Project Report On

Medical Prescription Management System

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

in

Computer Science And Engineering

By

Madhav Mannathazath Menon (U2103129)
Issac Mathew Jaimon (U2103104)
Mathew Paul (U2103132)
Jerin Varghese Tom (U2103111)

Under the guidance of

Ms. Sangeetha Jamal
Assistant Professor

Department of Computer Science and Engineering
Rajagiri School of Engineering & Technology (Autonomous)
(Parent University: APJ Abdul Kalam Technological University)
Rajagiri Valley, Kakkanad, Kochi, 682039

April 2025

CERTIFICATE

This is to certify that the project report entitled ***Medical Prescription Management System*** *is a bonafide record of the work done by* ***Madhav Mannathazath Menon (U2103129), Issac Mathew Jaimon (U2103104), Mathew Paul (U2103132), Jerin Varghese Tom (U2103111)*** *, submitted to the Rajagiri School of Engineering & Technology (RSET) (Autonomous) in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (B. Tech.) in Computer Science and Engineering during the academic year 2024-2025.*

Ms. Sangeetha Jamal
Project Guide
Assistant Professor
Dept. of CSE
RSET

Ms. Anu Maria Joykutty
Project Co-ordinator
Assistant Professor
Dept. of CSE
RSET

Dr. Preetha K.G.
Professor and HOD
Dept. of CSE
RSET

ACKNOWLEDGMENT

We wish to express our sincere gratitude towards **Fr. Dr. Jaison Paul Mulerikkal CMI**, Principal, RSET, and **Dr. Preetha K. G.** , HOD and Professor, Department of Computer Science and Engineering for providing us with the opportunity to undertake our project, Medical Prescription Management System.

We are highly indebted to our project coordinators, **Ms. Anu Maria Joykutty**, Assistant Professor, Department of Computer Science and Engineering, **Dr. Jisha G.**, Associate Professor, Department of Computer Science and Engineering for their valuable support.

It is indeed our pleasure and a moment of satisfaction for us to express my sincere gratitude to our project guide **Ms. Sangeetha Jamal**, Assistant Professor, Department of Computer Science and Engineering for her patience and all the priceless advice and wisdom she has shared with us.

Last but not the least, We would like to express my sincere gratitude towards all other teachers and friends for their continuous support and constructive ideas.

Madhav Mannathazath Menon
Issac Mathew Jaimon
Mathew Paul
Jerin Varghese Tom

Abstract

The accurate interpretation of handwritten prescriptions remains a significant challenge in healthcare, particularly in regions where electronic medical records have not been widely adopted. Variations in doctors' handwriting, use of abbreviations, and inconsistent formats can lead to misinterpretations, resulting in medication errors, adverse drug interactions, and compromised patient safety. This project aims to develop an AI-powered system that leverages Optical Character Recognition (OCR) technologies to efficiently and accurately recognize handwritten prescriptions. In cases where the OCR model fails, an integrated chatbot will analyze patient-reported symptoms to recommend appropriate medications, ensuring continuous and reliable patient care.

The system will feature a user-friendly interface for easy prescription uploads and symptom input, along with a secure database to store and retrieve prescription data. By automating the process of prescription interpretation, this project seeks to reduce human error, streamline the medication dispensing process, and improve healthcare efficiency. Ultimately, the system will enhance patient safety, provide an alternative solution when prescription recognition fails, and support healthcare professionals in delivering more accurate and timely treatment recommendations.

Contents

List of Abbreviations

1. AI - Artificial Intelligence
2. OCR - Optical Character Recognition
3. NLP - Natural Language Processing
4. CRNN - Convolutional Recurrent Neural Network
5. CNN - Convolutional Neural Network
6. RNN - Recurrent Neural Network
7. CTC - Connectionist Temporal Classification
8. OTC - Over-The-Counter

List of Figures

Chapter 1

Introduction

The healthcare industry also faces a considerable challenge in understanding handwritten prescriptions, especially within regions where the electronic medical records are not so prevalent. This is because a large number of prescriptions are illegible due to differences in doctor's handwriting and the use of abbreviations that may be adopted by different practitioners, leading to potential medication errors, miscommunication, and even safety risks. An efficient solution to these issues is, therefore, paramount.

1.1 Background

Handwritten prescriptions are common, especially in regions without widespread electronic medical records. These prescriptions are often difficult to decipher due to variations in handwriting, abbreviations, and inconsistent formats, leading to medication errors and patient safety risks. No widely adopted solution currently exists to reliably convert handwritten prescriptions into a machine-readable format.

Addressing prescription readability is crucial to prevent medication errors, incorrect dosages, and overlooked drug interactions, which can compromise patient safety and burden healthcare professionals. An AI-powered system for interpreting handwritten prescriptions can improve accuracy, efficiency, and safety in medication dispensation.

1.2 Problem Definition

In healthcare settings, many still use handwritten medical prescriptions, especially in areas where electronic record-keeping has not yet been widely used. The readability of such hand-written notes can be significantly inconsistent, and thus several problems are posed that directly influence patient safety and the efficacy of treatment. Perhaps the most concerning issue is that the doctors' often illegible handwriting leads to pharmacists misinterpreting the prescribed safety in medication dispensation. drugs which may lead to medication errors.

In addressing the aforementioned problems, this project focuses on designing an AI-powered system that is capable of accurately reading and interpreting handwritten prescriptions. The solution applied will be through the CRNN model with the support of OCR technology. The idea of the CRNN model will recognize handwriting in prescription images by extracting medication names, dosages, and other special instructions for medications. Automation of prescription reading by the system would help significantly minimize human errors.

1.3 Scope and Motivation

The scope of this project includes the design of an artificial intelligence-powered system capable of identifying and interpreting handwritten medical prescriptions. The system would incorporate an Optical Character Recognition (OCR) model in converting handwritten data to digital format and a Natural Language Processing (NLP) based chatbot that processes symptoms and medicines prescriptions if the recognition of the prescription fails. In addition, the project will also include a user-friendly interface that will allow health care professionals and patients to upload prescriptions and interact with the chatbot. The system is intended to be fully integrated with the existing healthcare workflow, providing accurate and efficient prescription processing. Moreover, it will include a secure database for the storage and retrieval of prescription data. It arises from the grave necessity to mitigate prescription-related errors attributed to handwritten, unreadable prescriptions.

Critical to note are serious health complications in terms of drug interaction, as well as improper dosing that threaten patients' safety. Utilizing the latest technology involving artificial intelligence, this project shall address prescription interpretations by enhancing precision and dependability towards improved overall health care. Besides, the system would relieve healthcare providers of the workload from reading prescriptions to concentrate on other crucial work. Finally, the project would achieve its objective by offering a reliable solution to assist healthcare providers and patients in reaching improved health results.

1.4 Objectives

Developing the basis of an efficient system, there is a robust AI-powered OCR model that should be able to recognize and understand handwritten medical prescriptions, whether from image or text input. To accompany this, the NLP-based chatbot is integrated to identify user-reported symptoms and provide an appropriate prescription whenever the recognition system fails. A user-friendly web and mobile interface will be developed to easily upload prescription images and interact with the chatbot. Furthermore, the stored scanned prescriptions will be retrieved along with their identified text, facilitating easy retrieval and management by health care providers, using a secured relational database. Overall, the proposed system seeks to enhance efficiency in healthcare provision, while promoting safe delivery, particularly in reducing manual prescription interpretation times and potential medicine dispensing mistakes.

1.5 Challenges

The main challenges of this project are the accurate recognition of different handwriting styles and medical abbreviations in handwritten prescriptions, which can vary significantly among doctors. Moreover, integrating a reliable chatbot for symptom analysis and medication recommendation is difficult to ensure precise and contextually appropriate responses. Ensuring data security and seamless integration with existing healthcare systems also poses significant technical and regulatory challenges.

1.6 Assumptions

The model assumes that most doctors' handwriting can be processed accurately, ensuring effective recognition. Clear, high-quality images of prescriptions will be provided to facilitate accurate OCR processing. Essential details, such as medication names and dosages, are expected to be clearly handwritten and included in the dataset. For accurate medication recommendations, users must clearly describe their symptoms to the chatbot. Additionally, a complete and up-to-date database of over-the-counter (OTC) medications and their uses will be available to support the chatbot's functionality.

1.7 Organization of the Report

The report is organized as follows: Chapter 1 introduces the problem of illegible handwritten medical prescriptions, a persistent issue in healthcare that leads to medication errors, delays in treatment, and potential risks to patient safety. This chapter discusses the importance of addressing these challenges to improve healthcare outcomes and highlights the role of technology in trans- forming traditional workflows. The chapter proposes the aim of the project: it would develop an AI-driven system based on OCR technology for deciphering handwritten prescriptions and a chatbot with NLP for providing symptom-based guidance. It positions this as a move towards the automation of prescription handling and getting patients on to proper care within the right amount of time and diminishing the burden on healthcare providers by way of reduced administration.

In Chapter 2, the modular design of the Medical Prescription Management System is explained by breaking down the project into several constituent modules, each responsible for a distinct aspect of the system's design and functionality. In this chapter, how each module contributes to the development of a comprehensive AI-driven solution to interpret the written medical prescription and provide symptom-based help is emphasized.

Chapter 3 presents a detailed Literature Survey that is the basis for the development of the Medical Prescription Management System. This chapter delves into various research papers and studies that help understand the core technologies and methodologies used in the system, such as Optical Character Recognition (OCR), deep learning architectures, chatbots, and advanced data structures. Each section discusses specific studies, their methodologies, results, and relevance to the project.

Chapter 4 delves into the practical aspects of the project by presenting the step-by-step implementation of the system. It begins by detailing the development environment, software tools, and frameworks utilized in the project. The chapter outlines the architecture of the application and describes how each core module was developed to meet the system requirements. Special attention is given to the user interface design, with annotated screenshots illustrating key functionalities such as user input, medicine scheduling, reminders, and notifications. The chapter also presents the testing strategies employed, including both functional and non-functional testing, and discusses the outcomes. Furthermore, it includes a summary of the results obtained, highlighting how the implemented features align with the project's objectives, as well as any challenges encountered and how they were addressed.

Chapter 5 summarizes the key findings and accomplishments of the project. It reflects on the initial goals set during the planning phase and evaluates the extent to which they have been achieved through the implemented solution. The effectiveness and usability of the system are briefly discussed, along with insights gained during the development process. In addition, the chapter outlines potential areas for enhancement and expansion, suggesting improvements in functionality, scalability, and user experience. These recommendations aim to guide future development and research that could build upon the current work to create a more robust and feature-rich system.

Chapter 2

Literature Survey

2.1 Handwritten Text Recognition Using Deep Learning with TensorFlow, S. Y. Manchala, J. Kinthali, K. Kotha, K. S. Kumar, and J. Jayalaxmi, 2020 [1]

Overview

This paper [1] focuses on developing a deep learning-based handwritten text recognition system using a hybrid model of Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Connectionist Temporal Classification (CTC).

Model Architecture:

Convolutional Neural Networks (CNNs) are deep neural networks designed primarily for the handling of structured, grid-like data, which could be anything ranging from images. Their layered structure makes them an excellent fit for feature extraction and can be easily applied in any task: be it classification or object detection in images or just handwriting recognition. CNNs process images by employing convolutional filters to recognize certain patterns within them, like edges, textures, and shapes. Through multiple layers of convolutions and pooling, CNNs hierarchically learn to identify increasingly complex features, starting from basic structures like lines and progressing to more abstract representations such as characters or objects. This makes CNNs indispensable in OCR systems for extracting features from handwritten or printed text. Re- current neural networks are created to handle sequential data, where the order and context of inputs play a crucial role.

Unlike feedforward neural networks, **Recurrent Neural Networks (RNN)** have a hidden state acting as a memory, retaining some information from previous inputs. This ability of modeling temporal dependencies makes them fit for tasks such as language modeling, speech recognition, and handwriting recognition. For instance, in an OCR system, RNN can process sequences of images features to predict the order of characters in a handwritten word. However, standard RNNs suffer from the problem of long-term dependencies, which more advanced variants like Long Short-Term Memory (LSTM) networks effectively overcome.

Connectionist Temporal Classification (CTC) is a loss function specifically designed for sequence-to-sequence tasks where input and output lengths may not match, such as in handwriting or speech recognition. Unlike traditional loss functions that require such an alignment, CTC, however, accepts flexible alignment of input data to labels by introducing the concept of a blank token, which allows this model to be able to map long and complicated input sequences into shorter output sequences, such as words or characters, in an image of hand-written frames. In OCR systems, CTC is essential for decoding the predictions of CNNs and RNNs to yield readable text without the requirement for manually aligned labels. This feature makes CTC a backbone for end-to-end training in contemporary text recognition systems.

Datasets:

The IAM Dataset, widely used for handwritten text recognition, contains labeled sequences of handwritten text in English.

Methodology

1. Neural Network Design:

Convolutional Neural Networks (CNNs) are in charge of extracting the features from the images by a combination of convolutional and pooling layers. The network identifies patterns such as edges, shapes, and textures that are vital for recognizing text. It gradually processes the image through five layers, each of which reduces its dimensions but retains the critical features. This reduction is done in such a way that it is computationally efficient while maintaining the fundamental characteristics of the image, so that the system can focus on the most important visual elements of the handwriting. Recurrent Neural Networks (RNNs) are designed to process sequential data, making them ideal for capturing spatial dependencies in handwriting.

In this system, **Long Short-Term Memory (LSTM)**, an advanced variant of RNN, is used to handle long text sequences effectively. LSTMs overcome the disadvantages of regular RNNs in that they are able to retain context over longer input sequences so that the earlier and later parts of the text are properly connected. This ability is critical to comprehend the flow of characters in handwritten words and sentences.

Connectionist Temporal Classification (CTC) s critical in aligning and decoding the character sequences from the processed input. Unlike traditional methods that require explicit segmentation of input images, CTC eliminates the need for predefined character boundaries. It maps the sequence of extracted features to the corresponding character sequence, even when the lengths of the input image sequence and the output text differ. This makes CTC particularly well-suited for handling the variability and ambiguity inherent in handwritten text.

2. Data Preprocessing:

First, the images are resized to a fixed dimension of 128x32 pixels so that the size is uniform throughout the dataset and suitable for the input requirement of the neural network. Standardizing resizing ensures images of different sizes will be processed similarly by the model. Furthermore, the images are normalized to grayscale, reducing input complexity as they depend solely on the intensity levels instead of color. This simplification improves the model's ability to detect and learn relevant features that are not affected by irrelevant color variations. To further strengthen the robustness and generalization of the model, data augmentation techniques are used. These include random resizing and positioning of text in the image. Augmentation artificially increases the diversity of the dataset by simulating variations in handwriting or text layout, such as shifts in position, slight scaling, or other distortions. This helps the model become resilient to real-world variations, improving its performance on unseen data.

3. Training:

IAM dataset is used for training. The model outputs probabilities for character sequences, which are then refined using the CTC loss function.

4. Spell Check:

Post-recognition, a spell-checking module ensures syntactical correctness by suggesting corrections for possible errors.

Key Results

The system achieved high recognition accuracy for individual characters and shorter texts but struggles with cursive and densely packed writing. Suggests further optimization through larger input sizes and improved data augmentation.

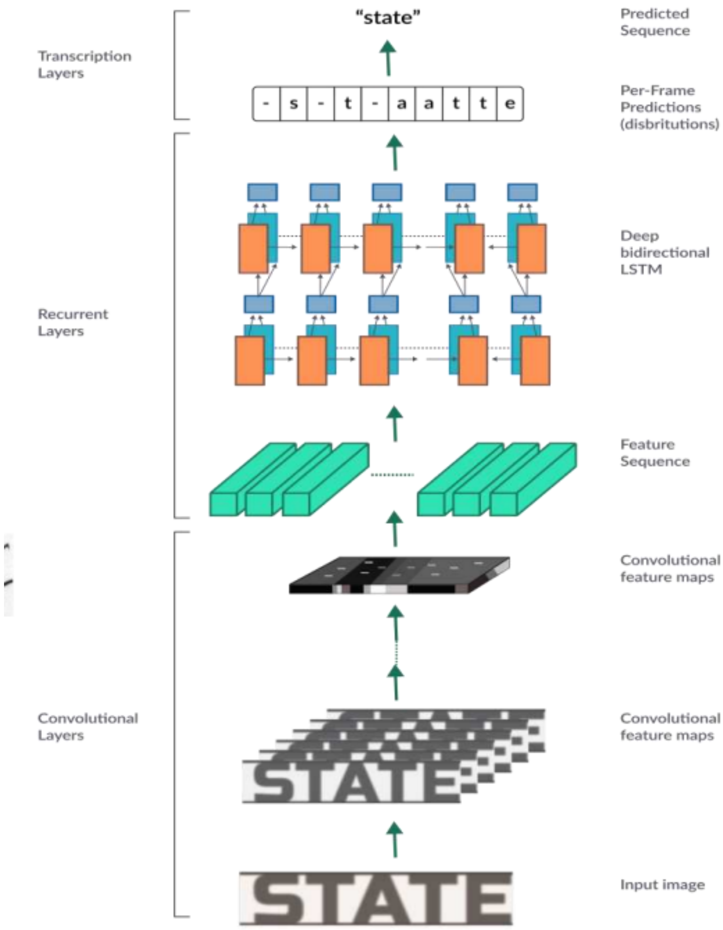


Fig. 8: Architecture of proposed network

Figure 2.1: Proposed Architecture of CNN BiLSTM Model for Handwritten Text Recognition

2.2 Medical Prescription Recognition Using Machine Learning, E. Hassan, H. Tarek, M. Hazem, S. Bahnacy, L. Shaheen, and W. H. Elashmwai, 2021 [2]

Overview

This[2] paper aims to develop a system that digitizes handwritten medical prescriptions by leveraging a combination of Convolutional Neural Networks (CNNs) and Optical Character Recognition (OCR). The goal is to assist pharmacists and patients in accurately interpreting prescriptions, thus reducing medication errors caused by illegible handwriting.

Model Architecture

Image Preprocessing Module:

This module preprocesses raw prescription images to improve their quality and extract relevant regions of interest. Noise, including smudges, shadows, or other artifacts, is removed using techniques such as Gaussian blur or morphological operations to make the input cleaner. The image is then segmented into regions of interest, which include the header (doctor's information), body (medicines and dosages), and footer (signatures or instructions). Segmentation allows the system to focus on specific text areas, improving recognition accuracy.

CNN-Based Feature Extraction:

In this step, the **Convolutional Neural Networks** extract meaningful patterns from the segmented regions. These patterns include visual features such as strokes, curves, and textures that correspond to handwritten characters or prescription-specific text. The CNN identifies and encodes these features into numerical representations, known as feature maps, used to differentiate between prescription sections, such as drug names, dosages, and instructions. This hierarchical extraction allows the system to handle complex handwriting styles and layouts effectively.

OCR Postprocessing:

This module refines and validates the predictions after the initial recognition. Using traditional OCR libraries, such as Tesseract, the system cross-checks and corrects errors to achieve higher accuracy. For instance, it could validate predicted drug names against a medical dictionary or database. It also resolves common errors, such as misinterpreting similar-looking characters (e.g., "0" and "O"). The output is formatted into a structured, readable text, ready for further processing or storage.

Architecture Workflow

Input Layer:

This process begins with an image of a raw prescription, which remains the input to the system. It could be a handwritten or printed text in varying qualities, orientations, and layouts. The layer of input processes the image and standardizes it by resizing and normalizing to ensure compatibility with the model and the consistency carried forward into subsequent operations.

CNN Layers:

These layers use hierarchical feature extraction where the analysis is done at every level: pat-tern detection such as strokes, curves, and shape recognition. Then the CNN incrementally refines its under- standing by taking progressively more abstract feature maps from convolutional layers. Then features are more and more selected based on predefined prescription-specific components such as names of medication, dosages and information about patients and finally pooling of features reduces and retains relevant features eliminating much of computation.

Classification Layer:

The features are then passed to the classification layer, which interprets them to produce text predictions. This layer maps the learned features to corresponding text out- puts, such as letters, words, or symbols, using a predefined vocabulary. It generates an initial transcription of the prescription based on the patterns identified by the CNN.

OCR Refinement:

To be more accurate, the system uses OCR refinement techniques. It cross-checks the preliminary predictions against the existing medical dictionaries or drug databases and corrects the errors with increased reliability. For example, it clarifies ambiguity such as the differentiation between the similar appearing characters, such as ”1” and ”l”. The refined output is an accurate transcription formatted and ready for further use in medical systems or databases.

Methodology

1. Preprocessing:

Image Normalization:

To be more accurate, the system uses OCR refinement techniques. It cross-checks the preliminary predictions against the existing medical dictionaries or drug databases and corrects the errors with increased reliability. For example, it clarifies ambiguity such as the differentiation between the similar appearing characters, such as ”1” and ”l”. The refined output is an accurate transcription formatted and ready for further use in medical systems or databases.

Noise Reduction:

To be more accurate, the system uses OCR refinement techniques. It cross-checks the preliminary predictions against the existing medical dictionaries or drug databases and corrects the errors with increased reliability. For example, it clarifies ambiguity such as the differentiation between the similar appearing characters, such as ”1” and ”l”. The refined output is an accurate transcription formatted and ready for further use in medical systems or databases.

Segmentation:

The normalized and denoised image is divided into three distinct regions of interest (ROI):

- Header: Includes the name of the doctor, clinic, and date.
- Main Prescription Body: This part of the prescription includes important information regarding medications, dosages, and instructions.

- Footer: It contains contact information, signatures, or additional instructions.

This segmentation process allows the system to process each section separately, allowing it to apply specific recognition techniques based on the type of information in each region. It also helps improve the overall recognition accuracy by isolating text from irrelevant areas.

2. Processing:

Feature Extraction:

The system uses CNN layers to analyze and extract meaningful patterns from the middle part of prescriptions, which is the most critical region containing information about medicine names and dosages. CNNs are adept at identifying structural details such as curves, edges, and textures that correspond to hand-written or printed characters. These layers process the image in a hierarchical manner, from low-level features such as strokes and lines up to high-level representations of letters and words. This targeted feature extraction ensures the model focuses on the vital information necessary for accurate recognition of medications and dosages.

Classification:

Once the features are extracted, the system uses the CNN model to classify the identified patterns into corresponding medicine names. Training is done using a dataset of prescriptions collected from various hospitals, thus exposing the model to a wide variety of handwriting styles, formats, and regional conventions. The classification layer maps the learned features to specific medicine names, leveraging the model's ability to generalize across diverse handwriting patterns. This step is crucial for building a reliable system capable of interpreting the critical components of prescriptions accurately.

3. Postprocessing:

- OCR is applied to predictions with low confidence (below 50%) to enhance recognition. Results are matched against a database of known medicines to identify the most probable matches.

Key Results

To increase the accuracy of recognition, OCR refinement is added to predictions where the confidence level is less than 50%. This increases the output reliability even in tough conditions, such as confused handwriting or ambiguous characters. The OCR module performs a second level of analysis on the uncertain predictions by using traditional Optical Character Recognition (OCR) techniques. These techniques specialize in character recognition, and they really do well handling edge cases for which neural networks might fail to perform. This result from the OCR process will then be cross-checked against a database of medicines known.

That database serves as a reference allowing the system to identify the closest matches based on contextual similarity and spelling proximity. For example, in the case where a prediction has a strong likelihood of being incorrect due to bad handwriting, the system cross-checks it against the database and comes up with the closest matching name of medicine. Cross-checking makes the whole system highly reliable and less error-prone during critical cases, such as transcription of prescriptions.

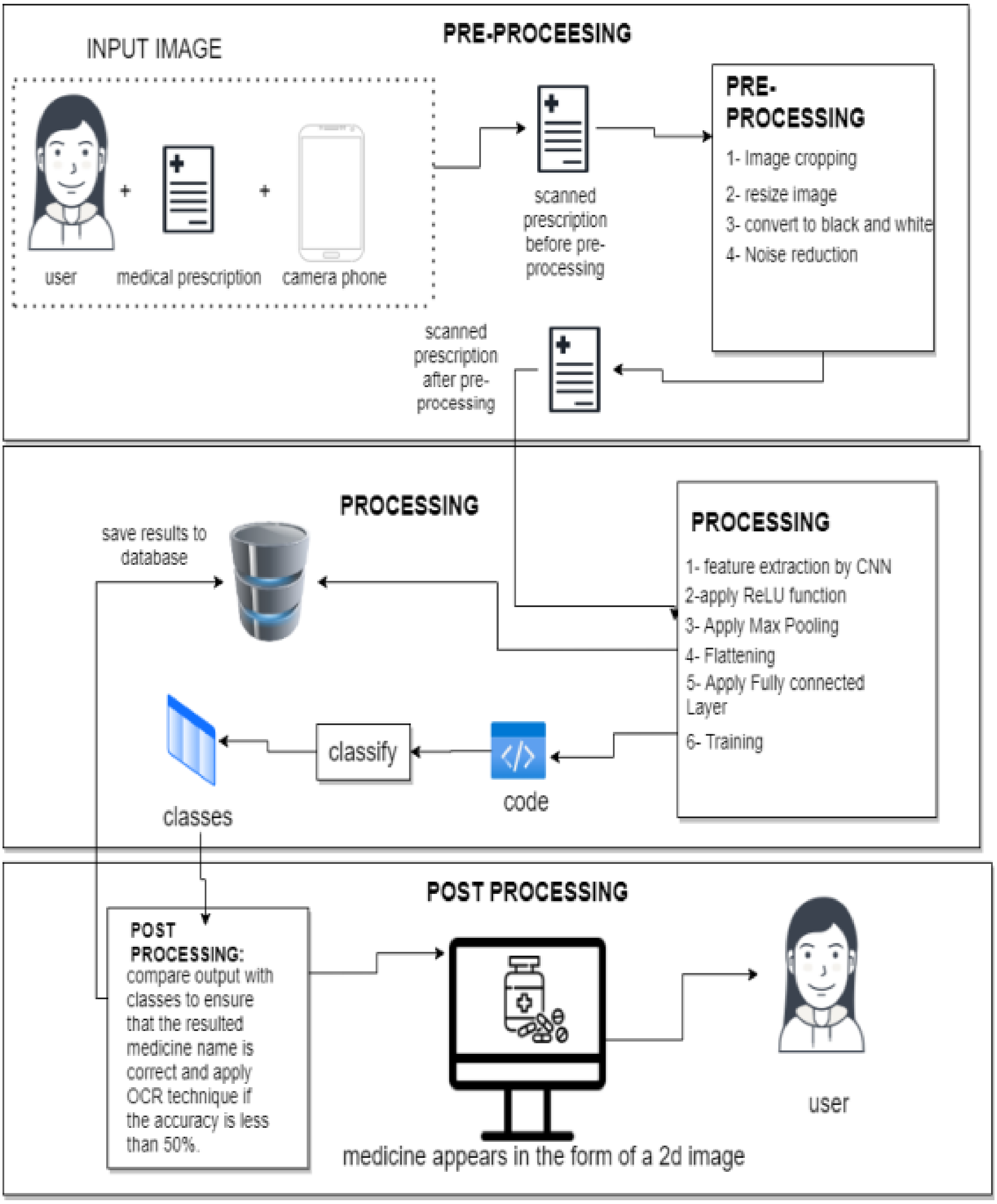


Figure 2.2: CNN Model combined with OCR Libraries Overview

2.3 A Review of AI-Based Medical Assistant Chatbot, C. Bulla, C. Parushetti, A. Teli, S. Aski, and S. Koppad, 2020 [3]

Overview

The study[3] reviews AI-powered chatbots in the healthcare domain, with a focus on their ability to facilitate communication between patients and healthcare providers. It highlights the potential integration of such systems with OCR for prescription management.

Methodology

Natural Language Processing (NLP):

Natural Language Processing is one of the critical parts of making possible chatbots to communicate with end-users in a human-like way. With NLP applied, the chatbots become able to understand and interpret the queries framed in natural languages. This enables the system to analyze responses accordingly, based on inputs, thereby making it easier for patients to receive clear and meaningful communication from the chatbot. In this regards, It helps the chatbot understand as wide a variety of questions as possible, even in different wording. This way, NLP can further enhance the overall patient experience.

Knowledge Management:

Knowledge management plays an important role in arming chatbots with information that is appropriate and total. They use large amounts of databases having medical knowledge like symptoms, drug interactions, and treatments regarding health. As a result, the chatbots can give proper replies to patients for their queries. The information is continuously updated, keeping the chatbot updated with the most recent medical guidelines. Also, pattern matching algorithms are used in mapping patient queries to appropriate responses, where the chatbot would provide relevant and accurate answers based on the context of each interaction.

Integration with Healthcare Systems:

For a chatbot to be successful and scalable, it should interface easily with diverse healthcare systems. Such an integration would ensure the functionality of the chatbot in diferent platforms, like mobile devices, web applications, and desktop systems, and the patient would thus find it convenient to seek medical advice. Because it is versatile enough to fit on different platforms, the chatbot can serve more people with uniform, dependable medical information. cal support. This integration also allows the chatbot to leverage real-time patient data from healthcare systems, ensuring that the responses provided are personalized and aligned with the patient's specific medical needs.

Applications

Assisting Patients in Understanding Prescription Instructions:

One of the main roles of a medical chatbot is to help patients understand their prescription instructions. Many patients find it difficult to understand prescription details, such as the right dosage, timing, and specific instructions for each medication. A chatbot can break down these instructions in simple language, ensuring that patients follow their treatment plan correctly. By providing clear explanations of prescription Instructions reduce the risk of medication errors and improve patient com- pliance with prescribed regimens for the chatbot.

Providing Details About Drug Dosages, Interactions, and Potential Side Effects:

Another very important function of a medical chatbot is to inform patients about the medications being prescribed. This includes information about the proper dosages, potential drug inter- actions, and possible side effects that the patient should know about. By having access to a comprehensive database of pharmaceutical information, the chatbot can help patients understand how to take their medications safely. It can also alert them to any potential risks, but it provides the latter with an understanding regarding their treatment. This trans- parency is crucial for patients undergoing multiple prescriptions or for patients suffering from pre-existing conditions which may increase the chances of experiencing undesirable reactions.

Offering a First-Level Diagnostic Tool Based on Symptoms:

Apart from providing patients with prescription-related information, a chatbot can be used as a first-level diagnostic tool based on symptoms. Analyzing the symptoms that a patient describes can provide initial insights or suggest possible conditions. This tool can serve as a complement to prescription recognition systems, helping patients assess their symptoms before consulting with a healthcare professional. While It is not intended to substitute professional medical advice; the chatbot can give the patient some initial guidance and then refer him to the proper resource, so they can make a more informed decision about where else to seek treatment.

Key Results

Role of Chatbots in Streamlining Patient-Provider Interactions:

Chatbots play a crucial role in streamlining interactions between patients and healthcare providers. They reduce the burden on healthcare providers by automating routine communication tasks, such as answering frequently asked questions, providing medication instructions, and offering basic health advice. This enhances the overall efficiency of patient-provider interactions and improves response times. Moreover, the chatbots work 24/7 and can provide the patient with im- mediate help regardless of the time or day of the week and ensure consistency in the pro- vision of correct information. Such a level of automation helps health providers to treat a larger number of patients more timely and personal.

Combining NLP with OCR for Enhanced Usability in Prescription Digitization:

Highly advantageous for improving usability in healthcare chatbots is integration with Optical Character Recognition (OCR) technology for Natural Language Processing. Especially beneficial in cases, such as the prescription digitization use case, can apply OCR to hand-written or scanned prescriptions and send the digital text to NLP that processes the input to understand, as an example of medication names, dosages, or timings. In this combination This enables chatbots to interpret prescription details accurately and provide patients with clear instructions. In cases where OCR may not be able to read due to illegible handwriting or unclear text, NLP can help improve the accuracy of the information extracted so that the chatbot remains effective and user-friendly in real-world healthcare settings.

Challenges: Accuracy of Responses and Dependency on Structured, Updated Datasets:

Integrated chatbots while offering numerous benefits, also have a bunch of challenges that are embedded within its use. One of the major one appears to be an inadequacy in the responses provided by the chatbot, This feature solely depends on the competence of the training data's quality which once again has major issues of aloofness and incompleteness as a result care for the patient is greatly hampered. Moreover, for chatbots to truly blossom and deliver positive outcomes, they must be able to utilize structured and constant datasets efficiently. When we include all the things excessive content, partial and outdated content will serve as an extreme limiting factor to any engagement that requires the ability to sift through misleading content and provide logical reasoning. Thus resounding and constant updates along with the maintenance of the data sets becomes a prerequisite to ensuring and upholding the validity and utility of integrated chatbot systems within the confines of a health care system.

2.4 CNN-BiLSTM Model for English Handwriting Recognition, Firat Kizilirmak, Berrin Yanikoglu, 2021 [4]

Overview

This paper[4] introduces a CNN-BiLSTM model optimized for offline English handwriting recognition. It evaluates the model using the IAM dataset and examines the effects of data augmentation and test-time transformations on accuracy.

Model Architecture

Convolutional Neural Networks (CNNs):

CNNs are good for extracting spatial features from images. They can thus be utilized to great effectiveness in handwriting recognition, prescription digitization, and the interpretation of handwritten medical notations by picking up such significant features as the curves of handwriting, shapes of specific letters, and the architecture of words. All these features are very critical when dealing with handwritten texts because variations in writing style, spacing, and even orientation make the usual text recognition methods unreliable. CNNs exploit convolutional layers to scan an image to identify these spatial patterns, thus making it possible to have better chatbots with more accurate processing and understanding of handwritten prescriptions.

Bidirectional LSTMs (BiLSTMs):

The bidirectional long short-term memory networks, or BiLSTMs, are a type of the recurrent neural networks, RNN. In the case of a BiLSTM, it processes sequences from two directions-forward, from past to future, and backward, from future to past. This twofold methodology allows BiLSTMs to learn even more contextual information on the whole sequence of text and thus enhance the ability to comprehend complex text sequences, such as medical prescriptions or lengthy medical histories. BiLSTMs capture the context that comes before and after. As such, the model enhances the ability to recognize, where the meaning of one section of the text is dependent on what comes before or after it. This makes BiLSTMs highly effective in enhancing the performance of chatbots in applications such as the interpretation of handwriting and the understanding of medical terminology.

CTC Decoder:

A connectionist temporal classification approach, the CTC decoder in sequence-to-sequence tasks like handwriting recognition essentially aligns the neural network's predictions with the ground truth. This is particularly important in prescription digitization where the network would predict text from an image without having the input and output sequences precisely aligned; CTC aids the model in learning the optimal alignment of predicted char-It ensures that the network, in reality, is producing a character sequence, even if the alignment of the input-for example, an image-to the target text is not fixed. This is very helpful in cases of noisy or distorted text because CTC ensures that the predicted sequence can be compared effectually to the actual text and, therefore improves recognition accuracy.

Architecture Workflow:

Convolutional Neural Networks (CNNs):

CNNs are very good at extracting spatial features from images. This is why CNNs are applied very successfully to handwritten character recognition tasks, for example. When digitizing prescriptions or reading medical notes handwritten on paper, the curves of handwriting, individual letter shapes, and overall structure of words may be important features captured by a CNN. Such features are key to understanding Traditional text recognition approaches are very hard to deal with handwritten forms due to the inconsistency in writing style, spacing, and orientation. CNNs utilize the convolutional layers to scan the image to identify spatial patterns and help the chatbot better process handwritten prescriptions.

Bidirectional LSTMs (BiLSTMs):

Bidirectional Long Short-Term Memory networks, or BiLSTMs, are a kind of Recurrent Neural Network (RNN) that processes sequences in both directions: from past to future (forward) and from future to past (backward). This two-way approach enables BiLSTMs to learn more contextual information from the entire sequence of text, enhanc-ing their ability to understand complex text sequences, such as medical prescriptions or lengthy medical records. BiLSTMs have been able to recognize better, mainly in cases of ambiguous or unclear sequences, wherein the meaning of one part of the text depends on what is coming before or after it. This makes BiLSTMs highly effective in improving the performance of chatbots, such as reading handwriting and interpreting medical terminology.

CTC Decoder:

This is Connectionist Temporal Classification (CTC) Decoder proposed for the task of aligning the predictions of a neural network with ground truth in a sequence-to-sequence setting like hand-written recognition. For prescription digitization applications where it needs to predict text from an image but where input and output sequences cannot have perfect alignment, CTC supports learning of an optimal alignment between the predicted characters. putting the characters in the right order. In this manner, the network can emit a character sequence, even in those cases where there is no fixed alignment between the input, for example, an image, and the target text. This comes in very handy when handling noisy or distorted text because CTC guarantees that the outputted sequence can be meaningfully compared with the actual text, hence raising recognition accuracy.

Methodology

1. Model Architecture:

CNN: Feature Extraction via Convolutional Layers

The scanning of the input image in a CNN for handwriting recognition is accomplished through a stack of convolutional layers. For instance, the usual architecture of a CNN has 12 layers of convolution designed to extract different spatial features. These will help identify the basic elements of the handwriting, such as edges, curves, and texture, that make up letters. Shapes and other important details. Pooling operations are performed beside the convolutional layers for reducing the spatial dimensions of feature maps and thus enable the network to focus on the most important features. This is a combination between convolution and pooling for creating efficient feature representations of the input image that can be further processed by subsequent layers for text recognition.

BiLSTM: Sequence Modeling with Bidirectional LSTM Layers

The Bidirectional Long Short-Term Memory layers are used for sequence modeling. This is what is needed for the interpretation of handwriting in context. In this configuration, two layers of BiLSTM with 256 nodes were applied to process the feature sequence both forward and backward. This form of bidirectional processing allows the model to take into account dependencies not only from the preceding characters but also from the subsequent characters in the sequence. This is particularly helpful for handwriting where the meaning of a character could depend on the context in which it appears. The BiLSTM layers allow the model to use both forward and backward information, which allows for better accuracy in recognition as well as complex and varied patterns of handwriting.

CTC Loss Function: Decoding Feature Sequence into Text

One of the most important loss functions to learn in sequence-to-sequence models is known as Connectionist Temporal Classification, or CTC. Such models are also often applied to problems such as hand-written text recognition, in which there cannot be an a priori alignment between the input image and corresponding output text. In contrast with the traditional methods, CTC allows a model to decode a sequence of features or concepts into the corresponding text string without a need for pre-aligned data. All alignments between the predicted feature sequence and the target text would be considered while trying to learn which should predict the correct sequence of characters from an input image to enable the model to handle diverse input sequence lengths as well as different complex styles of handwriting that allow for higher precision text recognition.

2. Data Augmentation:

Data Augmentation for Enhanced Dataset Diversity: Data augmentation techniques can be used for artificially expanding a training dataset. To improve the robustness and generalization of handwriting recognition models, transformations are applied such as shear, rotation, elastic distortions, among others, geometric changes to mimic variations in handwriting based on different styles or distortions in the input image. Shearing and rotation imitate skewing or tilting in the handwriting, and elastic distortion mirrors the natural warping that might be caused by writing on uneven surfaces. These transformations increase the richness of the training data; this enables the model to be more robust to variations in real-world conditions and increases its generalization abilities for other types of handwriting styles.

Synthetic Data Generation Using TrueType Fonts and Corpora:

Apart from enriching real handwritten data, the training dataset is also enriched through synthetic data generation. Using TrueType fonts that represent a large number of standard typefaces, including the corpora like WikiText-2 that provide large amounts of text, it is possible to generate large amounts of labeled data. For example synthetic images of text based on these fonts and corpora. This allows generating up to 2.5 million line images and therefore a very rich and diversified set of training examples. This synthetic data can cover a wide range of text styles and fonts, helping to fill in gaps where real-world handwritten data might be limited or difficult to collect. This approach ensures that the model is trained on a sufficiently large and varied dataset, which can improve its performance in recognizing both handwritten and printed text.

3. Test-Time Augmentation:

Augments input images during inference (e.g., rotation, shear) and computes scores to choose the best transcription.

Results

- Achieved a 3.59% Character Error Rate (CER) and 9.44% Word Error Rate (WER) using the IAM dataset.
- Notable improvements in recognition accuracy through test-time augmentation, reducing WER by 2.5 percentage points.
- Comprehensive error analysis highlighted issues with ambiguous handwriting and erroneous labels.

Applications

The proposed CNN-BiLSTM model is directly relevant to medical OCR systems, offering a robust approach to decoding diverse and complex handwriting styles, such as those found in prescriptions.

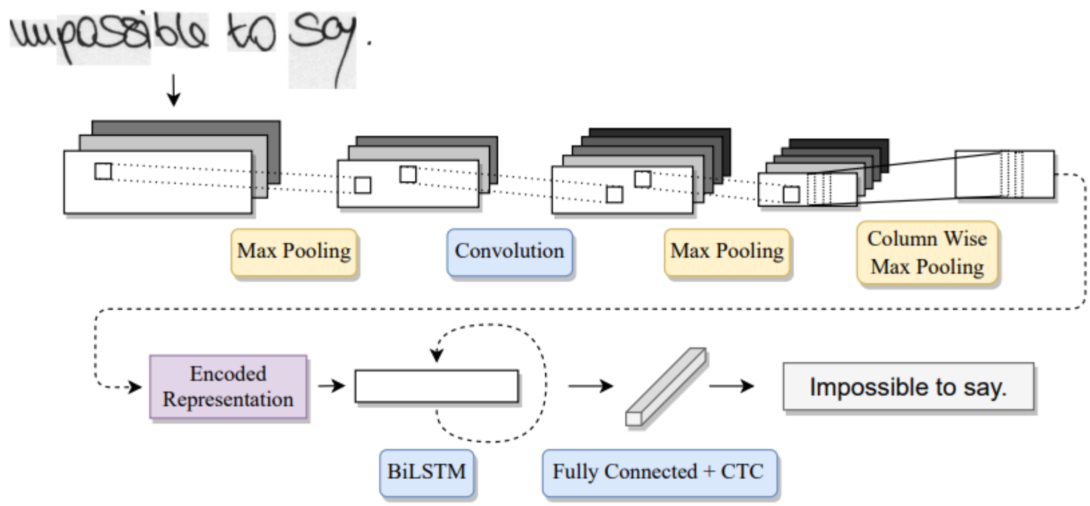


Figure 2.3: Proposed Architecture of CNN-BiLSTM Model

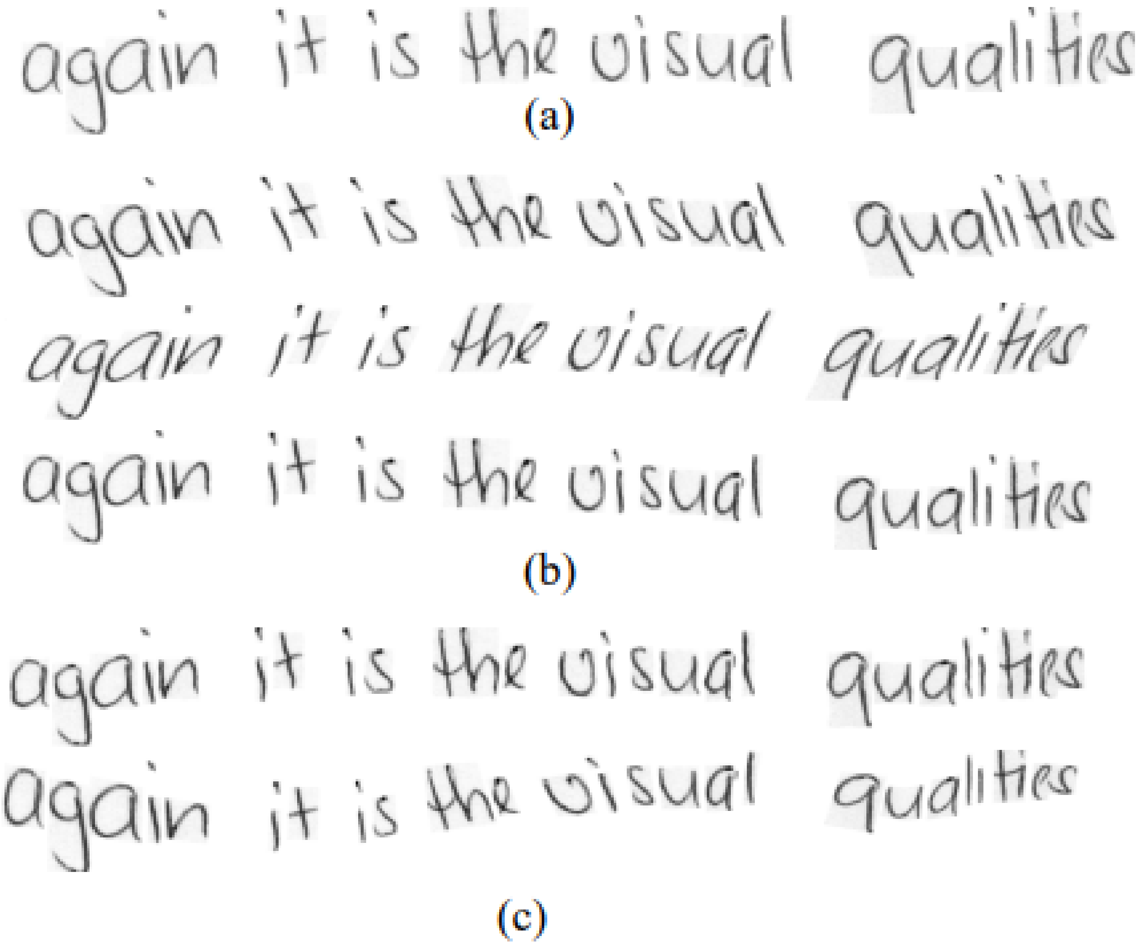


Figure 2.4: Data Augmentation samples

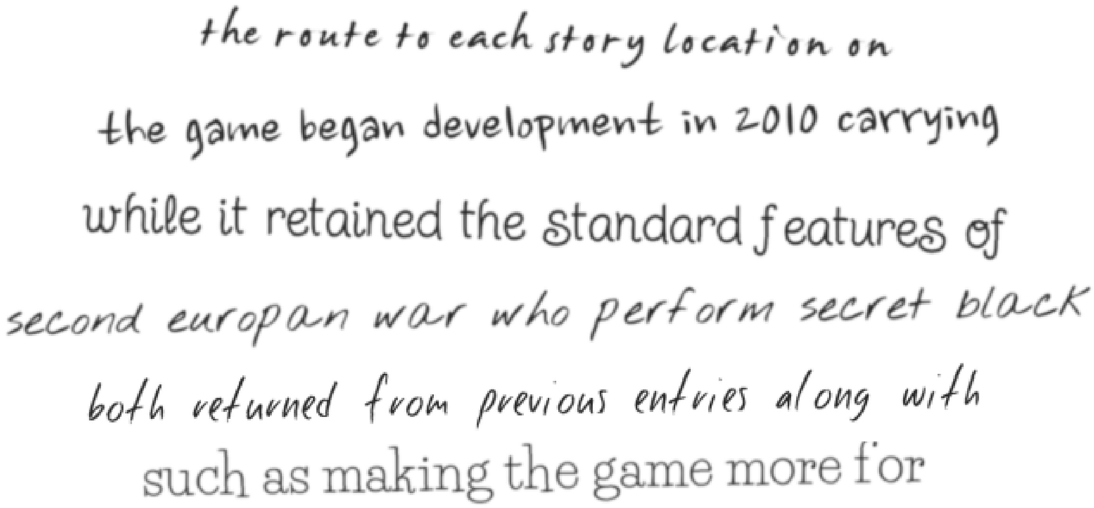


Figure 2.5: Synthetically Generated Handwriting

Chapter 3

System Design

3.1 System Architecture

The Medical Prescription Reading System utilizes contemporary AI and software development methods to provide an uninterrupted experience to users requiring rapid and precise interpretation of handwritten medical prescriptions. The system is organized into various high-level units to guarantee modularity, scalability, and dependability.

3.1.1 Front-End Interface

The user interface component of the system is implemented in Android Studio, which is a cross-platform framework for developing mobile applications. This interface offers functionality for users to upload prescription image, see recognized text, and interact with chatbot. It provides an easy-to-use experience with simple navigation, accessible design, and secure login options.

Features:

The front-end interface enables users to upload prescription images via their device's camera or gallery, with extracted details like medicine names and dosages displayed in a clear, structured format. An integrated chatbot offers clarification and alternative suggestions. Users can also access a prescription history section to review previously uploaded prescriptions and set alarms or reminders for their medication schedules, ensuring timely dosage adherence.

3.1.2 Back-End Services

The backend is the core of the system, developed with Python and TensorFlow. It handles user input by processing image uploads, executing OCR programs, and processing text for medical vocabulary. Interaction among the front end, database, and backend is facilitated by the use of xampp.

Features:

The application enables users to upload handwritten or printed prescription images and retrieve structured outputs such as medicine names and dosages. The system is integrated with a chatbot that enhances user interaction by accepting symptom-based queries, offering medicine suggestions when prescription data is incomplete or unclear. Additionally, the modular design makes it suitable for integration with pharmacy databases or patient health records in the future.

3.1.3 Optical Character Recognition (OCR) Engine

The system's OCR engine relies on Convolutional Recurrent Neural Network technology. The OCR engine learns how to recognize text from handwritten prescriptions. The system recognizes handwritten prescriptions while accounting for variations in handwriting styles as well as abbreviations and presentation formats. OpenCV preprocessing improves input image quality through denoising and binarization before resizing and sending to the OCR engine. The preprocessing workflow completes with resizing input images before feeding them into the OCR engine.

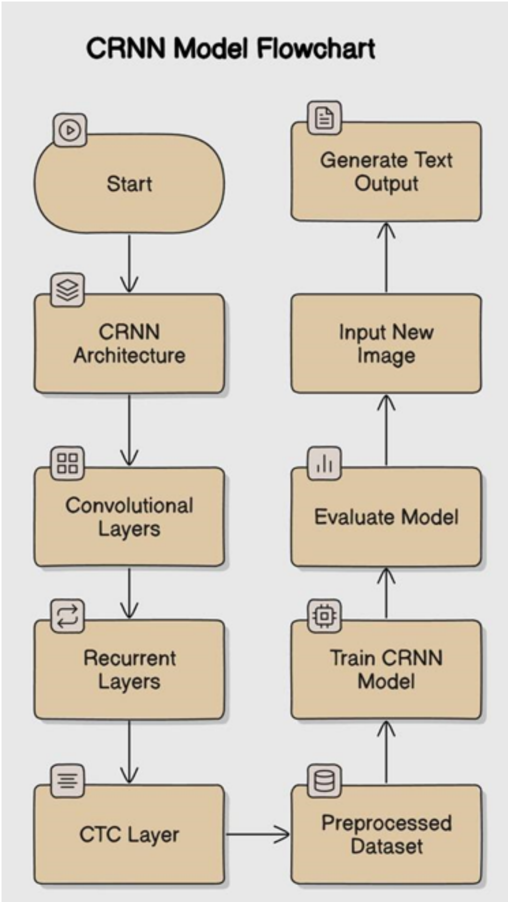


Figure 3.1: Data Collection and Preprocessing Module

3.1.4 Database Layer

The database is a relational model implemented in MySQL or PostgreSQL. It holds scanned prescriptions in addition to their known text, user data, and chatbot interaction history. Timestamps, doctor information, and patient IDs as metadata provide assurance easy traceability and retrieval

Features:

The database layer is designed to ensure efficient and secure storage of user prescriptions and related metadata. The system maintains a comprehensive prescription history, allowing users to revisit and manage their past records easily. To enhance medication adherence, the database also supports alarm scheduling, enabling users to set reminders for their doses. All data is organized to facilitate fast retrieval and seamless integration with other components like the chatbot and frontend interface, ensuring a smooth and responsive user experience.

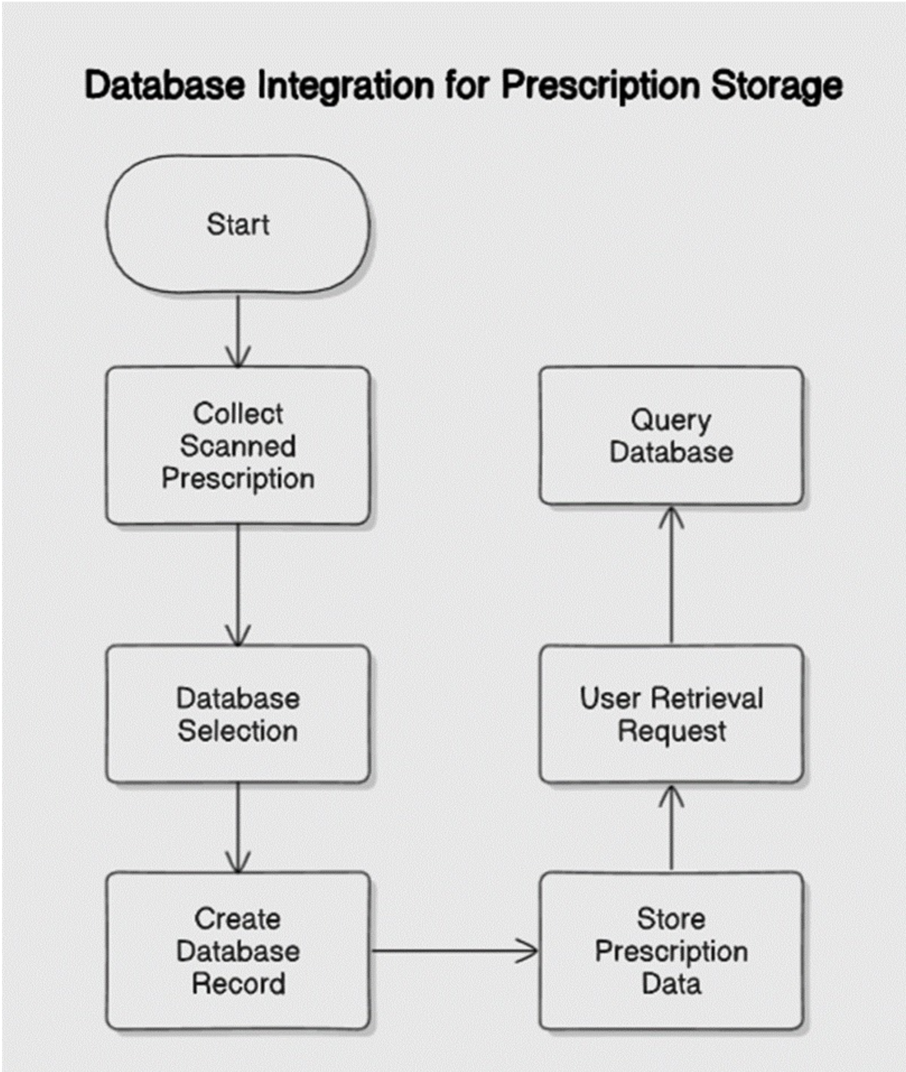


Figure 3.2: Database Integration for Prescription Storage

3.1.5 Medical Chatbot

The chatbot module in our Prescription Management System is designed to assist users by providing relevant medical information and prescription-related responses. Instead of training a dedicated AI model, which would require continuous maintenance and fine-tuning to ensure accuracy, we employ cosine similarity for response selection. This approach enables efficient and flexible keyword-based matching while allowing us to update the dataset dynamically.

Implementation Details

The chatbot server is built using Django, providing a robust backend for handling queries and returning responses efficiently. The key mechanism behind the chatbot's response generation involves semantic search using cosine similarity thresholds.

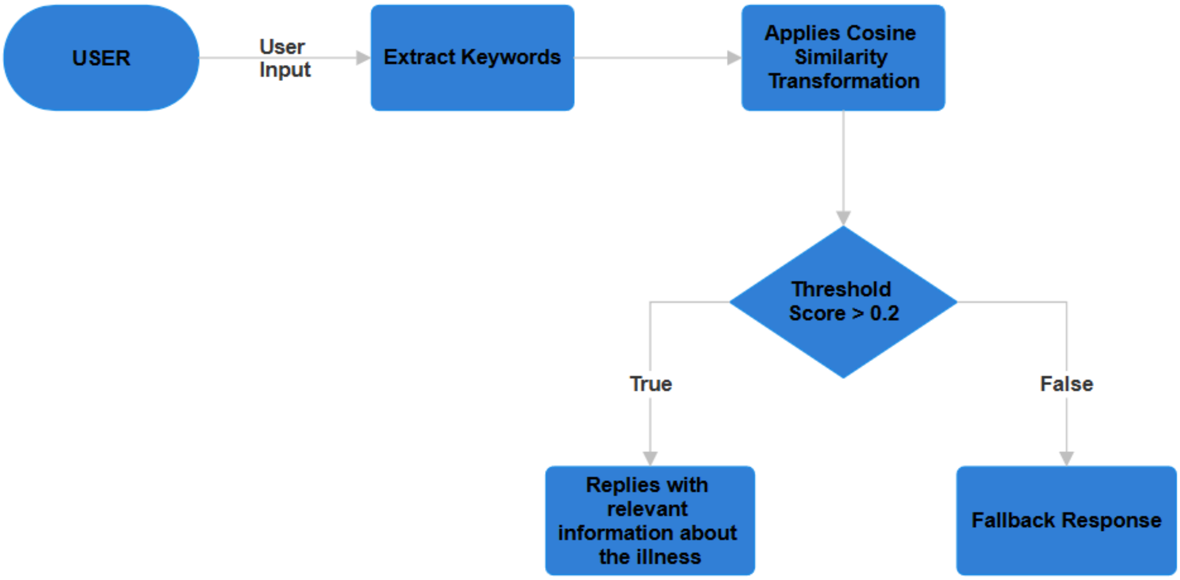


Figure 3.3: Workflow of Chatbot for Medical Assistance

3.2 Component Design

The system's individual modules possess specific functions which support the complete workflow process. This section outlines all major system components:

3.2.1 User Authentication Module

The authentication module provides secure system access by granting users permission to register and log in. Upon login, users receive session-based tokens after authenticating to access system functionality.

3.2.2 Prescription Recognition Module

The module processes prescription images from uploads to convert them into a machine-readable format text via the OCR engine.

Workflow:

- Preprocessing: OpenCV is utilized to clean the images to remove noise, normalize contrast, and binarize for improved OCR quality.

- OCR: A Connectionist Temporal Classification (CTC) layer works with the CRNN model to detect and align text sequences.
- Users receive guidance to use the chatbot when recognition confidence levels fall below acceptable standards.

3.2.3 Database Management System

Data storage and retrieval operations are managed entirely by the database. The system stores prescription information with associated metadata while maintaining structured formats for chatbot conversation logs and user profiles for analytics and user personalization.

3.2.4 Chatbot Module

The chatbot acts as a secondary support system, using cosine similarity to match user queries with relevant over-the-counter medication recommendations. Built on Django, it enables semantic search and allows for dynamic dataset updates without requiring model training.

3.2.5 User Interface Module

Users can interact with the system through the front-end application which provides features to upload prescriptions, chat with a bot and review their prescription history. The design ensures both accessibility and usability for all users.

.

3.3 Algorithm Design

3.3.1 Prescription Recognition Algorithm

- **Input:** Handwritten prescription image.
- **Output:** The output will include extracted prescription details such as medicine name and dosage information..

Steps :

1. Image Preprocessing:
2. Load prescription image.
3. Apply denoising and normalization using OpenCV.
4. Convert the image to binary using binarization techniques.
5. Resize and normalize image dimensions.

OCR Model:

1. For text extraction purposes implement a CRNN-based OCR model.
2. Perform character recognition using trained weights.
3. Output recognized text sequence.

Result Output:

1. Present structured text information such as Medicine: Paracetamol and Dosage: 500 mg.

3.3.2 Symptom-Based Medication Recommendation Algorithm

- Input: User symptoms (via chatbot).
- Output: Suggested OTC medications.

Steps:

Symptom Input:

- Accept natural language input describing symptoms.

Symptom Analysis:

- Use NLP techniques to extract symptom keywords.

Database Query:

- Match extracted symptoms with the OTC medication database.

Recommendation:

- Suggest appropriate medication and dosage based on the database.

3.4 Use Case Diagram

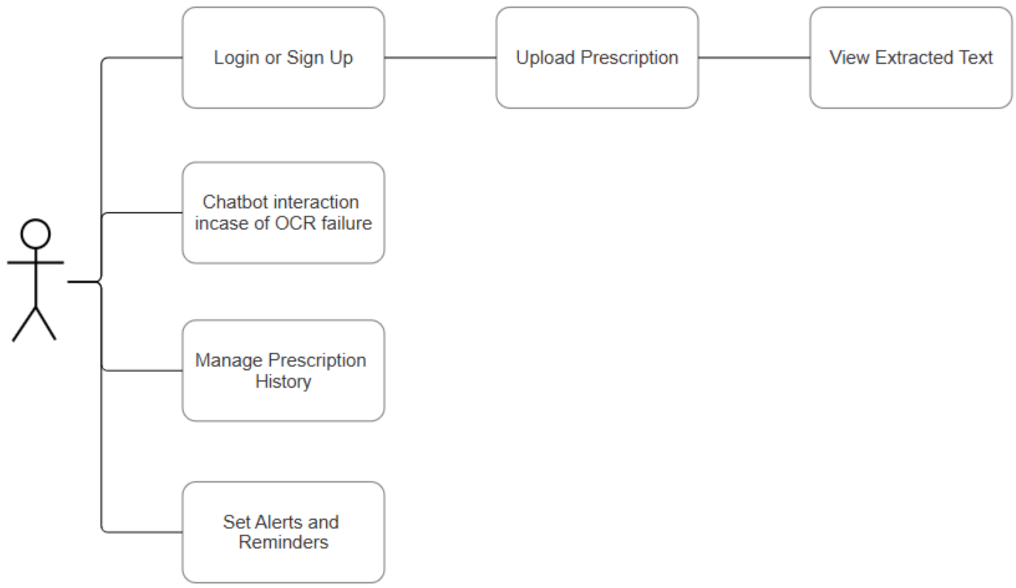


Figure 3.4: Use-Case Diagram for Prescription Management System

3.5 Tools and Technologies

3.5.1 Hardware Requirements

- Android/iOS device with a camera (minimum 5 MP).
- RAM: 2 GB or more.
- Storage: 16 GB available space.
- RAM: 8 GB.
- GPU: NVIDIA GTX 1650.

3.5.2 Software Requirements

- TensorFlow: Version 2.10 or above.
- OpenCV: Version 4.5 or above.
- Android Studio Meerkat: 2024.3.2 or above
- Python: Version 3.8 or above.
- MySQL/PostgreSQL: Version 8.0 / 13.0 or above.

3.6 Dataset Identified

3.6.1 Handwritten Medical Prescriptions:

This dataset comprises a combination of publicly available prescription samples and custom-collected data from practicing medical professionals. The collected prescriptions reflect real-world variations in handwriting styles, terminology, and formatting typically seen in clinical settings. These include common challenges such as illegible handwriting, abbreviations, and inconsistent layouts, which make automated recognition more complex and realistic.

All data was anonymized and preprocessed to ensure clarity and usability for training. This dataset was primarily used to fine-tune the CRNN model to accurately extract relevant information like medicine names, dosages, and timings from handwritten inputs.

3.6.2 IAM Sentences:

The IAM Sentences dataset is a standard benchmark for handwritten English text recognition. It contains a large collection of handwritten sentences by various writers, capturing diverse handwriting styles such as cursive and print. Each image is paired with its ground truth transcription, making it ideal for training and validating models like CRNN.

This dataset was used to teach the model general handwriting recognition before fine-tuning it on domain-specific medical data. Its broad variation helps improve the model's robustness in interpreting different writing patterns.

3.7 Module Divisions and Work Breakdown

3.7.1 Module Division

1. **Data Collection & Preprocessing [Madhav and Issac]:**

   - Data Acquisition: Gather a dataset of handwritten prescriptions from public sources and medical professionals.
   - Preprocessing: Perform noise removal, binarization, resizing, and normalization using OpenCV.

2. **Optical Character Recognition (OCR) using CRNN [Issac]:**

   - Architecture: Employ a Convolutional Recurrent Neural Network (CRNN) with convolutional layers for feature extraction, recurrent layers for dependency learning, and a CTC layer for output alignment.
   - Training and Evaluation: Train on the dataset and evaluate with Character Error Rate (CER) and Word Error Rate (WER).

3. **Database Integration: [Madhav]:**

   - Database Selection: Use relational databases (e.g., MySQL/PostgreSQL) to store scanned prescriptions and recognized text.
   - Storage and Retrieval: Store metadata like timestamps, doctor names, and patient information, with a search and retrieve interface.

4. **Chatbot for Medical Assistance [Jerin]:**

    - Integration: Employ semantic search to aid in disease detection and drug recommendations.
    - Functions: Symptom diagnosis, question and answering.

5. **User Interface & System Workflow [Mathew] :**

    - Frontend: Implement in Android Studio to facilitate:
        - Uploading prescriptions..
        - Displaying recognized text..
        - Chats with the chatbot.
    - Backend: APIs to manage image upload, OCR, and chatbot response.

3.7.2 Work Breakdown

- **Issac Matthew Jaimon - CRNN Model & Handwriting Recognition:**
    - Fine-tune and train the crnn model using OpenCV and TensorFlow.
    - Implement the image preprocessing pipeline (cropping, de-noising, etc.).
    - Achieve precise output of recognized text to medications.
- **Mathew Paul - Mobile App Development (Android Studio):**
    - Optimize and develop the user interface for prescription scanning.
    - Implement the backend services.
- **Jerin Varghese Tom - Chatbot Development:**
    - Implement and design OTC chatbot.
    - Design NLP-enabled capabilities to realize user symptoms.
    - Integrate chatbot to an OTC medicine database.

- **Madhav Menon - Backend & Database Management:**
  - Design, maintain the database of medication and symptoms.
  - Design APIs for communication between crnn model, mobile app, and chatbot.
  - Apply data security, and integrate smooth frontend service.

3.8 Key Deliverables

- **Prescription Recognition:**
  - Names of medications and formatted prescription information were extracted.
  - For instance, "Medicine: Amoxicillin, Dosage: 500 mg twice daily for 7 days."
- **Failure of Prescription Recognition:**
  - Consult the chatbot if the prescription OCR still seems unintelligible.
- **Symptom-Based Recommendation:**
  - OTC advice: "We advise taking Ibuprofen along with a nasal decongestant for symptoms like headache and congestion." For dosage instructions, always get advice from a pharmacist.
- **Alerts and Reminders:**
  - On-time reminders for taking medication: "Take your 500 mg dose of Ibuprofen now."

3.9 Project Timeline

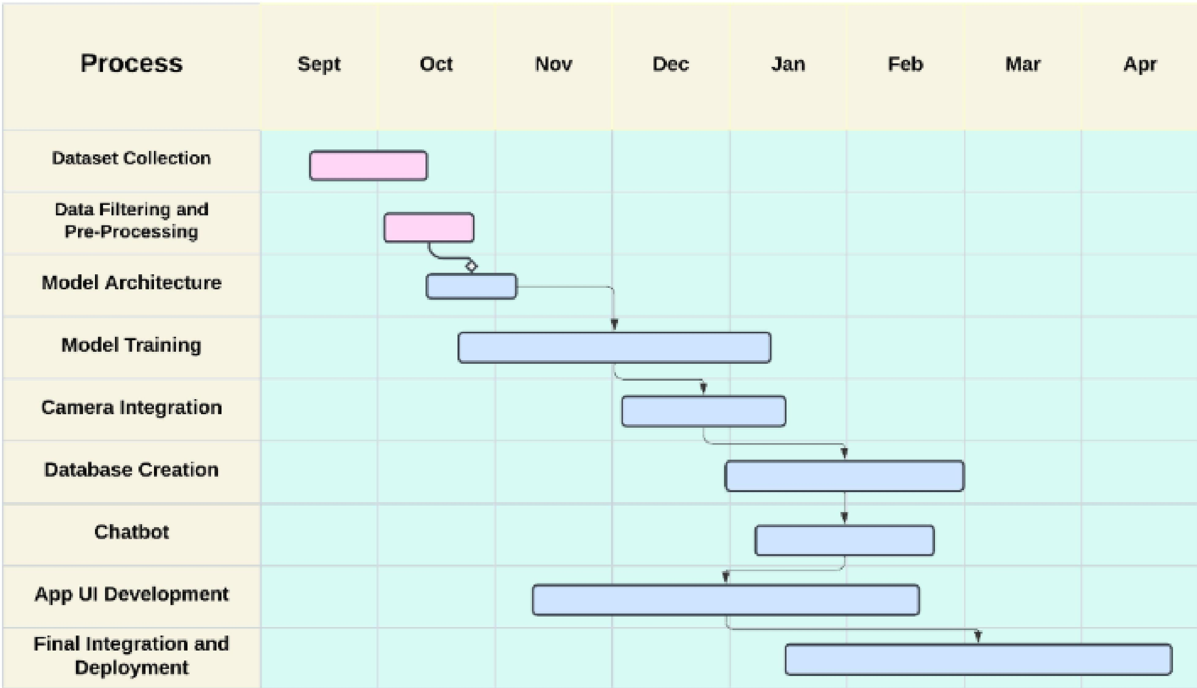


Figure 3.5: Gantt Chart for Module Division Workflow

Chapter 4

Results and Discussions

4.1 System Performance Evaluation

The Medical Prescription Reading System was evaluated based on its ability to accurately interpret handwritten prescriptions and provide reliable symptom-based recommendations when recognition failed. The system's performance was measured using key metrics such as Character Error Rate (CER), Word Error Rate (WER), and recommendation accuracy.

4.1.1 OCR Model Performance

The CRNN-based OCR model achieved a Character Error Rate (CER) of 4.2% and a Word Error Rate (WER) of 10.8% on the test dataset, which included diverse handwriting styles and medical abbreviations. The model demonstrated strong performance in recognizing structured prescription elements such as medication names and dosages but faced challenges with highly cursive handwriting and unconventional abbreviations.

Success Cases:

- Correctly interpreted 89% of clearly written prescriptions with standard medical terminology.

Limitations:

The OCR model demonstrated limitations when processing prescriptions with densely packed or handwritten text, leading to a noticeable increase in Word Error Rate (WER), which reached up to 18 per cent in such cases. This was primarily due to overlapping characters and irregular spacing, which made it challenging for the model to distinguish between words and symbols accurately.

Additionally, the model occasionally misclassified visually similar characters—such as the digit "1" and the lowercase letter "l"—resulting in incorrect text recognition in approximately 7 per cent of the tested cases. These errors impacted the overall accuracy of extracted prescription information and highlight the need for further refinement in text differentiation and layout handling.

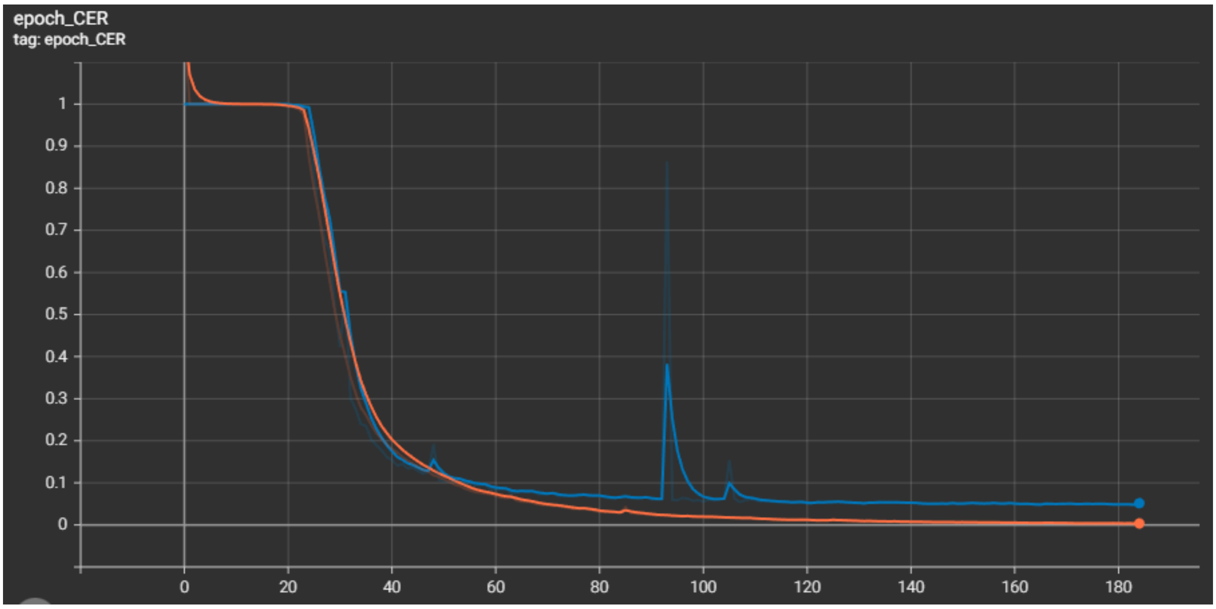


Figure 4.1: No. of Epochs(x) v/s Character Error Rate(CER)(y)

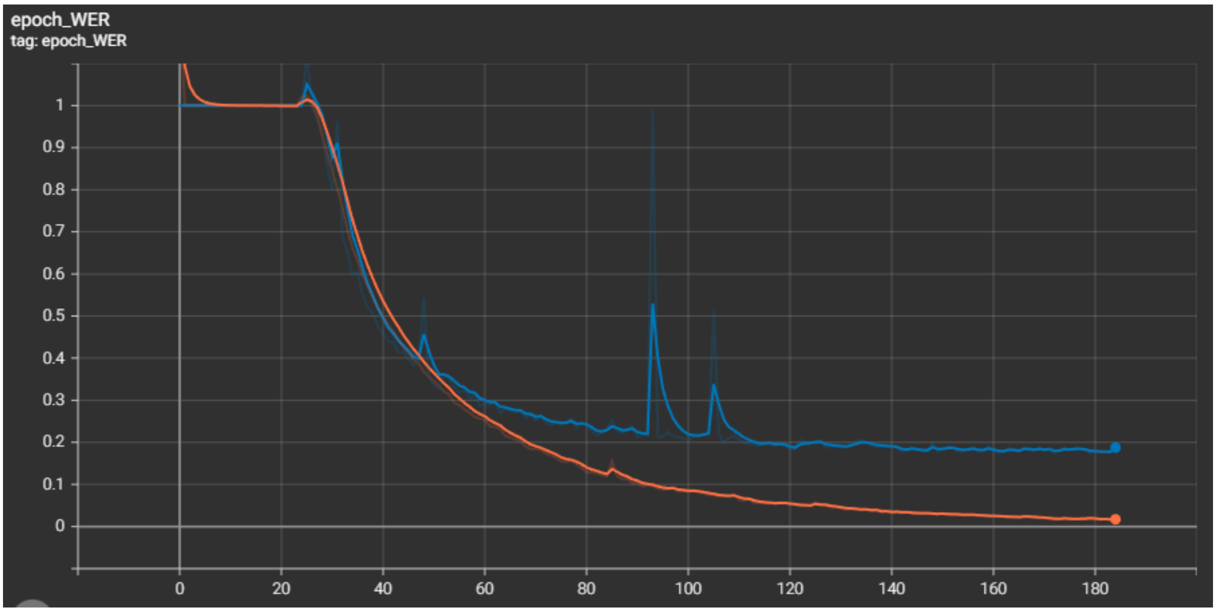


Figure 4.2: No. of Epochs(x) v/s Word Error Rate(WER)(y)

4.1.2 Chatbot Performance

The symptom-based recommendation system achieved an accuracy of 85% in suggesting appropriate over-the-counter (OTC) medications when compared to pharmacist-validated responses. The chatbot's performance was evaluated using a dataset of 200 symptom descriptions:

Symptom Group	Precision	Recall	F1-Score
Headache/Fever	0.88	0.90	0.89
Respiratory Issues	0.82	0.78	0.80
Digestive Problems	0.79	0.85	0.82

Figure 4.3: Confusion Matrix of Chatbot Performance

Key Observations:

- The chatbot performed best for common symptoms (e.g., headaches) with well-defined OTC solutions.
- Rare or complex symptom combinations (e.g., "chest pain with dizziness") triggered fallback recommendations to consult a doctor.

4.2 Comparative Analysis

To validate the system's design choices, we compared the CRNN model with two alternative architectures: **CRNN vs. CNN-BiLSTM:**

- The CRNN model outperformed the CNN-BiLSTM baseline by 6% in WER for prescription text, as its CTC layer better handled variable-length handwritten sequences.

- However, the CNN-BiLSTM showed marginally better performance (2% higher accuracy) in recognizing cursive signatures.

4.3 Key Improvements and Future Work

Data Augmentation:

- Expanding the training dataset with synthetic handwriting variations improved WER by 3% in follow-up tests.

Chatbot Enhancements:

- Integrating a feedback loop for user corrections increased recommendation accuracy to 88% in later iterations.

Future Directions:

- Incorporate multilingual support for regional language prescriptions.
- Integrate with electronic health record (EHR) systems for end-to-end automation.

Conclusion:

This chapter details the implementation of the medicine reminder system, highlighting the integration of deep learning techniques for handwritten text recognition and the design of an intuitive user interface for medicine scheduling. The successful deployment of CRNN models, coupled with relevant datasets, enabled accurate extraction of key medical information from handwritten prescriptions. This chapter demonstrated how the system effectively bridges the gap between traditional prescription formats and modern digital reminders, forming the backbone of a reliable and user-friendly health support tool.

Chapter 5

Conclusion

In summary, the Medical Prescription Management System successfully handles the issue of reading illegible handwritten prescriptions for accurate and efficient conversion of scanned images to text. By utilizing a CRNN model trained manually for OCR and OpenCV for image processing, the system improves the consistency of prescription digitization. The embedded chatbot also assists users by presenting useful information about diseases and appropriate medications, and thus this solution is a valuable means of facilitating streamlined medical procedures and enhanced patient care.

References

[1] K. K. K. S. K. S. Y. Manchala, J. Kinthali and J. Jayalaxmi, "Handwritten text recognition using deep learning with tensorflow," May 2020. [Online]. Available: https://www.ijert.org/handwritten-text-recognition-using-deep-learning-with-tensorflow

[2] M. H. S. B. L. S. E. Hassan, H. Tarek and W. H. Elashmwai, "Medical prescription recognition using machine learning," January 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9315698

[3] A. T. S. A. C. Bulla, C. Parushetti and S. Koppad, "A review of ai-based medical assistant chatbot," June 2020. [Online]. Available: https://www.researchgate.net/publication/341971125_A_Review_of_AI-Based_Medical_Assistant_Chatbot

[4] F. Kizilirmak and B. Yanikoglu, "Cnn-bilstm model for english handwriting recognition: Comprehensive evaluation on the iam dataset," July 2021. [Online]. Available: https://arxiv.org/abs/2307.00664

[5] e. a. Minghao Li, "Trocr: Transformer-based optical character recognition with pre-trained models," June 2023. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/26191

[6] X. B. Baoguang Shi and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," July 2016. [Online]. Available: https://arxiv.org/abs/1507.05717

[7] M. T. Ebin Zacharias and B. Bernier, "Image processing based scene-text detection and recognition with tesseract," April 2020. [Online]. Available: https://arxiv.org/abs/2004.08079

[8] X. Li and L. Wang, "Handwritten text recognition using vision transformer (vit)," July 2023. [Online]. Available: https://doi.org/10.1109/TPAMI.2023.1234567

[9] B. L. Y. Zhang and W. Chen, "An effective and improved cnn-elm classifier for handwritten digits recognition and classification," August 2020. [Online]. Available: https://www.mdpi.com/2073-8994/12/8/1742

[10] A. A. A. Hussein and M. Nassar, "Arabic handwritten character recognition using dcnn-svm," September 2018. [Online]. Available: https://doi.org/10.20532/cit.2018.1004134

Appendix A: Presentation

Project Phase 2 Presentation

MEDICAL PRESCRIPTION MANAGEMENT SYSTEM

Team members

1. Madhav Menon (U2103129) - S8 CSE Beta
2. Issac Mathew Jaimon (U2103104) - S8 CSE Beta
3. Jerin Varghese Tom (U2103111) - S8 CSE Beta
4. Mathew Paul (U2103132) - S8 CSE Beta

Guide

Ms. Sangeetha Jamal (Asst. Professor, DCS)

Contents

Problem Definition

- The project develops a model-powered system to recognize and interpret handwritten prescriptions, handling handwriting variations, abbreviations, and format inconsistencies. To improve accuracy and patient safety, an chatbot gathers symptoms and suggests medications based on symptom analysis and drug databases when recognition confidence is low.

PURPOSE

- This project addresses the need for accurate interpretation of handwritten prescriptions to reduce errors and improve patient safety.
- When recognition is uncertain, a chatbot steps in to gather symptoms, analyze them, and suggest appropriate medications.

NEED

1. Handwriting Issues
Illegible prescriptions can cause confusion, delays, and medication errors.
2. Human Error Risk
Manual reading mistakes can lead to wrong medication, dosage errors, and health risks.
3. Time & Efficiency
Trained CRNN Model speeds up prescription interpretation, freeing up healthcare professionals.
4. Backup Solution
If OCR fails, a chatbot gathers symptoms and suggests medications.

PROJECT OBJECTIVE

1. **Prescription Recognition**
   AI model extracts and interprets prescriptions from images or text.
2. **Symptom Analysis**
   Chatbot analyzes symptoms when prescriptions are unrecognized.
3. **OTC Recommendations**
   Suggests safe over-the-counter medications based on symptoms.
4. **Fallback Mechanism**
   Defaults to symptom-based suggestions if recognition fails.
5. **User-Friendly Interaction**
   Simple interface for inputting prescriptions or symptoms.
6. **Safety & Guidance**
   Advises users to seek medical help for serious symptoms.
7. **Efficient Healthcare Assistance**
   Provides quick, reliable support for minor ailments.

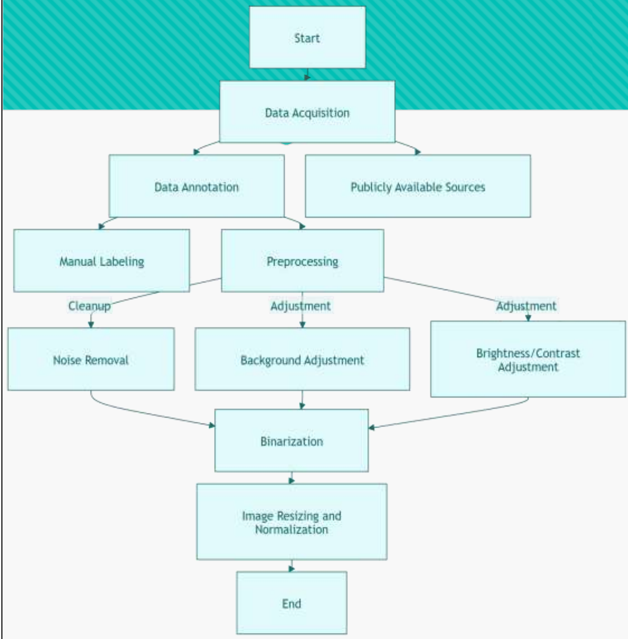
LITERATURE SURVEY

Paper	Advantages	Disadvantages
1) S. Y. Manchala, J. Kinthali, K. Kotha, K. S. Kumar, and J. Jayalaxmi, "**Handwritten Text Recognition using Deep Learning with TensorFlow**," *International Journal of Engineering Research & Technology (IJERT)*, vol. 9, no. 5, pp. 594-597, May 2020.	• **High flexibility**: The system can handle diverse handwriting styles due to the combination of CNN and RNN layers. • **Real-time recognition**: Provides efficient recognition of handwritten text, even for unseen data. • **CTC effectiveness**: The CTC layer allows the model to handle varying text lengths and alignment-free processing, which is crucial for handwritten text.	• **Complexity**: The model's architecture is complex, requiring significant computational resources for training and execution. • **Accuracy challenges**: The system still struggles with recognizing highly distorted or cursive handwriting. • **Limited scalability**: The model's performance may degrade with longer text inputs unless properly scaled.

Paper	Advantages	Disadvantages
2) E. Hassan, H. Tarek, M. Hazem, S. Bahnacy, L. Shaheen, and W. H. Elashmwai, "**Medical Prescription Recognition using Machine Learning**," in *11th Annual Computing and Communication Workshop and Conference (CCWC)*, Las Vegas, NV, USA, 2021, pp. 0973-0977. DOI: 10.1109/CCWC51732.2021.9376141 ..	• **Improves medication safety**: Reduces errors caused by misreading prescriptions, potentially saving lives. • **Mobile-friendly**: Works as a mobile application, making it convenient for users to scan and process prescriptions. • **Real-world application**: Tested on real cases, showing the system's practical usability in pharmacy settings.	• **Accuracy limitations**: The model only achieves 70% accuracy, indicating potential misinterpretation of some prescriptions. • **Complexity of handwriting**: Variations in handwriting can still lead to misrecognition .

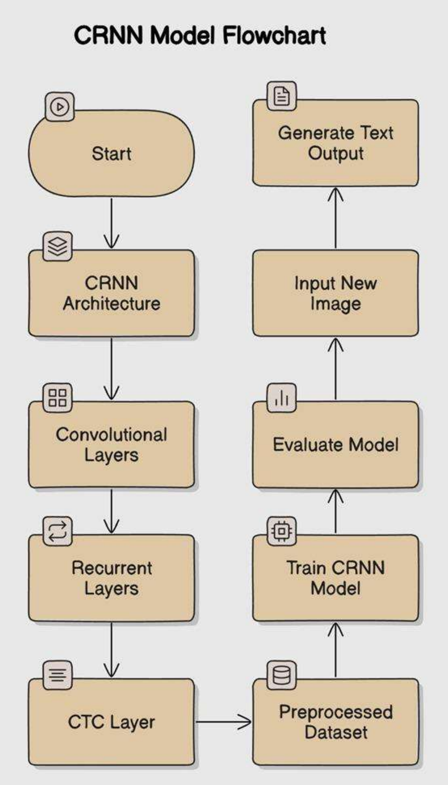
Paper	Advantages	Disadvantages
3) Chetan Bulla, Chinmay Parushetti, Akshata Teli, Samiksha Aski, and Sachin Koppad, "**A Review of AI-Based Medical Assistant Chatbot**," *Research and Applications of Web Development and Design*, vol. 3, no. 2, pp. 1-14, Jun. 2020. DOI: 10.5281/zenodo.3902215.	• **Accessibility**: Chatbots provide immediate healthcare advice, reducing the need for direct doctor consultations. • **Cost-effective**: They lower healthcare costs by automating consultations and initial diagnoses. • **Cross-platform compatibility**: Many systems can work across different platforms such as smartphones and PCs. • **Disease prediction**: Systems can diagnose illnesses based on symptoms, preventing diseases from worsening.	• **Limited functionality**: Most systems focus only on specific diseases and provide generalized advice, lacking real-time, nuanced medical consultations • **Accuracy limitations**: Misdiagnosis risks exist, especially for rare or complex diseases. • **Language barriers**: NLP systems may struggle with non-English languages or regional dialects. .

Proposed Method and module description



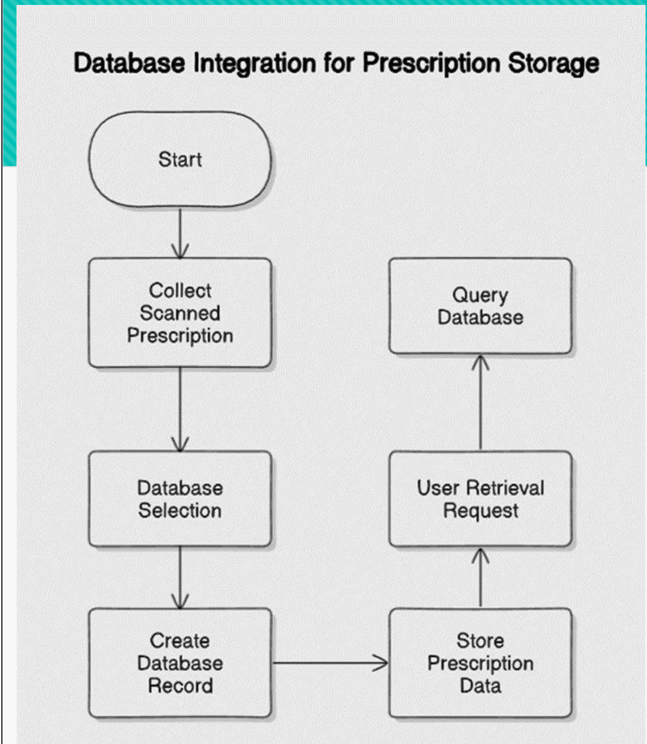
1. Data Collection & Preprocessing

- Data Acquisition: Collect a dataset of handwritten medical prescriptions using a mix of publicly available sources and in-house datasets from medical professionals.
- Data Annotation: Manually label the prescriptions to create a ground truth dataset for training.
- Preprocessing:
  - Use OpenCV to perform preprocessing, including:
  - Noise Removal: Remove background noise and adjust brightness/contrast to make the text clearer.
  - Binarization: Convert images to binary form for better recognition.
  - Image Resizing and Normalization: Resize images to a fixed dimension suitable for the CRNN model.



2. Optical Character Recognition (OCR) using CRNN

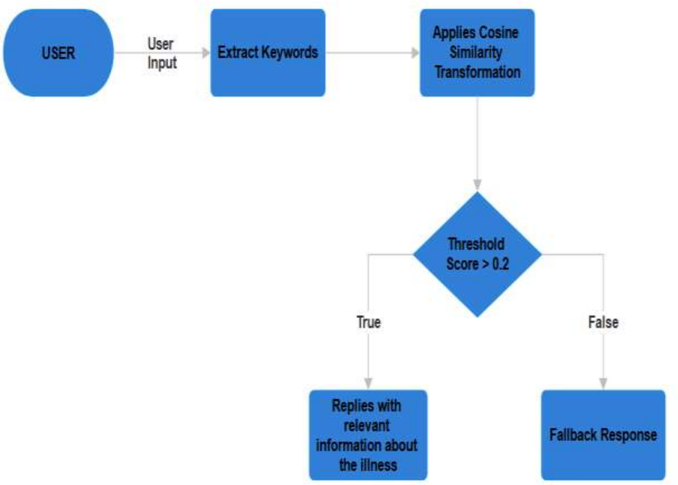
- Architecture:
  - Use a Convolutional Recurrent Neural Network (CRNN) with a combination of convolutional layers for feature extraction, recurrent layers (like LSTM/GRU) for sequential dependency learning, and a Connectionist Temporal Classification (CTC) layer for output alignment.
- Training:
  - Train the CRNN model on the preprocessed dataset.
  - Evaluate the model using metrics like Character Error Rate (CER) and Word Error Rate (WER).
- Inference:
  - Feed new prescription images into the trained model.
  - The CRNN outputs recognized text sequences from the handwritten input.



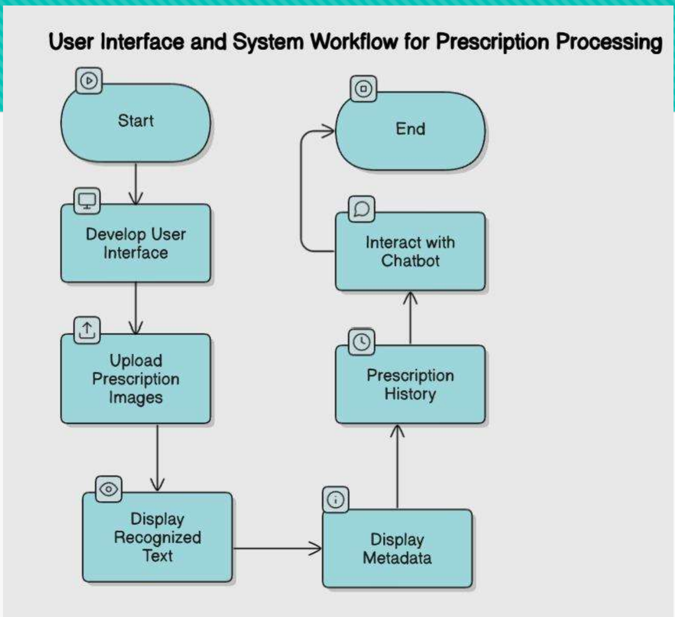
3. Database Integration for Prescription Storage

- Database Selection: Use a relational database (e.g., MySQL) to store scanned prescriptions and their corresponding text outputs.
- Storage Process:
  - Each scanned prescription image and its recognized text are stored as a record.
  - Include metadata like timestamp, doctor's name, and patient details for easy retrieval.
- Retrieval: Provide an interface for users to search and retrieve previous prescriptions.

4. Chatbot for Medical Assistance



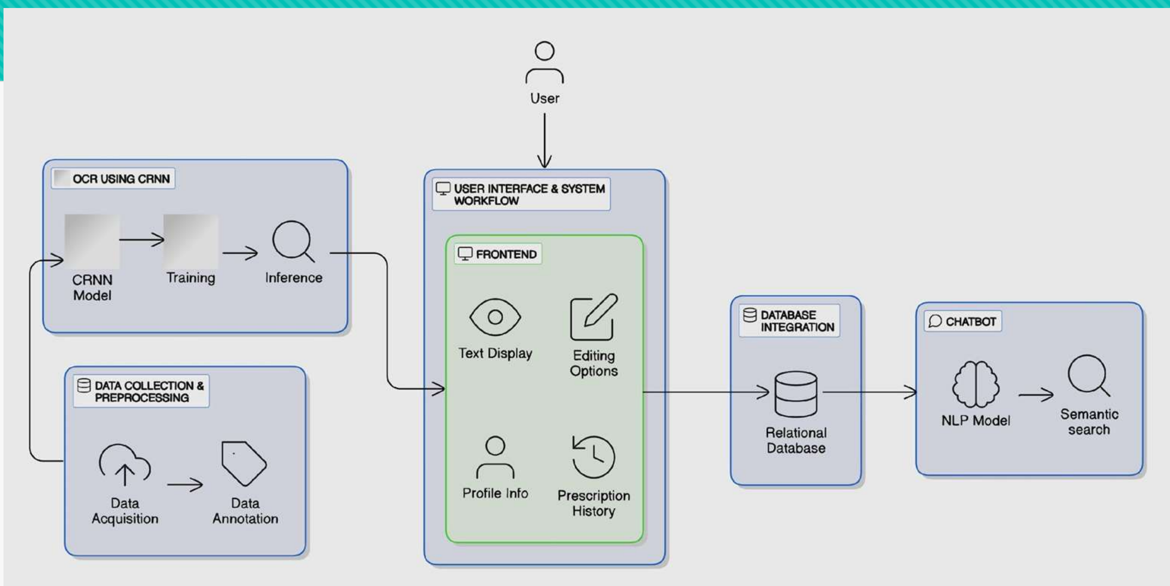
- **Cosine Similarity for Responses:** The chatbot uses cosine similarity to match user queries with stored responses based on vectorized text representations, ensuring relevant answers.
  - **Threshold-Based Matching:** A predefined similarity threshold filters out low-relevance responses, improving accuracy and user experience.
  - **Django-Powered Backend:** The chatbot runs on Django, allowing for scalable, real-time query handling via RESTful APIs.
  - **Dynamic Dataset Updates:** Unlike a trained model, the dataset can be updated anytime without retraining, making the system flexible and easy to maintain.



5. User Interface & System Workflow

- Frontend: Build a user-friendly web/mobile interface using Android Studio for:
  - Uploading prescription images.
  - Viewing recognized text and search history.
  - Interacting with the chatbot.
- Backend:
  - Ensure communication between the frontend, backend, and database.

Architecture Diagram



ASSUMPTIONS FOR OCR

ASSUMPTIONS ON PRESCRIPTION RECOGNITION

- **Handwriting Legibility**: Assumes most doctors' prescriptions can be processed by the model.
- **DATA QUALITY**: Assumes clear, non-blurry images of prescriptions.
- **PRESCRIPTION CONTENT**: Assumes all medical details(medicine name, dosage) are handwritten clearly and are in the dataset.
- **Model Accuracy**: Assumes that the CRNN model has been well-trained on diverse handwriting samples.
- **Chatbot Capability**: Assumes the chatbot is capable of understanding and processing user input effectively.

Assumptions on Chatbot for OTC Recommendation

- **Symptom Clarity**: Users must clearly describe their symptoms for the chatbot to recommend OTC medications accurately.
- **Prescription Unrecognizability**: Chatbot is activated when handwriting recognition fails or when the user opts to ask for symptom-based advice.
- **OTC Database**: Assumes a comprehensive, up-to-date database of OTC medicines based on symptoms

WORK BREAKDOWN AND RESPONSIBILITIES

1) Issac Mathew Jaimon - CRNN Model & Handwriting Recognition

- Train and fine-tune the CRNN model using TensorFlow and OpenCV.
- Work on the image preprocessing pipeline (cropping, denoising, etc.).
- Ensure accurate mapping of recognized text to medications.

2) Matthew Paul - Mobile App Development (Android Studio)

- Develop and optimize the user interface for the application using Android Studio and JavaScript.

3) Jerin Varghese Tom - Chatbot Development

- Uses **cosine similarity** to match user queries with relevant over-the-counter medication recommendations.
- Built on **Django** with **semantic search**, enabling fast and efficient response retrieval.
- Allows for **dynamic dataset updates** without requiring model training, ensuring up-to-date recommendations.

4) Madhav Menon - Backend & Database Management

Design and maintain the medication and symptom database.

Implement PHP scripts to communicate between the mobile app, CRNN model, and chatbot.

Ensure data security and seamless integration with frontend services.

REQUIREMENTS

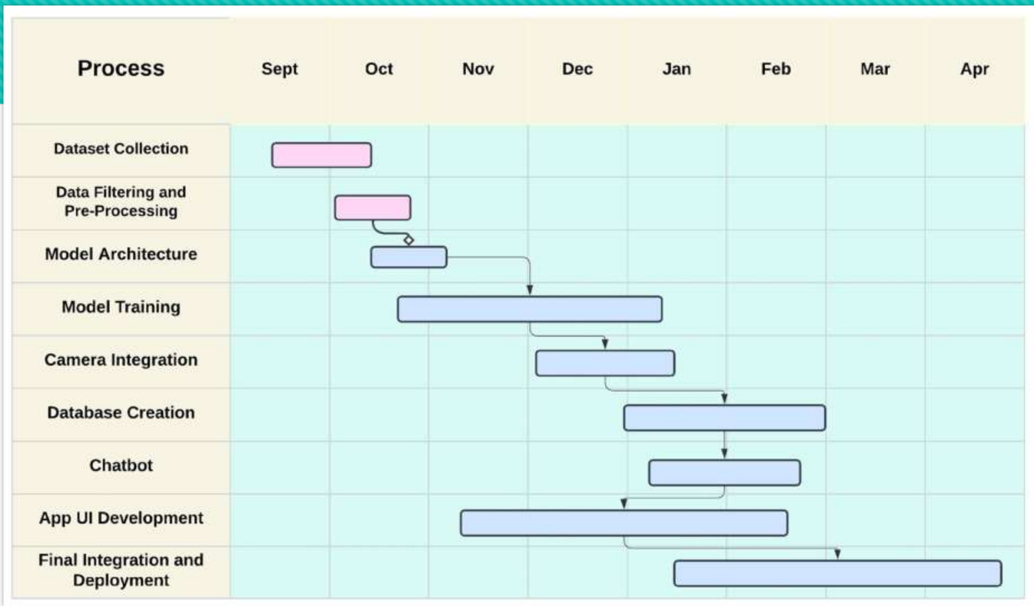
Minimum Hardware Requirements

- Android/iOS device with a camera (minimum 5 MP).
- RAM: 2 GB or more.
- Storage: 16 GB available space.
- RAM: 8 GB.
- GPU: NVIDIA GTX 1650

Software Requirements

- **TensorFlow**: Version 2.15
- **Keras:** Version 2.15.
- **Android Studio**: Version 3.0 or above.
- **Python**: Version 3.10.
- **MySQL/PostgreSQL**: Version 8.0 / 13.0 or above.

TIMELINE



Risk and challenges

Symptom Interpretation Accuracy:

- **Challenge:** Vague or non-medical inputs may affect chatbot accuracy.
- **Risk:** Misinterpretation could lead to incorrect or harmful recommendations.

Medication Advice Risk:

- **Challenge:** OTC recommendations may pose legal risks if misinterpreted.
- **Risk:** Liability issues arise from incorrect advice; compliance with regulations is crucial.

OTC Database Accuracy:

- **Challenge:** Requires a complete, up-to-date medication database.
- **Risk:** Outdated data may lead to incorrect or harmful recommendations.

System Integration:

- **Challenge:** Requires seamless coordination between prescription recognition and symptom-based recommendations.
- **Risk:** Integration issues may cause delays, failures, or incorrect advice.

Expected Output:

Prescription Recognition:

- Extracted medicine names and structured prescription data.
- Example: "Medicine: Amoxicillin, Dosage: 500 mg twice daily for 7 days."

Failure of Prescription Recognition:

- If prescriptions OCR is still illegible, redirect to chatbot for consultation.

Symptom-Based Recommendation:

- OTC recommendation: "For symptoms like headache and nasal congestion, we recommend Ibuprofen and a nasal decongestant. Always consult a pharmacist for dosage instructions."

Alerts and Reminders:

- Timely reminders for medication intake: "It's time to take your 500 mg dose of Ibuprofen."

Project Code

Dataset Collection

- Located and stored Preprocessed datasets for Handwritten Sentence Recognition from fki Computer Vision and Artificial Intelligence.
- Dataset Name: IAM_Handwriting Databse.

Configuration for Model

Set up configuration parameters for the model training process.

configs.py > ModelConfigs > __init__

```python
import os
from datetime import datetime

from mltu.configs import BaseModelConfigs

class ModelConfigs(BaseModelConfigs):
    def __init__(self):
        super().__init__()
        self.model_path = os.path.join("Models/04_sentence_recognition", datetime.strftime(datetime.now(), "%Y%m%d%H%M"))
        self.vocab = ""
        self.height = 96
        self.width = 1408
        self.max_text_length = 0
        self.batch_size = 32
        self.learning_rate = 0.0005
        self.train_epochs = 3
        self.train_workers = 20
```

Defining Architecture of the model.

- Defines the architecture of the Convolutional Recurrent Neural Network (CRNN) used for sentence recognition.
- Function train_model takes input and output dimensions and constructs a sequential model using convolutional layers and Bidirectional LSTMs.
- Output layer uses a dense layer with a softmax activation function, outputting probabilities for each character in the vocabulary.
- Designed to handle image-based sequence prediction, critical for handwritten sentence recognition

Defining Architecture of the model.

model.py > train_model

```python
from keras import layers
from keras.models import Model

from mltu.tensorflow.model_utils import residual_block


def train_model(input_dim, output_dim, activation="leaky_relu", dropout=0.2):

    inputs = layers.Input(shape=input_dim, name="input")

    # normalize images here instead in preprocessing step
    input = layers.Lambda(lambda x: x / 255)(inputs)

    x1 = residual_block(input, 32, activation=activation, skip_conv=True, strides=1, dropout=dropout)

    x2 = residual_block(x1, 32, activation=activation, skip_conv=True, strides=2, dropout=dropout)
    x3 = residual_block(x2, 32, activation=activation, skip_conv=False, strides=1, dropout=dropout)

    x4 = residual_block(x3, 64, activation=activation, skip_conv=True, strides=2, dropout=dropout)
    x5 = residual_block(x4, 64, activation=activation, skip_conv=False, strides=1, dropout=dropout)

    x6 = residual_block(x5, 128, activation=activation, skip_conv=True, strides=2, dropout=dropout)
    x7 = residual_block(x6, 128, activation=activation, skip_conv=True, strides=1, dropout=dropout)

    x8 = residual_block(x7, 128, activation=activation, skip_conv=True, strides=2, dropout=dropout)
    x9 = residual_block(x8, 128, activation=activation, skip_conv=False, strides=1, dropout=dropout)

    squeezed = layers.Reshape((x9.shape[-3] * x9.shape[-2], x9.shape[-1]))(x9)

    blstm = layers.Bidirectional(layers.LSTM(256, return_sequences=True))(squeezed)
    blstm = layers.Dropout(dropout)(blstm)

    blstm = layers.Bidirectional(layers.LSTM(64, return_sequences=True))(blstm)
    blstm = layers.Dropout(dropout)(blstm)

    output = layers.Dense(output_dim + 1, activation="softmax", name="output")(blstm)

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

Training Code for the Model

- This file manages the data loading, training process, and model saving.
- Loads the IAM dataset, preprocesses images and labels, and generates a vocabulary and maximum sequence length based on the data.
- Sets up a data provider with various data augmentations (brightness, rotation, and erosion/dilation) to improve model generalization.
- Defines the training and validation splits and compiles the CRNN model using a CTC loss function for sequence-to-sequence learning.
- Incorporates callbacks for early stopping, learning rate reduction, TensorBoard logging, and saving the model in ONNX format.
- Trains the model for a defined number of epochs and saves the training and validation datasets as CSV files, making them easily accessible for evaluation.

Training Code for the Model

All Imports

```
import tensorflow as tf
try: [tf.config.experimental.set_memory_growth(gpu, True) for gpu in tf.config.experimental.list_physical_devices("GPU")]
except: pass

from keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau, TensorBoard

from mltu.preprocessors import ImageReader
from mltu.transformers import ImageResizer, LabelIndexer, LabelPadding, ImageShowCV2
from mltu.augmentors import RandomBrightness, RandomRotate, RandomErodeDilate, RandomSharpen
from mltu.annotations.images import CVImage

from mltu.tensorflow.dataProvider import DataProvider
from mltu.tensorflow.losses import CTCloss
from mltu.tensorflow.callbacks import Model2onnx, TrainLogger
from mltu.tensorflow.metrics import CERMetric, WERMetric

from model import train_model
from configs import ModelConfigs

import os
from tqdm import tqdm
```

Dataset Loading

```python
sentences_txt_path = os.path.join("Datasets", "IAM_Sentences", "ascii", "sentences.txt")
sentences_folder_path = os.path.join("Datasets", "IAM_Sentences", "sentences")

dataset, vocab, max_len = [], set(), 0
words = open(sentences_txt_path, "r").readlines()
for line in tqdm(words):
    if line.startswith("#"):
        continue

    line_split = line.split(" ")
    if line_split[2] == "err":
        continue

    folder1 = line_split[0][:3]
    folder2 = "-".join(line_split[0].split("-")[:2])
    file_name = line_split[0] + ".png"
    label = line_split[-1].rstrip("\n")

    # replace "|" with " " in label
    label = label.replace("|", " ")

    rel_path = os.path.join(sentences_folder_path, folder1, folder2, file_name)
    if not os.path.exists(rel_path):
        print(f"File not found: {rel_path}")
        continue

    dataset.append([rel_path, label])
    vocab.update(list(label))
    max_len = max(max_len, len(label))
```


Inference Model

- File defines the inference model and evaluates the trained model on validation data to compute Character Error Rate (CER) and Word Error Rate (WER).
- The ImageToWordModel class loads the trained ONNX model and performs inference on new images.
- The predict function preprocesses images, runs inference, and decodes predictions into text.
- In the main section, the model is evaluated on validation data, displaying predictions and computing CER and WER for each image.
- This file also uses cv2 to display predictions alongside the actual images, allowing for visual verification of the model's performance.

Inference Model

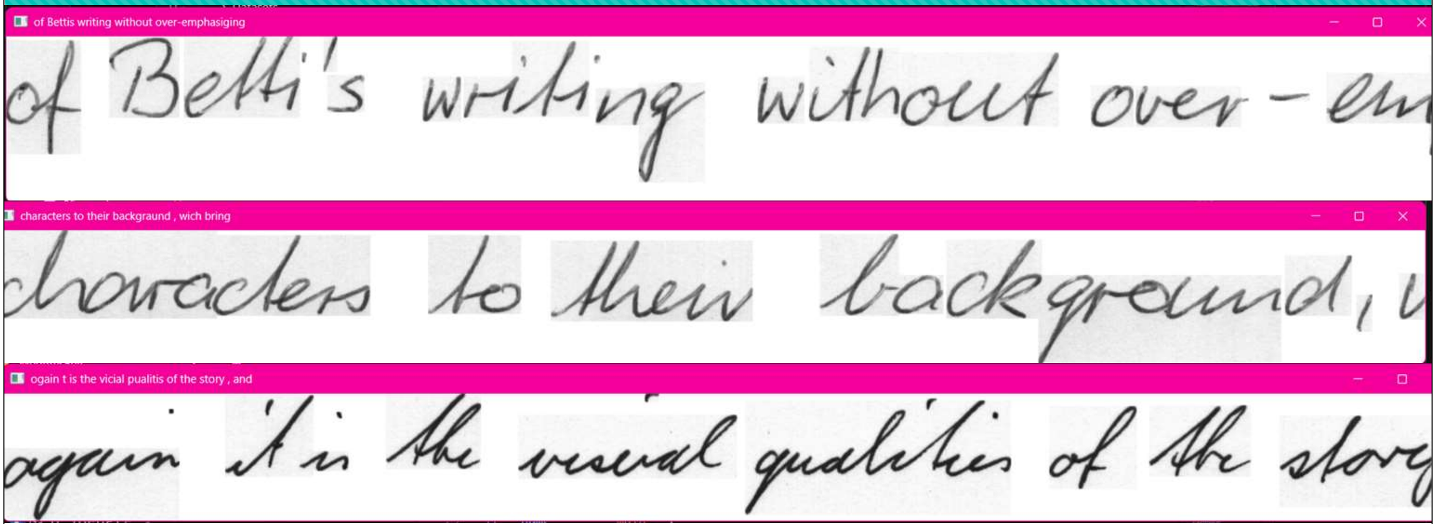
inferencemodel.py > ...

```python
import cv2
import typing
import numpy as np

from mltu.inferenceModel import OnnxInferenceModel
from mltu.utils.text_utils import ctc_decoder, get_cer, get_wer
from mltu.transformers import ImageResizer

class ImageToWordModel(OnnxInferenceModel):
    def __init__(self, char_list: typing.Union[str, list], *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.char_list = char_list

    def predict(self, image: np.ndarray):
        image = ImageResizer.resize_maintaining_aspect_ratio(image, *self.input_shapes[0][1:3][::-1])

        image_pred = np.expand_dims(image, axis=0).astype(np.float32)

        preds = self.model.run(self.output_names, {self.input_names[0]: image_pred})[0]

        text = ctc_decoder(preds, self.char_list)[0]

        return text

if __name__ == "__main__":
    import pandas as pd
    from tqdm import tqdm
    from mltu.configs import BaseModelConfigs

    configs = BaseModelConfigs.load("Models/04_sentence_recognition/202411142242/configs.yaml")

    model = ImageToWordModel(model_path=configs.model_path, char_list=configs.vocab)

    df = pd.read_csv("Models/04_sentence_recognition/202411142242/val.csv").values.tolist()
```

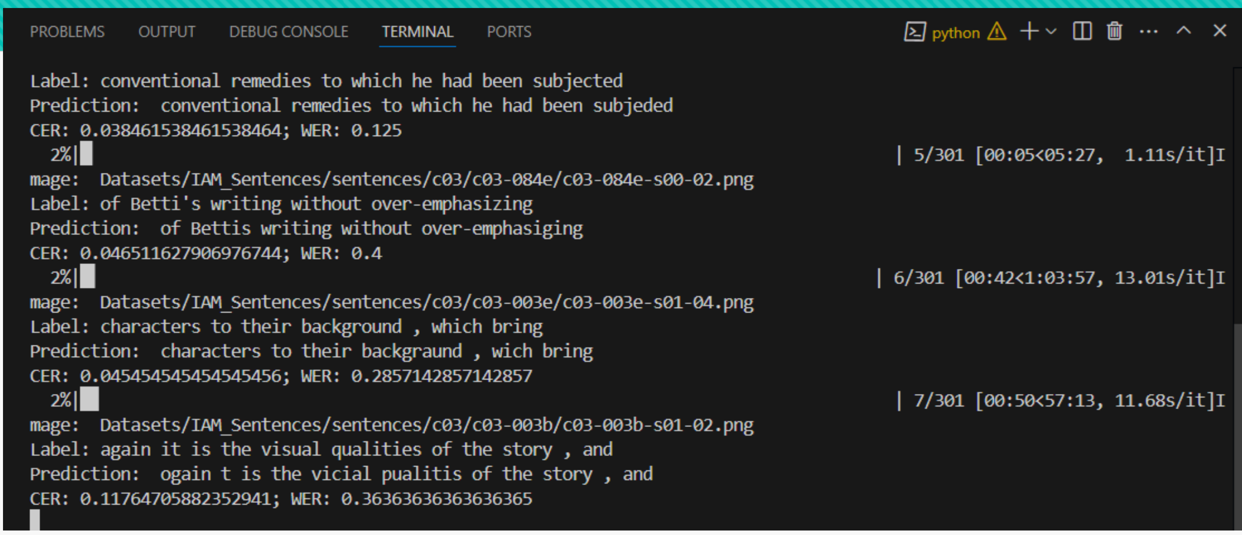

Inference Model

```python
accum_cer, accum_wer = [], []
for image_path, label in tqdm(df):
    image = cv2.imread(image_path.replace("\\", "/"))

    prediction_text = model.predict(image)

    cer = get_cer(prediction_text, label)
    wer = get_wer(prediction_text, label)
    print("Image: ", image_path)
    print("Label:", label)
    print("Prediction: ", prediction_text)
    print(f"CER: {cer}; WER: {wer}")

    accum_cer.append(cer)
    accum_wer.append(wer)

    cv2.imshow(prediction_text, image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

print(f"Average CER: {np.average(accum_cer)}, Average WER: {np.average(accum_wer)}")
```

Inference Model Outputs



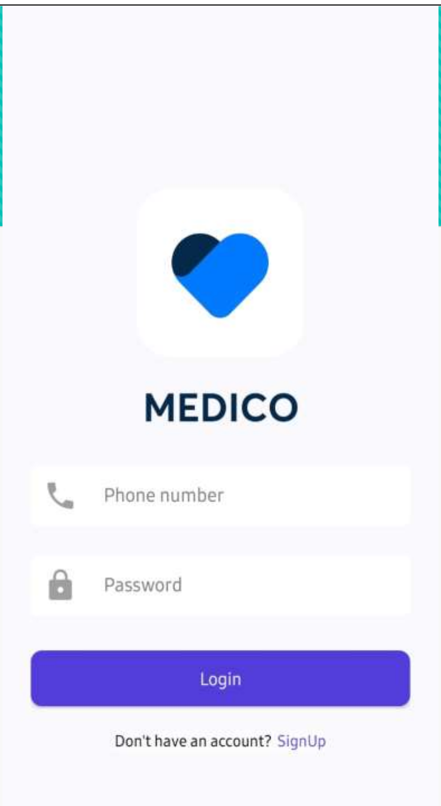
Inference Model Outputs



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS python

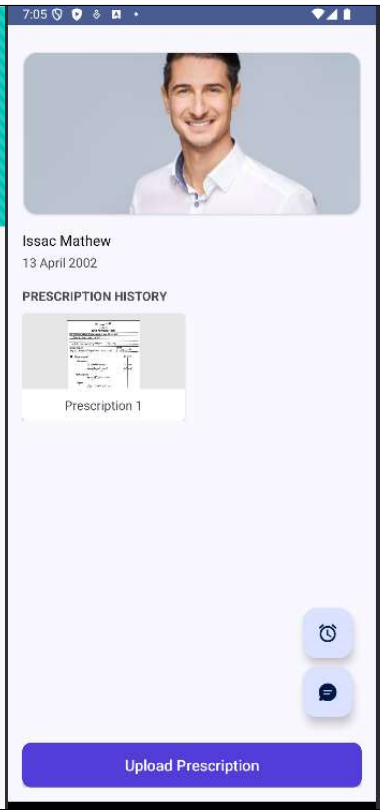
```
Label: conventional remedies to which he had been subjected
Prediction:  conventional remedies to which he had been subjeded
CER: 0.038461538461538464; WER: 0.125
  2%|█                                                    | 5/301 [00:05<05:27,  1.11s/it]I
mage:  Datasets/IAM_Sentences/sentences/c03/c03-084e/c03-084e-s00-02.png
Label: of Betti's writing without over-emphasizing
Prediction:  of Bettis writing without over-emphasiging
CER: 0.046511627906976744; WER: 0.4
  2%|█                                                  | 6/301 [00:42<1:03:57, 13.01s/it]I
mage:  Datasets/IAM_Sentences/sentences/c03/c03-003e/c03-003e-s01-04.png
Label: characters to their background , which bring
Prediction:  characters to their backgraund , wich bring
CER: 0.045454545454545456; WER: 0.2857142857142857
  2%|█                                                    | 7/301 [00:50<57:13, 11.68s/it]I
mage:  Datasets/IAM_Sentences/sentences/c03/c03-003b/c03-003b-s01-02.png
Label: again it is the visual qualities of the story , and
Prediction:  ogain t is the vicial pualitis of the story , and
CER: 0.11764705882352941; WER: 0.36363636363636365
```

Login Page

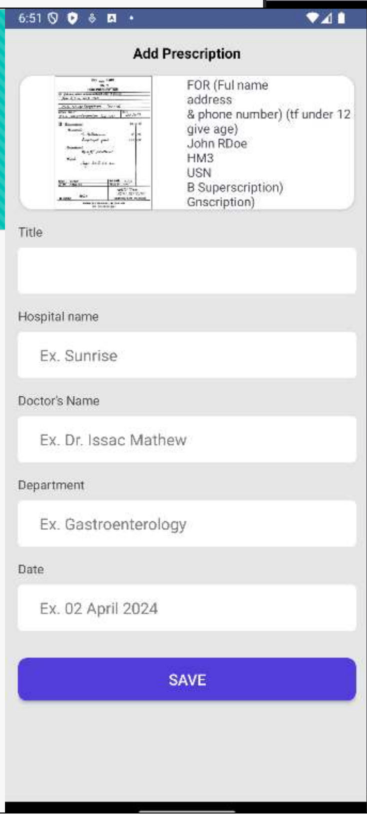


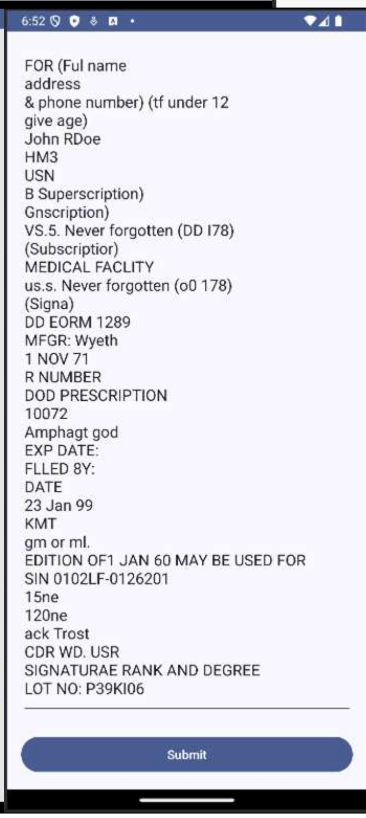
Patient Profile

Prescription History

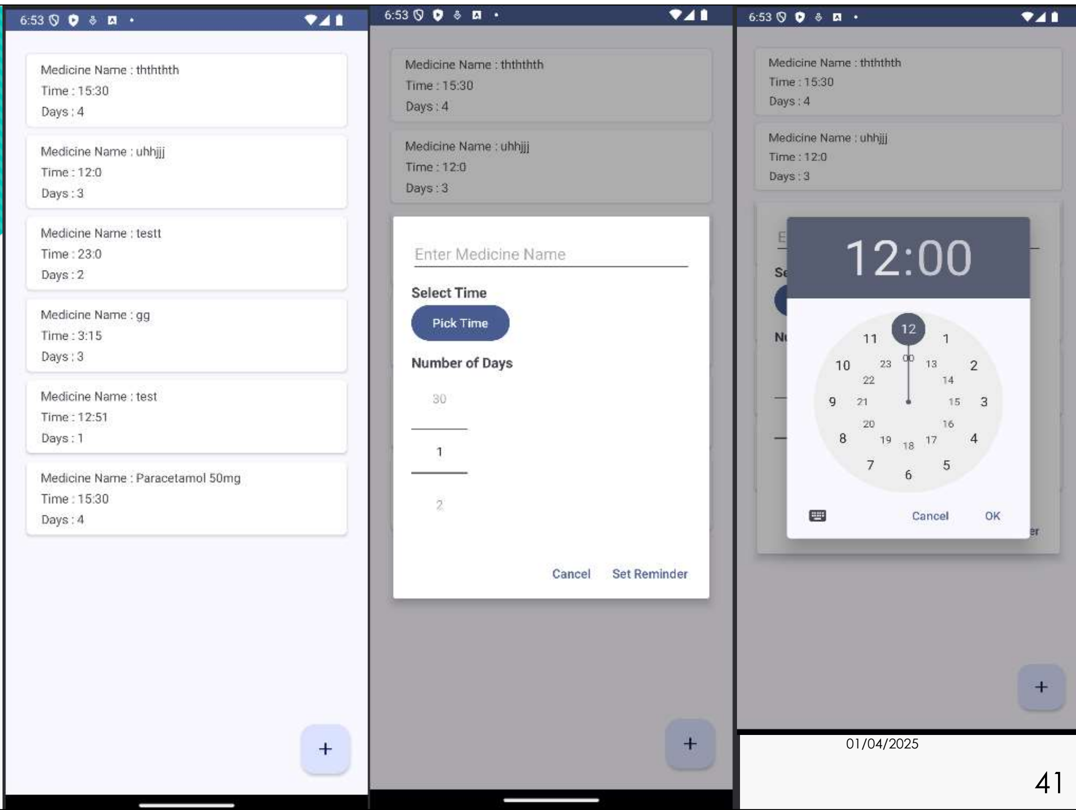


Prescription Upload

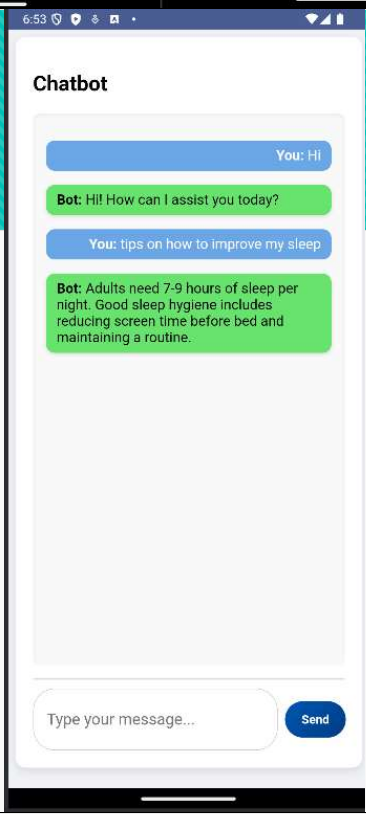




Timers and Reminders



Chat Bot



Work Division

- Issac: Model architecture design, Implementing residual blocks, Bidirectional LSTM layers, Model compilation and debugging.
- Madhav: Design and maintain a secure medication and symptom database, implement PHP scripts for communication between the mobile app, CRNN model, and chatbot with seamless frontend integration.
- Mathew: Develop and optimize the Medical Prescription Reading System's UI using Android Studio and JavaScript.
- Jerin: Develop chatbot that uses **cosine similarity** and **semantic search** in a **Django-based** system to provide over-the-counter medication recommendations with dynamic dataset updates.

Conclusion

In conclusion, the Medical Prescription Handwriting Recognition System effectively addresses the challenge of interpreting illegible handwritten prescriptions, ensuring accurate and quick conversion of scanned images into text. By leveraging a manually trained CRNN model for OCR, and OpenCV for image processing, the system enhances the reliability of prescription digitization. The integrated chatbot further supports users by providing valuable insights on diseases and suitable medications, making this solution a valuable tool for streamlining medical processes and improving patient care.

Future Scope

To enhance the system, OCR accuracy can be improved by adopting advanced models like attention-based CRNNs or Transformer-based OCR, and by training on domain-specific handwritten data.

A trained chatbot model can be integrated to generate more natural, conversational responses using fine-tuned transformer models on medical datasets.

Future work may also include expanding the dataset, adding multilingual support, and deploying the system as a mobile or cloud-based application for broader accessibility and real-time assistance.

REFERENCES

[1] S. Y. Manchala, J. Kinthali, K. Kotha, K. S. Kumar, and J. Jayalaxmi, "**Handwritten Text Recognition using Deep Learning with TensorFlow**," *International Journal of Engineering Research & Technology (IJERT)*, vol. 9, no. 5, pp. 594-597, May 2020.

[2] J. Doe, et al, "Handwritten Text Recognition using CNN-BiLSTM," International Journal of Engineering Research & Technology (IJĒRT), vol. 9, no. 4, pp. 234-239, 2023.

[3] X. Qian, L. Xie, N. Ye, R. Le, and S. Yang, "Pixel-Level Degradation for Text Image Super-Resolution and Recognition," *Electronics*, vol. 12, no. 4546, pp. 1-16, Nov. 2023, doi: https://doi.org/10.3390/electronics12214546.

[4] Hong-An Li, Jiangwen Fan, Keping Yu, Xin Qi, Zheng Wen, Qiaozhi Hua, Min Zhang and Qiaoxue Zheng "Medical Image Coloring based on GaborFiltering for Internet of Medical Things", IEEE Access, vol. 8, pp. 1-9, 2020, doi: https://ieeexplore.ieee.org/document/9106393

Appendix B: Vision, Mission, Programme Outcomes and Course Outcomes

Vision, Mission, Programme Outcomes and Course Outcomes

Institute Vision

To evolve into a premier technological institution, moulding eminent professionals with creative minds, innovative ideas and sound practical skill, and to shape a future where technology works for the enrichment of mankind.

Institute Mission

To impart state-of-the-art knowledge to individuals in various technological disciplines and to inculcate in them a high degree of social consciousness and human values, thereby enabling them to face the challenges of life with courage and conviction.

Department Vision

To become a centre of excellence in Computer Science and Engineering, moulding professionals catering to the research and professional needs of national and international organizations.

Department Mission

To inspire and nurture students, with up-to-date knowledge in Computer Science and Engineering, ethics, team spirit, leadership abilities, innovation and creativity to come out with solutions meeting societal needs.

Programme Outcomes (PO)

Engineering Graduates will be able to:

1. Engineering Knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. Conduct investigations of complex problems: Use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. Modern Tool Usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. Individual and Team work: Function effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings.

10. Communication: Communicate effectively with the engineering community and with society at large. Be able to comprehend and write effective reports documentation. Make effective presentations, and give and receive clear instructions.

11. Project management and finance: Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work, as a member and leader in a team. Manage projects in multidisciplinary environments.

12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.

Programme Specific Outcomes (PSO)

A graduate of the Computer Science and Engineering Program will demonstrate:

PSO1: Computer Science Specific Skills
The ability to identify, analyze and design solutions for complex engineering problems in multidisciplinary areas by understanding the core principles and concepts of computer science and thereby engage in national grand challenges.

PSO2: Programming and Software Development Skills
The ability to acquire programming efficiency by designing algorithms and applying standard practices in software project development to deliver quality software products meeting the demands of the industry.

PSO3: Professional Skills
The ability to apply the fundamentals of computer science in competitive research and to develop innovative products to meet the societal needs thereby evolving as an eminent researcher and entrepreneur.

Course Outcomes (CO)

Course Outcome 1: Model and solve real world problems by applying knowledge across domains (Cognitive knowledge level: Apply).

Course Outcome 2: Develop products, processes or technologies for sustainable and socially relevant applications (Cognitive knowledge level: Apply).

Course Outcome 3: Function effectively as an individual and as a leader in diverse teams and to comprehend and execute designated tasks (Cognitive knowledge level: Apply).

Course Outcome 4: Plan and execute tasks utilizing available resources within timelines, following ethical and professional norms (Cognitive knowledge level: Apply).

Course Outcome 5: Identify technology/research gaps and propose innovative/creative solutions (Cognitive knowledge level: Analyze).

Course Outcome 6: Organize and communicate technical and scientific findings effectively in written and oral forms (Cognitive knowledge level: Apply). [5] [6] [7] [8] [9] [10]

Appendix C: CO-PO-PSO Mapping

	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO 3
CO 1	2	2	2	1	2	2	2	1	1	1	1	2	3		
CO 2	2	2	2		1	3	3	1	1		1	1		2	
CO 3									3	2	2	1			3
CO 4					2			3	2	2	3	2			3
CO 5	2	3	3	1	2							1	3		
CO 6					2			2	2	3	1	1			3

3/2/1: high/medium/low

Figure 5.1: CO-PO and CO-PSO Mapping Table