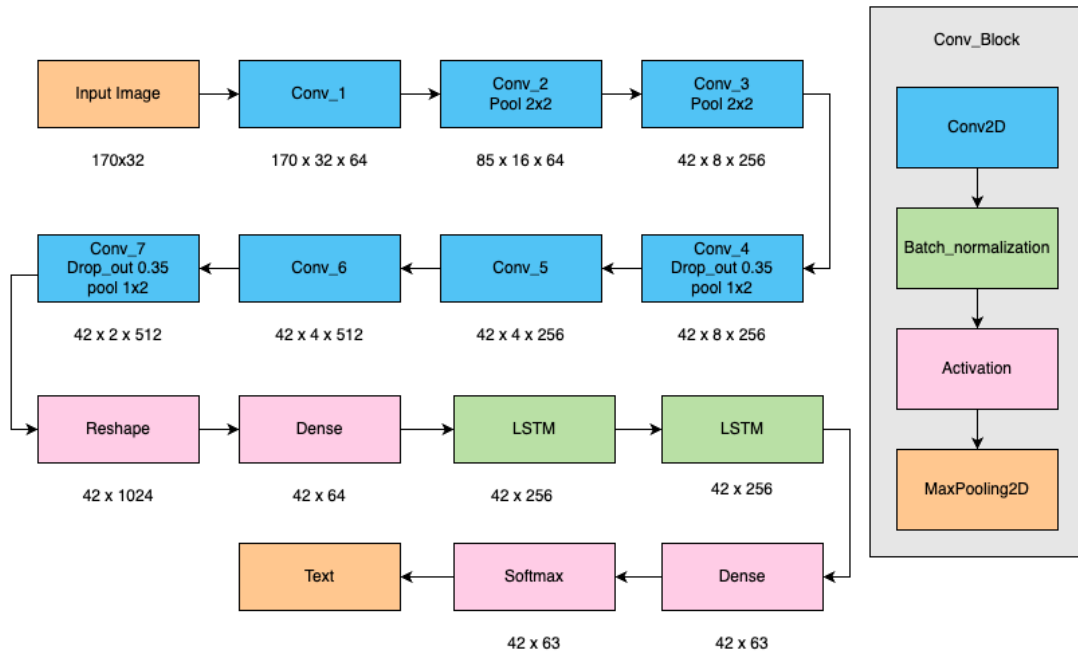


Figure 4.6: Flow diagram of convolutional neural networks and long short-term memory networks (CNN+LSTM) model

The CNN+LSTM model consists of two main parts: the seven convolutional layers and the LSTM layers of 256 hidden units. The input to the model is a 2D image fed to the conv layer, and the output of the conv layer is pooled features that become input to the LSTM model. The output of the LSTM layers is then passed through a fully connected layer with a softmax activation function to generate the predicted output. The CNN layers in the model are the same as mentioned in SubSection 4.4.1.

4.4.3 Convolutional neural network with a bidirectional long short-term memory layer (CNN+BiLSTM)

In the previous subsection, we utilized LSTM layers to take advantage of their memory cells, which allow for the storage of important information over time. However, in the current section, we have made modifications to the architecture by replacing the LSTM layer with BiLSTM layers, modification is shown in Fig. 4.7. These BiLSTM layers consist of both forward and backward LSTM units, which enhance the model’s ability to capture contextual information in both directions.

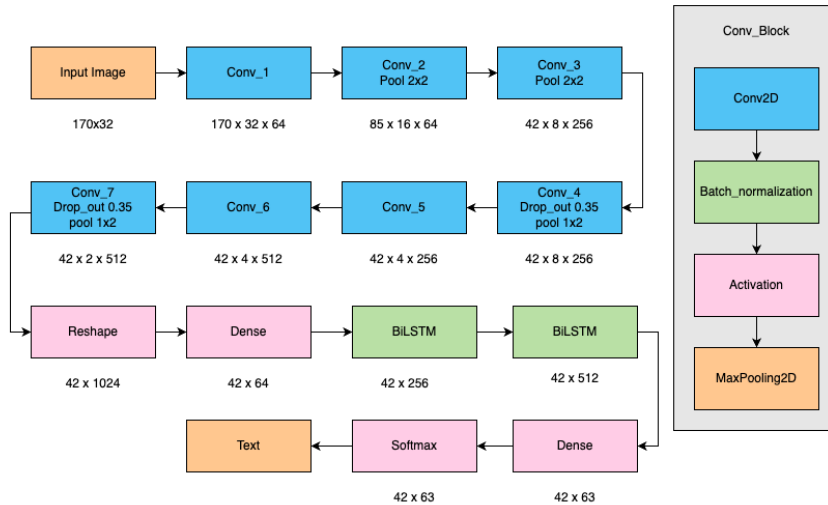


Figure 4.7: Flow diagram of convolutional neural network with a bidirectional long short-term memory (CNN+BiLSTM) model

After the output of the Convolutional (Conv) layer, the data is reshaped and fed into a dense layer. The dense layer processes the input data and produces an intermediate representation. This intermediate representation serves as the input for the BiLSTM layers.

The output of the BiLSTM layers is a label distribution, which represents the probabilities of different labels for each input sequence. These probabilities are then passed through the softmax function, which normalizes them to obtain a probability distribution over all possible labels. From this distribution, the predicted text is generated by selecting the label with the highest probability.

Additionally, in this model implementation, there is a third layer called the transcript layer. The transcript layer employs the Connectionist Temporal Classification (CTC) loss function [27]. CTC loss is used to establish a mapping between the input sequence and the output sequence without the need for aligned training data. It allows for handling variable-length input and output sequences, making it suitable for tasks such as speech recognition and handwriting recognition.

By incorporating the transcript layer with CTC loss, the model learns to align the input sequence with the corresponding output sequence, facilitating accurate text recognition.

4.4.4 Convolutional neural network with bidirectional gated recurrent units (CNN+BiGRU)

The final attempt in the recognition pipeline is shown in Fig. 4.8, we have implemented a CNN+BiGRU model. It is important to note that the CNN layers remain consistent throughout all the models we have developed. However, we have used different types of RNN layers in all the implementations.

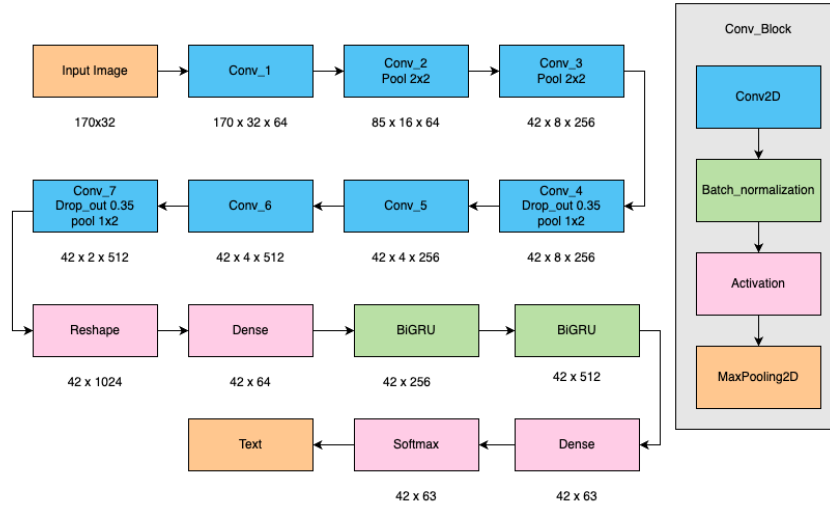


Figure 4.8: Flow diagram of Convolutional neural network with bidirectional gated recurrent units (CNN+BiGRU) model

In this particular model, we have replaced the BiLSTM layers with two layers of BiGRU (Bidirectional Gated Recurrent Unit). While both BiGRU and BiLSTM function similarly in capturing temporal dependencies, the architecture of BiGRU has fewer gates compared to BiLSTM. This leads to a reduction in the number of parameters in the model, making it faster than BiLSTM.

By utilizing the BiGRU layers in this model, we aim to leverage their ability to capture bidirectional contextual information, allowing the model to better understand the sequential patterns within the input data. This, in turn, enhances the accuracy and performance of the text recognition system.

It is worth mentioning that despite the change in the RNN layer architecture from BiLSTM to BiGRU, the overall model implementation remains largely the same as the previous model. The key difference lies in the type of RNN layers used. This enables us to compare and analyze the performance of the two architectures in the context of text recognition.

By applying this modified model, we aim to assess the impact of using BiGRU layers in the recognition pipeline and evaluate their effectiveness in achieving accurate and efficient text recognition.

4.5 Results

This section is divided into two subsections. Section 4.5.1 covers insights from the visual results and Section 4.5.2 covers the analysis of quantitative results.

4.5.1 Visual Results

This section presents the visual results obtained from evaluating five models: CNN, CNN+LSTM, CNN+BiLSTM, and CNN+BiGRU. The visual results provide an intuitive understanding of each model's performance by comparing the predicted text with the ground truth text. The ground truth text was manually created using an annotation tool, as described in Section 2.3.

Below mentioned tables: Table 4.1, Table 4.2, Table 4.3, and 4.4 display the output of each model, showcasing the cropped image of the word, the predicted word, and the ground truth. These visual results offer valuable insights into the strengths and weaknesses of the models. The visual results were analyzed in the following manner.

Table 4.1 showed the poor performance of the CNN model overall. It tended to make incorrect predictions when there were mistakes in the initial characters of the words. These initial errors often led to different word predictions, indicating the model's sensitivity to early misclassifications.




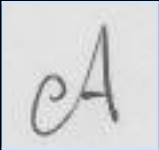
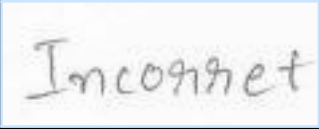

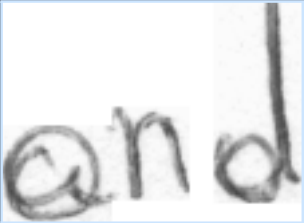
Word Image	Predicted Text	Ground Truth
	c	c
	correct	correct
	yes	yes
	Rd	a
	Incorret	Incorrect
	book	book
	aRnd	and

Table 4.1: Predicted and Ground Truth Text of CNN model

The CNN+LSTM model, which operates as a sequence-to-sequence model, showed potential for predicting the entire word by mapping one sequence to another. However, it initially struggled with mapping the input sequence to the correct character, resulting in inaccurate predictions. As the model progressed through the sequence, it eventually aligned with the correct character sequence, improving the accuracy of its predictions shown in Table 4.2. This suggests that the LSTM model can benefit from enhanced character mapping mechanisms.



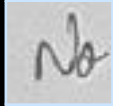

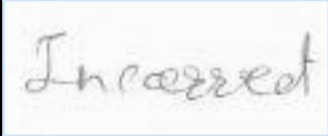
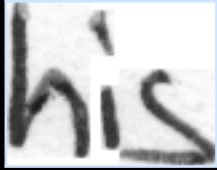

Word Image	Predicted Text	Ground Truth
	correct	correct
	b	d
	ao	no
	false	false
	Incorrect	Incorrect
	his	his
	that	that

Table 4.2: Predicted and Ground Truth Text of CNN+LSTM model

The CNN+BiLSTM model demonstrated better performance compared to the other models. However, it exhibited a consistent issue where the first character of the predicted text was frequently missing. Additionally, there were occasional incorrect predictions for single-character words. Another observation was that the model sometimes misclassified the “false” class. A few sample words and their predicted words are mentioned in Table 4.3.

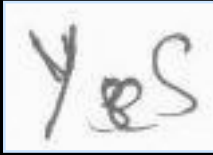



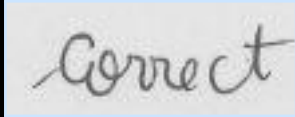


Word Image	Predicted Text	Ground Truth
	yes	yes
	True	True
	correct	correct
	false	false
	correct	correct
	whid	which
	hor	hour

Table 4.3: Predicted and Ground Truth Text of CNN+BiLSTM model

Similar to the CNN+BiLSTM model, the CNN+BiGRU model performed well shown in Table 4.4. However, there were occasional errors in predicting the “false” and “true” classes, such as missing or additional characters. These errors, while relatively minor, indicate room for refinement to achieve more precise predictions.

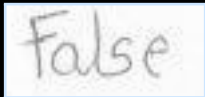




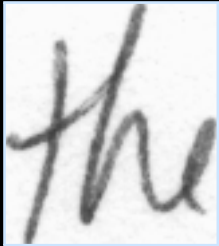
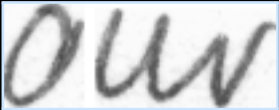
Word Image	Predicted Text	Ground Truth
	false	false
	Incorrect	Incorrect
	correct	correct
	yes	yes
	d	b
	the	the
	our	our

Table 4.4: Predicted and Ground Truth Text of CNN+BiGRU model

Based on the provided insights, it is evident that the CNN+BiLSTM and CNN+BiGRU models outperformed the other models in terms of accuracy and robustness. These models demonstrated more accurate predictions with lower occurrences of wrong predictions and errors in classifying “false” and “true” labels.

The CNN+BiLSTM model showed significant improvement in performance by effectively mapping the input sequence to the correct character sequence, especially after the initial character. This indicates that the model successfully captured the underlying sequential patterns in the data, leading to more accurate word recognition.

Similarly, the CNN+BiGRU model showcased promising results, with only minor errors observed in predicting the “false” and “true” classes. These errors, however, were relatively minimal and did not significantly impact the overall performance of the model.

The better performance of CNN+BiLSTM and CNN+BiGRU can be attributed to their ability to capture both forward and backward dependencies in the input sequence. This capability enables these models to effectively model the contextual information necessary for accurate word recognition.

These findings show the potential of CNN+BiLSTM and CNN+BiGRU models in text recognition tasks, specifically for the requirements of answer sheet evaluation. However, to substantiate these claims and validate the superiority of these models, we have provided supporting evidence in the form of quantitative measures in Section 4.5.2.

In subsequent sections of the thesis, we presented quantitative measures. This analysis will complement the visual results presented earlier and contribute to a comprehensive evaluation of the models’ performance.

4.5.2 Quantitative Analysis of Results

In this section, we present a quantitative analysis of the results obtained from the evaluation of the handwritten word recognition models. The analysis provides insights into the performance of each model by considering quantitative measures such as word accuracy, letter accuracy, and processing time.

Table 4.5 summarizes the performance of the different models in terms of word accuracy, letter accuracy, and processing time for a dataset consisting of 1112 images.

Model	Word Accuracy	Letter Accuracy	Processing Time (s)
CNN	55.57%	71.04%	132.35
CNN+LSTM	44.78%	76.91%	193.02
CNN+Bidirectional LSTM+CTC	83.90%	88.13%	102.45
CNN+Bidirectional GRU+CTC	88.93%	91.68%	100.77

Table 4.5: Comparison of Handwritten Word Recognition Models

The CNN model achieved a word accuracy of 55.57% and a letter accuracy of 71.04%. However, its accuracy was relatively lower compared to the other models. The blue line in Fig 4.9 shows the validation loss did not follow the decreasing trend throughout the training, at some point validation loss increased. It exhibited a processing time of 132.35 seconds.

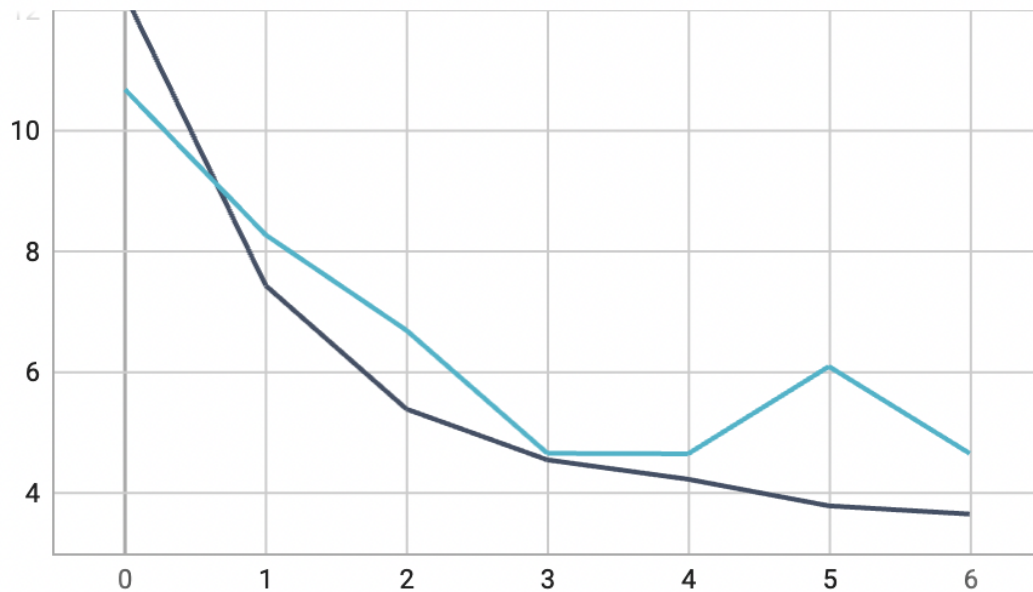


Figure 4.9: The training and validation loss per epoch in CNN

Integrating CNN and LSTM improved the word accuracy to 44.78% and the letter accuracy to 76.91%.the validation loss follows the decreasing trend with training loss shown in Fig. 4.10. However, this model required a longer processing time of 193.02 seconds.

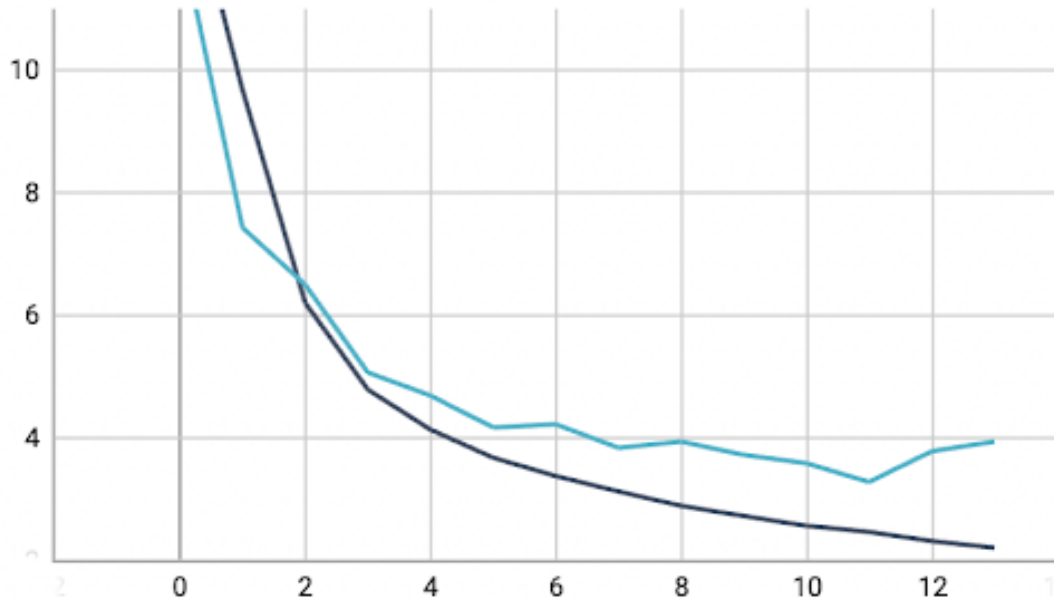


Figure 4.10: The training and validation loss per epoch in CNN+LSTM

The CNN combined with Bidirectional LSTM and CTC demonstrated significant improvements, achieving a word accuracy of 83.90% and a letter accuracy of 88.13%. This model also exhibited a notable reduction in processing time, with 102.45 seconds.

The CNN combined with Bidirectional GRU achieved the highest word accuracy of 88.93% and the highest letter accuracy of 91.68%. It showcased a comparable processing time of 100.77 seconds. This model outperformed all other models in terms of accuracy while maintaining a reasonable processing time.

As shown in Fig. 4.11 and Fig. 4.12, validation Loss and training loss of the BiLSTM model and BiGRU follow the smooth decreasing trend. Also, the quantitative analysis highlights the effectiveness of integrating Bidirectional LSTM and Bidirectional GRU with the CNN architecture with great letter accuracy of 88% and 91% respectively. These models demonstrated superior word and letter accuracy compared to the other models, indicating their capability in handwritten word recognition tasks.

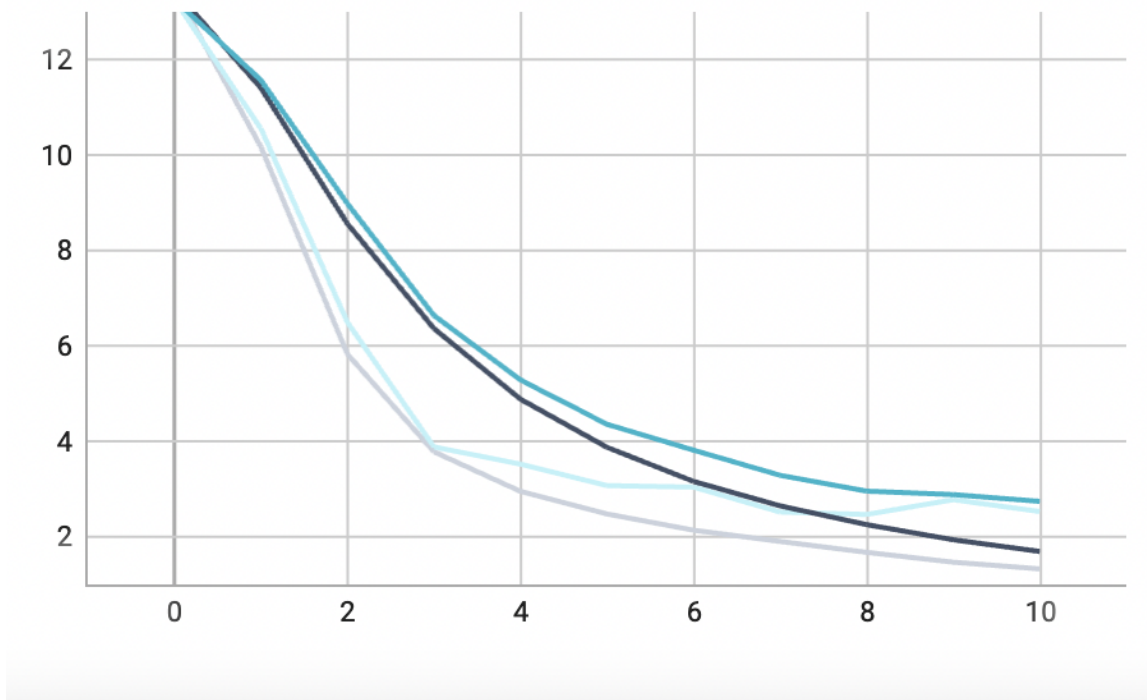


Figure 4.11: The training and validation loss per epoch in CNN+BiLSTM

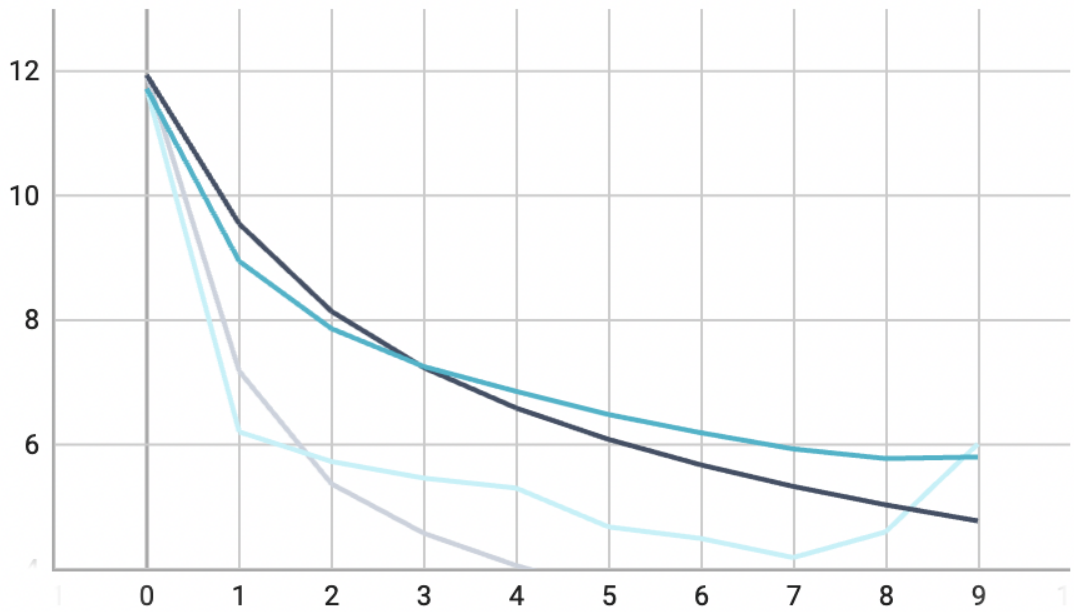


Figure 4.12: The training and validation loss per epoch in CNN+BiGRU

It is important to consider various factors such as computational complexity, scalability, and specific application requirements when selecting the most suitable model. The quantitative analysis presented here, based on the performance met-

rics of word accuracy, letter accuracy, and processing time, serves as a foundation for making informed decisions and further optimizing the models for improved performance.

4.6 Discussion

Our Literature review explored traditional text recognition methods for identifying handwritten words, including line segmentation techniques and full-page documentation recognition methods. We focused on recognizing one-word answers and employed an object detection algorithm to segment words. We used YOLO for the Segmentation task but also considered faster-RNN as an alternative method.

The single Handwritten Text Recognition (HTR) model is insufficient for accurately recognizing handwritten text. We addressed this issue by employing a sequence-to-sequence model to identify the optimal sequence of characters in the output text. Additionally, we compared the performance of a classification model and an autoregressive model and found that while the classification model was faster in its recognition capabilities, the recognition model was superior in identifying each character sequence.

Our analysis indicated that the classification model was under-defined due to its failure to account for answers outside a predetermined set of answers. The recognition model has perfectly overcome all the limitations of the classification approach and provides a large scope of post-processing and provides flexibility to add new answers to the predetermined set answers.

4.7 Structure of question paper and answer sheet

In this section, we have covered general instructions for students, question design and answer sheet information.

- **General instruction for student:**
 - Students need to write their names and roll numbers on both the question paper and answer sheet.
 - Write your answers in order only.
 - Do not leave any answers blank; provide a response for each question.

- For MCQ-type questions, use a, b, c, or d to indicate the correct answer.
 - Do not write the full text of the answer; mention the letter corresponding to the correct choice.
 - For one-word answer questions, write only the correct answer without providing any justifications. A single word answer is sufficient.
- **Question Paper and answer sheet design:**
 - The question paper will include a mixture of MCQs and one-word answer questions. If a question is of the MCQ type, it will be clearly indicated, and four options will be provided within the question.
 - If it is a one-word answer question, there will be three possible options for the answer: yes or no, true or false, or correct or incorrect. Each question will have a clear specification of the type of answer required.
 - The answer sheets provided by the invigilator are A4-sized ruled pages.

4.8 Scope and limitations of the system

- **Data Imbalance:** The dataset used for training the system contains a high proportion of alphabet data (99.95%) and a limited amount of digit data (0.05%). As a result, the system faces challenges in recognizing and accurately evaluating numeric responses.
- **Inability to Map Question Numbers to Answers:** Due to the lack of digit data, the system cannot effectively map the question numbers to corresponding answers. Consequently, it relies on the students to write their answers in order, and leaving blank answers unmarked could lead to difficulties in identifying which questions were left unanswered.
- **Inefficiency of the system:** The inefficiency of the current automated answer sheet evaluation system lies in its 91% accuracy in identifying MCQs and one-word answers. While the system utilizes post-processing techniques to match a fixed set of answers best, it still falls short of achieving 100% accuracy. Such inaccuracies can potentially impact students' futures adversely, and it is imperative to ensure fairness and accuracy in the assessment process. The responsibility for this 9% error in the system's evaluation lies with multiple stakeholders:

- **Development Team:** They must continuously refine and improve the algorithms and methodologies used to enhance the system’s accuracy.
- **Authority and Decision Makers:** The educational authorities and decision-makers who adopt and implement the automated evaluation system must also take responsibility. They should ensure proper vetting of the system and its capabilities before its widespread adoption.

4.9 Conclusion and Future work

In our study, we aimed to evaluate various text recognition methods and propose an object detection-based approach for the implicit region of interest (ROI) localization on answer sheets. The first part of our approach involved a classification model that achieved an impressive accuracy rate of 93%. However, we acknowledge the limitation of this model in terms of scalability and its inability to identify words that do not belong to the fixed set of answers.

To overcome this limitation, we incorporated autoregressive text recognition models in the second part of our approach. These models demonstrated the ability to recognize all characters with a letter accuracy of 91%. This approach provided us with the opportunity for post-processing, allowing us to refine the extracted text further.

By combining both approaches, we achieved accurate ROI localization and recognition of the text on answer sheets. The classification model provided efficient detection of predefined labels, while the text recognition models enhanced the recognition of handwritten words and offered flexibility in handling different answers.

Our proposed approach showcases the effectiveness of combining classification and autoregressive text recognition models. This hybrid approach overcomes the limitations of the individual models and contributes to more accurate and comprehensive answer sheet evaluation.

In addition to its application in multiple-choice and one-word answer questions, the pipeline we have developed holds promise for automating the evaluation of Short Answer Questions (SAQs) and Descriptive Answer Questions (DAQs) through the integration of Natural Language Processing (NLP) models.

The first part of our pipeline, which involves object detection and classification

models, can still be leveraged for SAQs and DAQs. By localizing the regions of interest on the answer sheets, we can extract the handwritten text. However, since SAQs and DAQs do not have predefined correct answers, we need to incorporate NLP models to understand the context and evaluate the responses.

We also recommend post-processing, considering cases where students write multiple answers in a single line, cut the letter in the middle, or do not write any answer to improve the accuracy of the answer sheet evaluation. Furthermore, we suggest using a dataset of text written on ruled paper to provide more realistic scenarios for prediction.

Finally, Other Proposal can be, The model should identify ten predefined labels and other labels that are not desired answers as unknown labels, totalling 10+1 classes. Consider a conditional language model which only understands 10 fixed label and mark all other text as unknown.

References

- [1] Unified district information system for education plus, 2020-21. UDISE+.
- [2] C. C. Aggarwal. *Neural Networks and Deep Learning*. Springer, New York, 2018.
- [3] S. Arora, D. Bhattacharjee, M. Nasipuri, D. Basu, and M. Kundu. Combining multiple feature extraction techniques for handwritten devnagari character recognition. 05 2010.
- [4] Y. Baek, B. Lee, D. Han, S. Yun, and H. Lee. Character region awareness for text detection. pages 9357–9366, 06 2019.
- [5] S. Bharadia, P. Sinha, and A. Kaul. Answer evaluation using machine learning. 03 2018.
- [6] T. Bluche, J. Louradour, and R. Messina. Scan, attend and read: End-to-end handwritten paragraph recognition with mdlstm attention. pages 1050–1055, 11 2017.
- [7] K. Chen. Introduction to lstm and gru. <https://kinder-chen.medium.com/introduction-to-lstm-and-gru-2fe6f50e3ef2>, 2021. Accessed: 05 2023.
- [8] C. K. Ch'ng and C. S. Chan. Total-text: A comprehensive dataset for scene text detection and recognition. pages 935–942, 11 2017.
- [9] J. Chung and T. Delteil. A computationally efficient pipeline approach to full page offline handwritten text recognition. pages 35–40, 09 2019.
- [10] R. Elanwar. *RULE-BASED ALGORITHMS FOR HANDWRITTEN CHARACTER RECOGNITION*. PhD thesis, 02 2007.
- [11] E. Grosicki, M. Carré, E. Geoffrois, and F. Prêteux. Rimes evaluation campaign for handwritten mail processing. 10 2006.

- [12] M. Horvat, L. Jelečević, and G. Gledec. A comparative study of yolov5 models performance for image localization and classification. 09 2022.
- [13] J. Jeong. The most intuitive and easiest guide for recurrent neural network. <https://towardsdatascience.com/the-most-intuitive-and-easiest-guide-for-recurrent-neural-network-873c29da73> 2019. Accessed: 05 2023.
- [14] G. Jocher, A. Stoken, A. Chaurasia, J. Borovec, NanoCode012, TaoXie, Y. Kwon, K. Michael, L. Changyu, J. Fang, A. V, Laughing, tkianai, yxNONG, P. Skalski, A. Hogan, J. Nadar, imyhxy, L. Mammana, AlexWang1900, C. Fati, D. Montes, J. Hajek, L. Diaconu, M. T. Minh, Marc, albinxavi, fatih, oleg, and wanghaoyang0106. ultralytics/yolov5: v6.0 - YOLOv5n 'Nano' models, Roboflow integration, TensorFlow export, OpenCV DNN support, Oct. 2021.
- [15] A. K and H. Krishnappa. Kannada handwritten document recognition using convolutional neural network. pages 299–301, 12 2018.
- [16] A. Khandelwal, P. Choudhury, R. Sarkar, S. Basu, M. Nasipuri, and N. Das. Text line segmentation for unconstrained handwritten document images using neighborhood connected component analysis. pages 369–374, 12 2009.
- [17] U.-V. Marti and H. Bunke. The iam-database: An english sentence database for offline handwriting recognition. *International Journal on Document Analysis and Recognition*, 5:39–46, 11 2002.
- [18] S. Mascarenhas and M. Agarwal. A comparison between vgg16, vgg19 and resnet50 architecture frameworks for image classification. In *2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON)*, volume 1, pages 96–99, 2021.
- [19] M. Murdock, S. Reid, B. Hamilton, and J. Reese. Icdar 2015 competition on text line detection in historical documents. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1171–1175, 2015.
- [20] K. O'Shea and R. Nash. An introduction to convolutional neural networks. *ArXiv e-prints*, 11 2015.
- [21] M. A. Rahaman and H. Mahmud. Automated evaluation of handwritten answer script using deep learning approach. *Transactions on Engineering and Computing Sciences*, 10(4), Aug. 2022.

- [22] K. Ratanghayara. Answer sheet dataset, 2023.
- [23] J. Redmon and A. Farhadi. Yolov3: An incremental improvement, 2018.
- [24] S. Sahay. India’s national education budget for 2023-24. *British Council’s blog*, 02 2023.
- [25] S. Singh and S. Karayev. *Full Page Handwriting Recognition via Image to Sequence Extraction*, pages 55–69. 09 2021.
- [26] N. Stamatopoulos, B. Gatos, G. Louloudis, U. Pal, and A. Alaei. Icdar 2013 handwriting segmentation contest. In *2013 12th International Conference on Document Analysis and Recognition*, pages 1402–1406, 2013.
- [27] R. Takashima, S. Li, and H. Kawai. Ctc loss function with a unit-level ambiguity penalty. pages 5909–5913, 04 2018.
- [28] A. Veit, T. Matera, L. Neumann, J. Matas, and S. Belongie. Coco-text: Dataset and benchmark for text detection and recognition in natural images. 01 2016.
- [29] K. Wu, H. Fu, and W. Li. Handwriting text-line detection and recognition in answer sheet composition with few labeled data. pages 129–132, 10 2020.
- [30] C. Yao, X. Bai, W. Liu, Y. Ma, and Z. Tu. Detecting texts of arbitrary orientations in natural images. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1083–1090, 2012.
- [31] R. Zhang, Y. Zhou, Q. Jiang, Q. Song, N. Li, K. Zhou, L. Wang, D. Wang, M. Liao, M. Yang, X. Bai, B. Shi, D. Karatzas, S. Lu, and C. V. Jawahar. Icdar 2019 robust reading challenge on reading chinese text on signboard. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1577–1581, 2019.
- [32] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang. East: An efficient and accurate scene text detector. 04 2017.
- [33] M. Zimmermann and H. Bunke. Automatic segmentation of the iam off-line database for handwritten english text. volume 4, pages 35 – 39 vol.4, 02 2002.

CHAPTER A

APPENDIX: Initial Attempts for Segmentation models

In this section, we present the initial attempts made for utilizing the CRAFT (Character-Region Awareness For Text Detection) model for handwritten text segmentation. The CRAFT model is renowned for its effectiveness in segmenting text in various contexts, making it a promising choice for our word recognition task. Here, we outline the steps taken during our initial exploration and experimentation with the CRAFT model.

A.1 Dataset

For training the segmentation model, we utilized six widely recognized and comprehensive datasets, namely ICDAR2013 [26], ICDAR2015 [19], ICDAR2017, MSRA-TD500 [30], TotalText [8], and CTW-1500 [4]. These datasets are renowned in the field of text detection and segmentation and provide a diverse range of text samples with varying characteristics and challenges.

1. ICDAR2013: The ICDAR2013 dataset comprises a collection of natural scene images with text annotations. It encompasses various real-world scenarios and encompasses a wide range of text styles, sizes, orientations, and background complexities.
2. ICDAR2015: Similar to ICDAR2013, the ICDAR2015 dataset contains scene images with text annotations. It was specifically curated for the robust reading challenge, focusing on challenging text conditions such as low resolution, perspective distortions, and occlusions.
3. ICDAR2017: The ICDAR2017 dataset is a more recent addition to the ICDAR series and features a broader range of text instances in various languages. It

includes scene text, born-digital text, and handwritten text samples, offering a comprehensive evaluation platform for text detection and recognition methods.

4. MSRA-TD500: The MSRA-TD500 dataset is a benchmark dataset for detecting and recognizing text in natural scenes. It consists of 500 high-resolution images with annotated text regions, capturing diverse text layouts and styles.
5. TotalText: The TotalText dataset is specifically designed to address challenges in detecting and segmenting curved text instances. It contains 1555 images with text annotations, including horizontal, multi-oriented, and curved text samples.
6. CTW-1500: The CTW-1500 dataset is another dataset that focuses on curved text detection. It consists of 1500 images with annotated curved text instances, allowing for the evaluation of algorithms under various curved text scenarios.

To evaluate the performance of our trained segmentation model, we utilized the DA-IICT dataset as our test dataset. The DA-IICT dataset comprises a diverse collection of handwritten word samples, collected from multiple sources and encompassing various writing styles, sizes, and word structures. This dataset was specifically chosen to assess the generalization and robustness of our segmentation model on handwritten text recognition tasks.

By training our model on these six comprehensive datasets and evaluating its performance on the DA-IICT dataset, we aimed to ensure that our segmentation model possesses the ability to handle a wide range of text variations and achieve accurate segmentation results in real-world scenarios.

The dataset collected from the DA-IICT institute needs to be localized to generate the ground truth of the dataset. Data localization is a method to create bounding boxes around the text area and label them. The online tool is used for labelling the data in pascal_voc format, which stores pixel values of x_{min} , y_{min} , x_{max} , and y_{max} . For the yolov5 format of localization, we used roboflow, which stores width, height, x_{centre} and y_{centre} for all labels and stores them into a text file.

A.2 Architecture

CRAFT architecture is Fully convolutional network architecture based on VGG-16. Without batch normalization is adopted as our backbone. Model has skip connections in the decoding part, which is similar to U-net in that it aggregates low-level features. The final output has two channels as score maps: the region score and the affinity score. The network architecture is schematically illustrated in A.1. Re-

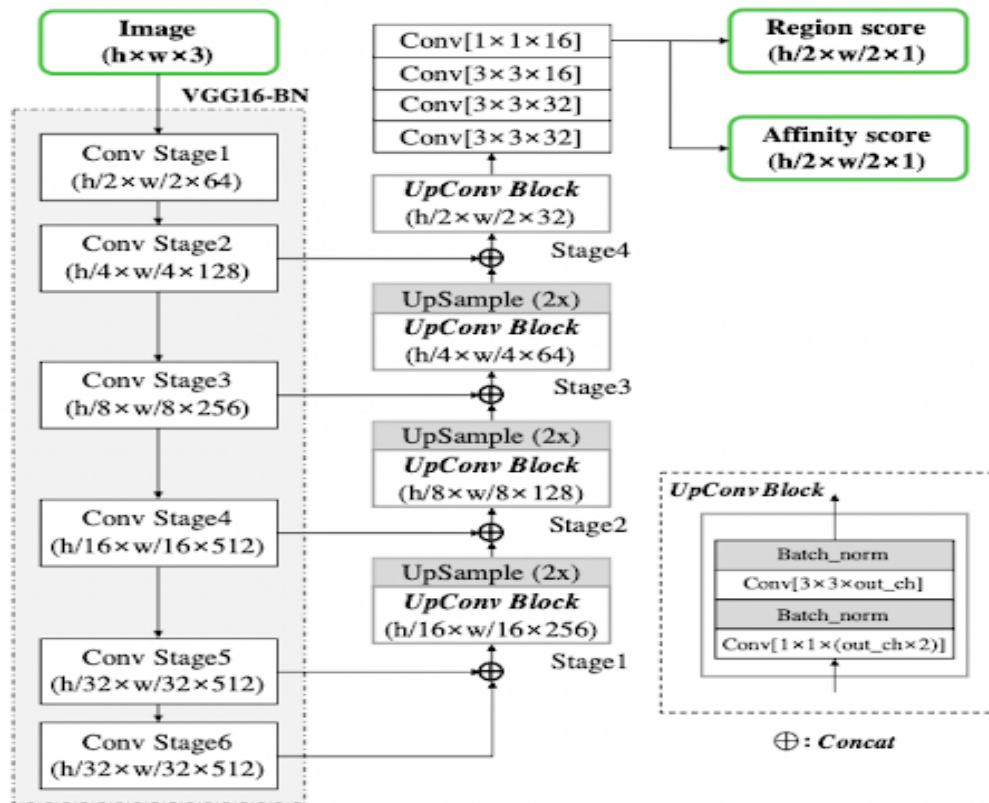


Figure A.1: Schematic illustration of CRAFT network architecture[4]

gion score is used to localize individual characters in images and the affinity score is used to group each character into a single instance.

A.3 Discussion and Result

As we can see in Figs. A.2 and A.3, The CRAFT model can segment words well, but it cannot identify single characters precisely. This model is trained on a scene text dataset, and we have tested it in handwritten text images. The reason is that handwritten text and scene text are different in nature. The reason is that hand-

written text and scene text are different in nature. To achieve better results model requires handwritten text image data in training.

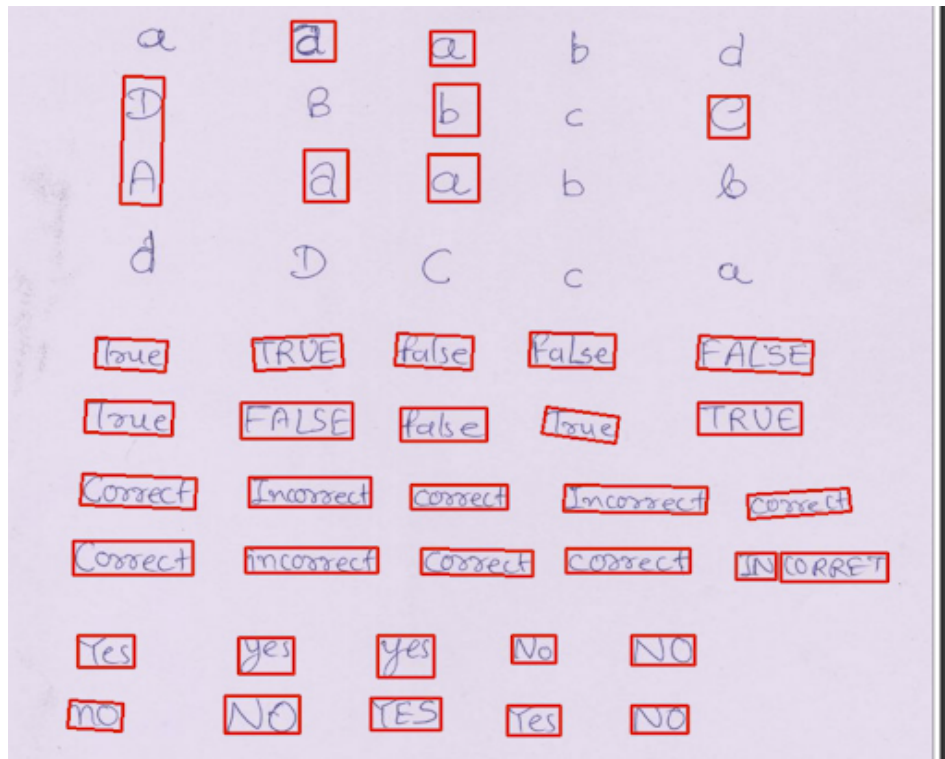


Figure A.2: Result of CRAFT model for all answers

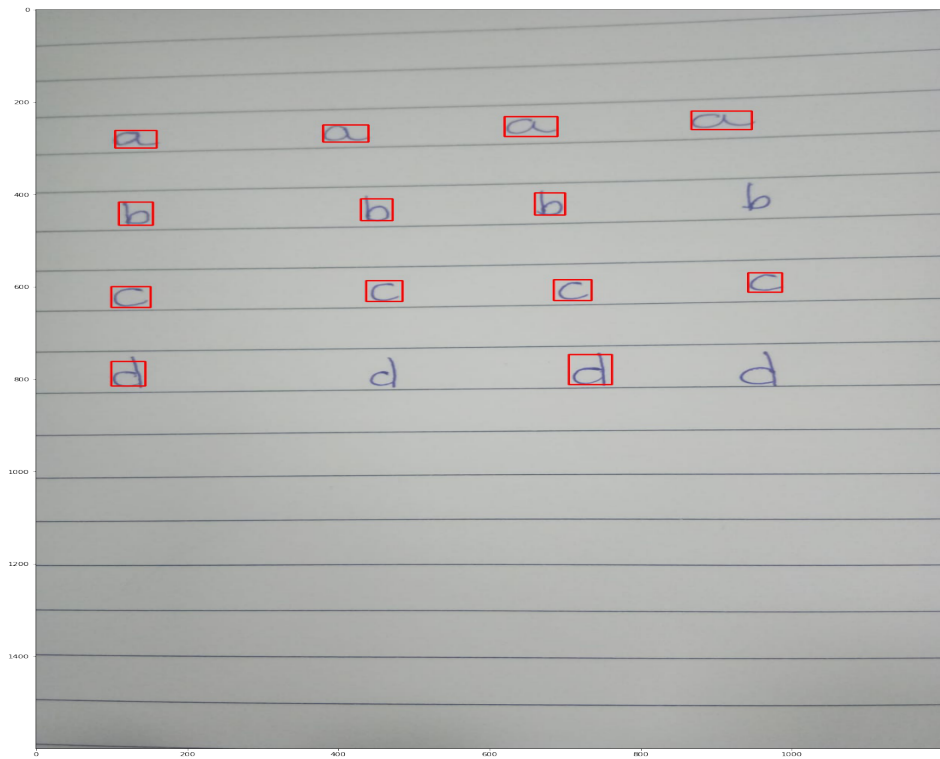


Figure A.3: Result of CRAFT model for a single character

CHAPTER B

Background Knowledge and Model design

The section contains basic concepts and terminologies required to understand the deep learning models.

B.1 Neural Network

Neural networks are a layered representation of neurons (perceptrons), or multi-layer perceptrons, that resembles the human brain. It creates an adaptive system that computers use to learn from their mistakes and improve continuously.

A neural network comprises multiple layers, each with its specific function. The complexity of the network determines the number of layers, leading to its alternative name, the multi-layer perceptron. The three main layers in a neural network are the input layer, hidden layer(s), and output layer. The input layer receives information from the external environment, and its nodes process, analyze, or categorize the data before passing it to the subsequent layer. The hidden layer(s) then analyze the output from the preceding layer, further process it, and transmit it to subsequent layers. The output layer, which can consist of either a single or multiple nodes, produces the final outcome. For instance, in a binary classification problem, the output layer would have a single node providing a result of either 0 or 1. However, in a multi-class scenario, the output layer might contain more than one node.

B.2 Computer Vision And Object Detection

Computer vision is the ability to extract information from images and videos, and with the use of neural networks, it can also distinguish and recognize images similar to humans.

Object detection is a set of computer vision techniques to help detect the object and its class. Object detection contains two tasks: object detection, also known as object localization and object recognition. Object localization refers to identifying the location of one or more objects in images and drawing a bounding box around their extent. Object recognition involves predicting the class of one object in an image.

- **Image Classification:** This stage involves predicting the type or class of an object in an input image. The input is typically a photograph or image containing a single object, and the output is a class label that represents the predicted object category. This can be achieved by mapping the input image to one or more integers that correspond to specific class labels.
- **Object Localization:** In this stage, the goal is to detect the presence of objects in an image and provide their precise location using bounding boxes. The input is an image that may contain one or more objects, and the output consists of one or more bounding boxes defined by coordinates (point, width, and height). The bounding boxes indicate the regions in the image where the objects are located.
- **Object Detection:** This stage combines both object localization and image classification. It involves detecting the presence of objects in an image, providing bounding boxes to localize the objects, and assigning class labels to the detected objects. The input is an image that may contain multiple objects, and the output includes one or more bounding boxes along with the corresponding class labels for each detected object.

B.3 Convolution Neural Network

The convolutional neural network [20] architecture is used for extracting features from the image dataset. All the operations performed in CNN architecture are mentioned below.

- **Convolution Operation:** Convolution refers to the straightforward process of applying a filter to an input, which leads to activation. When the same filter is applied to the entire input image, it generates a map of activation known as a feature map. This feature map reveals the locations and strength of a detected feature within the input, typically an image. The convolution

operation offers two key advantages: parameter sharing and sparsity of connection. Parameter sharing implies that a feature detector identified in one part of the image is likely to be beneficial in other areas as well. This concept allows the same set of filter weights to be applied across different regions of the input, reducing the number of parameters needed to be learned. The sparsity of connection means that in each layer, the output value relies only on a small subset of inputs rather than the entire image size. This characteristic ensures that each output is influenced by only a limited portion of the input, resulting in computational efficiency and reduced memory requirements.

- **Activation Function:** It identifies the output of the Neural Network, like yes or no. It maps the resulting values between 0 to 1 or -1 to 1. The activation function has two types: linear activation function and non-linear activation function. The activation function activates the node if it hits the activation threshold. Non-linear activation functions are the most used activation function. It makes it easy for the model to adapt the data as most real-world data is non-linearly separable.
- **Pooling:** Pooling layers consolidate the features learned by the convolutional layer feature map. It helps to reduce overfitting by the time of training of the model by compressing or generalizing the features in the feature map. These layers are very simple because they often use the maximum or average value of the input to downsample the data.
- **Fully Connected Layers:** Fully connected layers in neural networks refer to the layers where all the inputs from one layer are connected to every activation unit in the subsequent layers. These layers, typically located towards the end of the network, serve the purpose of consolidating the information extracted by the preceding layers to generate the final output.

B.4 Recurrent Neural Networks

Recurrent Neural Networks are powerful models to handle sequential data and also have the ability to retain memory and ability identify dependency over time. In this section, we have discussed different types of RNNs.

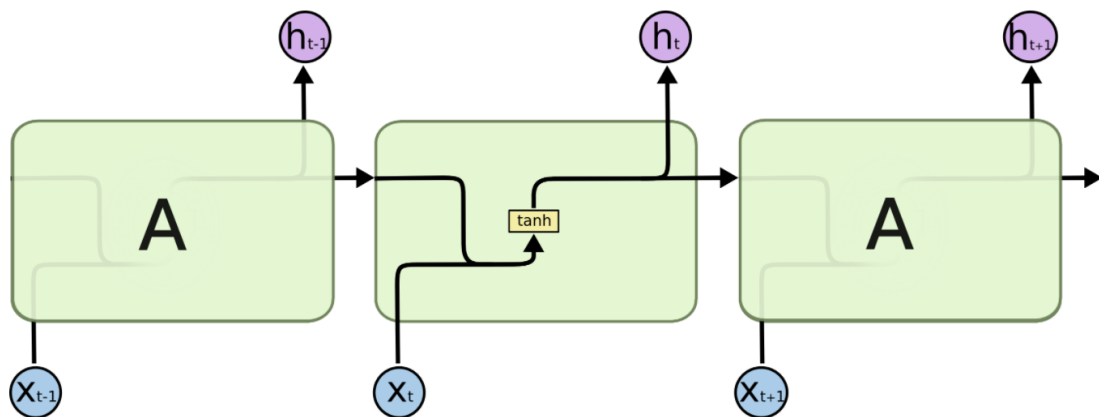
B.4.1 Simple Recurrent Neural Network

The basic architecture of an RNN in Fig. B.1 consists of three layers: an input layer (X_t), a hidden layer (h_t), and an output layer. The input layer receives the input at each time step (t), the hidden layer maintains a hidden state that summarizes the past inputs, and the output layer produces a prediction based on the current input and the hidden state.

At each time step, the input is first fed into the input layer, which applies a linear transformation to the input vector and passes it through a non-linear activation function (\tanh). The resulting output is then fed into the hidden layer (h_t), where it is combined with the previous hidden state (h_{t-1}) through another linear transformation and activation function. This produces a new hidden state (h_{t+1}) that summarizes the information from the past inputs.

The hidden state (h_t) is then passed through another linear transformation and activation function to produce the output at the current time step (t). This output is used to make a prediction, and the process is repeated for the next input at the next time step.

One of the key features of an RNN is that it maintains a memory of the past inputs through the hidden state. This allows it to capture long-term dependencies and context in sequential data, making it well-suited for natural language processing and time series analysis tasks.



The repeating module in a standard RNN contains a single layer.

Figure B.1: Architecture of a Simple Recurrent Neural Network (RNN) [13]

B.4.2 Long Short Term Memory

The LSTM architecture is composed of a series of repeating modules, or cells, that are connected to each other in a chain-like structure shown in Fig. B.2. Each cell contains a set of learnable parameters that allow it to remember or forget information over time selectively.

At the heart of each LSTM cell is a memory cell, which is responsible for storing the system's current state. The memory cell is controlled by three gates: the input gate, the forget gate, and the output gate.

The input gate determines which information from the current input should be stored in the memory cell. It takes the input (X) at the current time step (t), and the previous output as inputs, and the activation function (σ) produces a vector of numbers between 0 and 1, which represent the importance of different elements of the input. The element-wise multiplication between the input vector and the output of the input gate produces a candidate value that could be added to the memory cell.

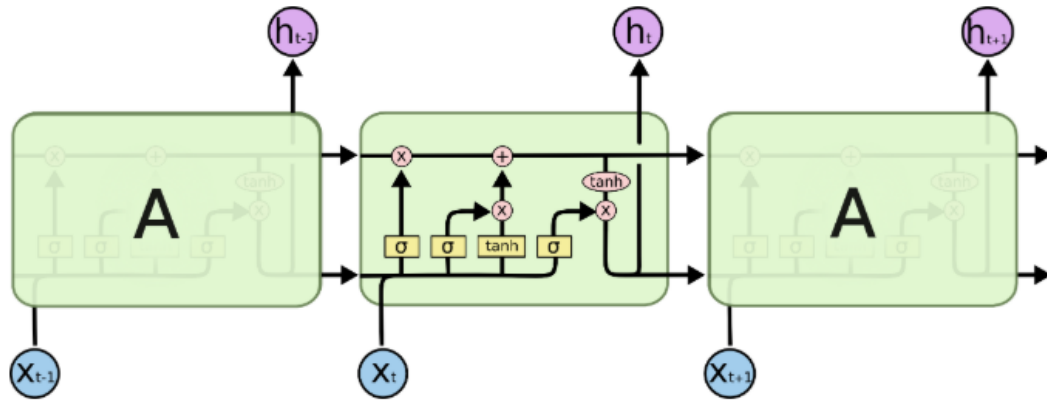
The forget gate determines which information from the previous memory cell should be discarded. It takes the previous output and the current input as inputs (X_t) and produces another vector of numbers between 0 and 1, representing the importance of different memory cell elements. The element-wise multiplication between the forget gate output and the previous memory cell produces a modified memory cell, where some information from the previous memory cell has been selectively forgotten.

The output gate determines which information from the current memory cell should be outputted. It takes the previous output and the current input as inputs and produces another vector of numbers between 0 and 1, representing the importance of different memory cell elements. The element-wise multiplication between the output gate output and the current memory cell produces the output of the current cell.

To summarize, the three gates in an LSTM cell collaborate to perform distinct functions of adding, selectively forgetting, and outputting information to and from the memory cell. The input gate determines which information should be added to the memory cell, the forget gate decides what information should be disregarded from the previous memory cell, and the output gate determines which information should be retrieved from the current memory cell. This enables the

LSTM to effectively store and retrieve information over extended periods, making it a potent tool for handling sequential data with long-term dependencies.

The architecture of the LSTM model is depicted in Figure 5.6. Additionally, there is another type of recurrent neural network called the Gated Recurrent Unit (GRU). GRU is similar to LSTM but has a reduced number of parameters. It was developed to tackle the issue of vanishing and exploding gradients, which can occur during the training of RNNs.



The repeating module in an LSTM contains four interacting layers.

Figure B.2: Architecture of LSTM (Long Short-Term Memory) model [7]

B.4.3 Gated Recurrent Unit

GRU is a type of recurrent neural network (RNN) similar to LSTM but with fewer parameters. It was introduced to address the problem of vanishing and exploding gradients in traditional RNNs and has shown to be effective in various natural language processing tasks.

Fig. B.3 shows the architecture of a GRU network is similar to LSTM, but it has only two gates: a reset gate and an update gate. These gates control the flow of information in and out of the hidden state, representing the sequence's current context.

In an LSTM cell, the reset gate plays a crucial role in deciding how much of the previous hidden state should be discarded or forgotten. Conversely, the update gate determines the portion of the new input that should be incorporated to update the hidden state.

The computation of the hidden state in a GRU is dependent on the current in-

put and the previous hidden state. The reset gate regulates the degree to which the previous hidden state influences the calculation of the new hidden state. Meanwhile, the update gate determines the amount of importance assigned to the new input for updating the hidden state. The hidden state is computed by combining the reset gate, the update gate, and the input at the current time step.

The equations governing the behavior of a GRU can be quite complex, but the basic idea is that the gates are controlled by sigmoid activation functions, which can be thought of as switches that can turn the flow of information on or off. The values of these switches are determined by learned parameters that are updated during training.

In summary, the architecture of a GRU consists of a hidden state that is updated based on the input at the current time step and the previous hidden state. The update and reset gates control the flow of information in and out of the hidden state. Sigmoid activation functions control the gates, and the values of these switches are learned during training. The GRU is designed to avoid the vanishing and exploding gradients problem in traditional RNNs.

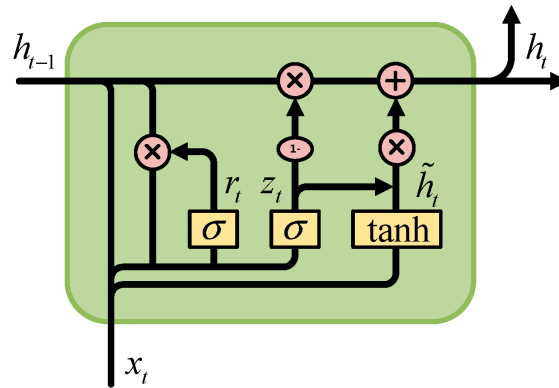


Figure B.3: Architecture of Gated Recurrent Unit (GRU) [7]