Enhanced Sign Language Recognition using Deep Learning And Computer Vision

By

Yasmin Hammad 32009304029

Layan Bilbeisi 32109304001

Malak Musameh 32009304010

Supervised By

Dr. Qusay Al-Zubi

This Project (**Graduation Project I/II**) is Submitted in Partial Fulfillment of the Requirements for Bachelor Degree in Data Science

Faculty of Artificial Intelligence

Al-Balqa Applied University

Al-Salt- Jordan

Fall, 2023

DEDICATION

We dedicate this achievement to those we cherish most in our hearts, our mothers, fathers, families, and loved ones, who have supported us with love and effort throughout this process. We are grateful for their unwavering support and belief in us, which has been instrumental in making this project a success. We would also like to express our appreciation to all those who have inspired us to pursue this project. Your passion and dedication to your work have been a constant source of motivation for us. We hope that this project will serve as a testament to the hard work and dedication of everyone involved. Thank you all for your contributions, support, and inspiration. This achievement would not have been possible without you.

ACKNOWLEDGMENTS

We would like to express our sincere gratitude to the director of this project, Dr Qusay Al-Zubi for his continuous support, patience, motivation, and immense knowledge. His guidance helped us throughout the research and writing process of this project. He has helped us sharpen our critical thinking and research skills and dedicated his time and effort to help us continue our research. He always gave us useful suggestions to improve our research and writing skills and encouraged us whenever we were in doubt. His continuous dedication to his students is both admirable and inspirational. We would also like to thank all the individuals who have contributed to the creation of this project. Your hard work and commitment to excellence have been invaluable in making this project informative and useful.

Finally, we would like to express our appreciation to all those who have supported us throughout this process. Your encouragement and guidance have been instrumental in helping us achieve our goals. We are grateful for your unwavering support and belief in us.

TABLE OF CONTENTS

List Of Tables

List Of Figures

LIST OF ABBRREVIATIONS

CNN	Convolutional Neural Networks
LSTM	Long Short-Term Memory
ASL	American Sign Language
BDSL	Bengali Sign Language
SVM	Support Vector Machines
NN	Neural Networks
HMMs	Hidden Markov Models
DNN	Deep Neural Network
YOLO	You Only Look Once
NLP	Natural Language Processing
SGD	Stochastic Gradient Descent
Sklearn	scikit-learn
GRU	Gated Recurrent Unit
ReLU	Rectified Linear Unit

ABSTRACT

Sign language is a unique form of communication used by deaf individuals, but the lack of understanding among the general population creates a significant communication gap. This project aims to develop a sign language translator that can capture real-time gestures and convert them into textual output or speech. The system utilizes a hybrid approach combining Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) algorithms. It also has the capability to accept text or speech input and transform it into 3D animations of sign language which we built using Blender. By leveraging advanced technologies, this project aims to bridge the communication gap and promote inclusivity between sign language users and non-sign language users. The proposed system comprises five main stages: Data Preprocessing, Feature Extraction, Hand Gesture Classification, Translation, and Output Generation. The system's accuracy is evaluated to ensure reliable translation output.

Chapter One

Introduction

1.1 Overview

Sign language is a widely used method of communication among Deaf and hard-of-hearing communities around the world. It is estimated that there are over 70 million deaf people worldwide who use sign language as their primary means of communication. Despite its widespread use, sign language recognition remains a challenging area of research, with numerous applications in fields such as education, communication, and accessibility. To contribute to this important area of research, our project aims to develop a robust sign language recognition system that utilizes computer vision techniques and machine learning algorithms to accurately translate sign language gestures into text, and vice versa. The system will also integrate with speech recognition and translation APIs to provide both textual and spoken input and will focus on transforming recognized gestures into 3D animated representations for enhanced visual communication. By developing an accurate and efficient sign language recognition system, we hope to contribute to the advancement of communication accessibility and inclusion using deep learning techniques and computer vision.

1.2 Problems

Developing a robust sign language recognition system involves overcoming several challenges. Real-time processing is a key challenge, requiring efficient algorithms and hardware to accurately interpret dynamic and expressive gestures **(Koller et al., 2019).** Ambiguity in gestures adds complexity, as multiple meanings and variations exist depending on context **(Starner et al., 2000)**. Limited training data poses another challenge, making it necessary to

address gaps in recognition capabilities **(Camgoz et al., 2018)**. Additionally, the system must adapt to various environments, accounting for lighting conditions, background clutter, and user positioning. Overcoming these challenges is crucial to creating an effective sign language recognition system that enhances communication accessibility and inclusivity. Through extensive research in deep learning techniques, computer vision, and machine learning algorithms, we can pave the way for enhanced communication experiences for individuals who use sign language.

1.3 Objectives

The aim of this research is:

1. Develop a real-time sign language recognition system that can accurately identify and interpret a wide range of sign gestures.
2. Implement a hybrid architecture combining Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) models to effectively capture spatial and temporal features of the sign gestures.
3. Integrate speech recognition and translation libraries to enable the system to convert text or speech into sign gestures, and from sign gesture to text.
4. Develop function to enable us to transform the recognized sign gestures into 3D animated representations for enhanced visual communication.

1.4 Contribution

Our research addresses the critical need for effective communication with individuals who use sign language as their primary means of expression. We present a comprehensive contribution comprising the development of a sign language translation system that encompasses multiple innovative objectives.

1. We developed a real-time sign language recognition system that accurately identifies and interprets a wide range of sign gestures.
2. Our system uses a hybrid architecture that combines Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) models to capture both spatial and temporal features of sign gestures.
3. We have integrated speech recognition and translation libraries to enable the system to convert recognized sign gestures into text, and vice versa.
4. Lastly, we have developed it so that it transforms written text/speech into 3D animated representations for enhanced visual communication.

1.5 Scope

Scope Section	Description
Data Set	The dataset will include images and videos with variations in angles, lighting conditions, and other factors to ensure robust model training
Preprocessing Techniques	Preprocessing techniques, such as image normalization, noise reduction, and background removal, will be applied
Deep Learning Architecture	A novel hybrid architecture combining CNNs, and LSTMs will be developed to capture spatial features of gestures and temporal dependencies.
Speech Recognition	The PyAudio library will be used for speech recognition, converting spoken language into text
3D Sign Animation	Blender will create 3D animated representations of recognized sign gestures.

Table 1.1 Scope

1.6 Organization

The rest of project documentation is organized as follows:

Chapter 2: Related Work This chapter provides an overview of the work related to the project topic, including an analysis of previous efforts in the field of sign language processing and recognition.

Chapter 3: Research Methodology This chapter is divided into several sections:

- Section 3.1 introduces the specific requirements of the project, outlining the essential functionalities and features that the sign language translating system (SLTS) aims to incorporate.
- Section 3.2, Project Modeling, details the methodology and techniques employed, emphasizing the integration of deep learning, computer vision, Natural Language Processing (NLP), and the PyAudio library to develop an innovative sign language translator.

Chapter 4: Project Design & Implementation This chapter consists of two sections:

- Section 4.1 introduction about the chapter

- Section 4.2 covers the design of screens and the features they are expected to encompass, providing a visual representation of the user interface and its functionalities.

- Section 4.3 focuses on the programming language chosen for the implementation of the project, detailing the technical aspects of the system's development.

Chapter 5: Conclusion and Future Work This chapter offers a concise summary of the project's key contributions and outlines the planned objectives for the subsequent phase of the graduation project.

Chapter Two

Related Work

2.1 Introduction

Creating an effective sign language processing system necessitates a profound grasp of both Deaf culture and sign language itself. This dual understanding is crucial for developing systems that genuinely resonate with user preferences and aspirations, while also accommodating the intricate linguistic nuances inherent in sign languages. In the following sections, we delve into this foundational knowledge and address the limitations of existing sign language processing reviews, which often fall short of providing a holistic perspective on this intricate challenge.

2.2 Sign Language Translator

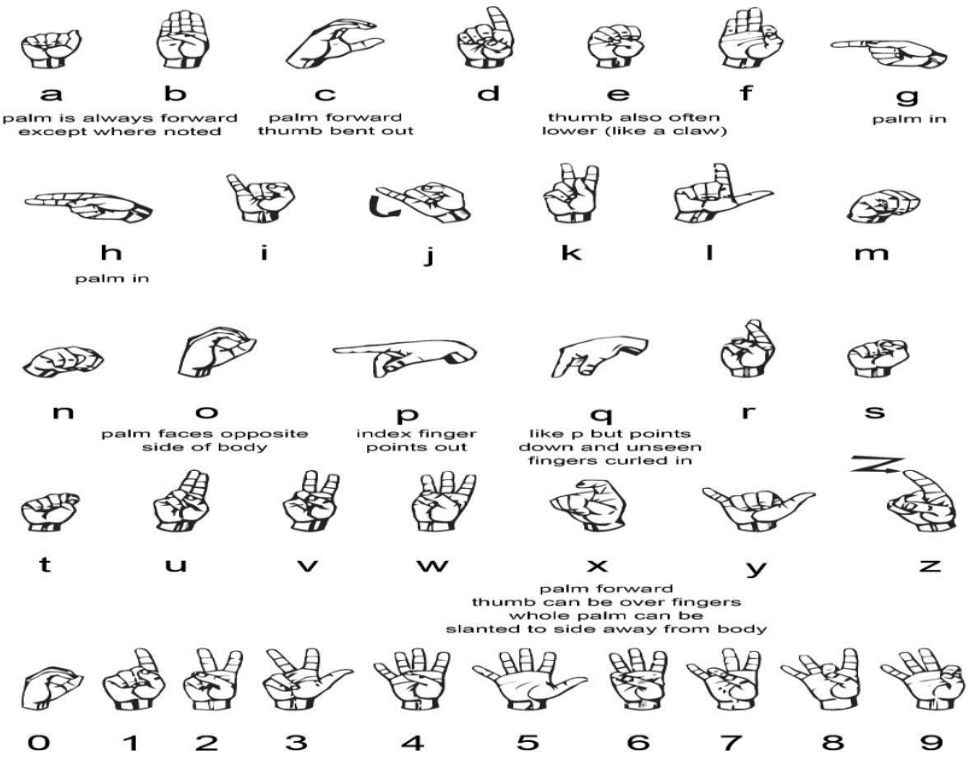

Figure 2.1 American Sign Language Hand Gesture

Sign language recognition can be approached using two methods: hardware and software. However, hardware devices like gloves or sensors can be expensive and inaccessible for many individuals. These devices capture and interpret hand movements, and algorithms are developed to analyze the sensor data in real-time. One notable system proposed by **(Saquib, 2017)** utilizes wearable sensors integrated into data gloves to accurately detect both static and dynamic symbols in American Sign Language (ASL) and Bengali Sign Language (BDSL). The development process involves identifying the necessary sensors and their suitable positions in the gloves, constructing the glove itself, collecting a significant amount of data, training a machine learning model to classify the captured data, and finally determining intelligible messages from a stream of sensor data. While hardware-based solutions like this have shown impressive capabilities in recognizing sign language, it is important to consider alternative approaches that may address the limitations of cost and accessibility constraints. The second approach, software-based solutions, includes gesture recognition using Support Vector Machines (SVM), Neural Networks (NN), Hidden Markov Models (HMMs), etc. Software-based solutions require image processing before classifying gesture images. Computer vision techniques are used to capture video sequences of sign language gestures, which are then processed to extract relevant features. These features are subsequently used to train machine learning models for gesture recognition. **(Razieh Rastgoo, 2021)** highlights the complexity of sign language recognition, breaking it down into five crucial parameters: hand shape, palm orientation, movement, location, and expression. These parameters collectively define the precision of sign language communication. The study emphasizes the challenge of devising systems that transform sign language into text or voice, with the intention of bridging communication gaps between the hearing-impaired and hearing communities. **(Rivera-Acosta et al., 2021)** developed an innovative approach to tackle these challenges in recognizing

hand gestures, particularly within the realm of American Sign Language (ASL), using static images. Their model consists of two pivotal stages: a deep neural network (DNN) for precise hand shape segmentation and classification, and a bidirectional long short-term memory (LSTM) network for accurate spelling correction. **(Natarajan B., 2022)** constructed a comprehensive framework for recognizing, translating, and synthesizing sign language using pose details and text generation. They employed a hybrid convolutional neural network with a bidirectional long-term memory component and integrated technologies like MediaPipe and a dynamic generative adversarial network model. In recent years, significant progress has been made in the field of generating sign language from speech/text input**. (M. Kavya Mounika.,2022)** proposed a comprehensive four-step process for speech/text to sign language conversion. This process involves capturing audio input using PyAudio, converting audio to text using the JavaScript web speech API with text tokenization and NLP techniques, converting text to visual sign language using NLP algorithms and a video dataset sign language library, and finally merging matching videos to display the final sign language output. The methodology includes capturing input voice, converting it to text, removing irrelevant parts using NLP, mapping each word/character to a visual sign word library, and linking the videos to create a cohesive video that renders all the sign language text on the final display. **(Chakladar et al., 2021)** presents a unique method known as the 3D Avatar Approach for Continuous Sign Movement Using Speech/Text. This groundbreaking technique aims to bridge the gap between sign language and spoken language by leveraging advanced technologies. The 3D Avatar Approach offers a fresh perspective on Sign language movements, enabling a more efficient and accurate translation process. By employing cutting-edge algorithms and sophisticated machine learning models, this approach opens new possibilities for communication and inclusion for individuals who rely on sign language.

2.3 literature review

Name	Techniques	Advantage	Disadvantage
Nazmus Saquib and Ashikur Rahman (2017)	Data Gloves, SVM, KNN, Random Forest	- Achieved Accuracy: Impressive 87.5%. - Effectiveness of NLP: Shows how NLP helps bridge communication gaps. - Targeted Approach: Focuses on various language styles. - Benefit: Helps the hearing-impaired by improving communication.	- Limited Generalization: High accuracy may not extend to varied sign language variations. - Scalability: Unexplored scalability for larger vocabularies or complex structures. - Real-Time Performance: Lack of real-time system performance information. - User Interface: Potential usability issues for signers and non-signers.
Rastgoo's (2021)	CNN	- Comprehensive Resource: Covers many techniques and advancements. - Useful for Researchers: Offers valuable insights for their work in communication. - Focus: Enhancing communication for the deaf and hard-of-hearing.	- Model Choice: Single deep learning model used. - Accuracy: Attained 76.7%. - Comparison: Falls short against other methods. - Improvement Need: Emphasizes complex models and better training.
Miguel (2021)	DNN+LSTM	- The system employs a YOLO (You Only Look Once) network and LSTM (Long Short-Term Memory) models. - The primary goal is to enhance communication inclusivity for the hearing-impaired community.	- Insufficient Data: May lead to reduced model robustness. - Confusion between Similar Letters: Similar gestures could result in translation inaccuracies.

Natarajan B (2022)	MediaPipe+ CNN+ Bi-LSTM	- Holistic approach: integrates sign language recognition, translation, and video generation.	Resource Intensive Framework: The integrated approach may require significant computational resources, potentially limiting real-time performance and accessibility.
M. Kavya Mounika (2022)	PyAudio+ NLP	- Bridges language gaps enable communication between spoken and sign languages.	Limited Vocabulary Handling
Debashis Das Chakladar (2021)	3D avatar	- Achievement: 87.5% accuracy in "Speech/Text to Sign Language Converter" development. - Effective NLP: Bridges communication gaps using Natural Language Processing. - Impact: Benefits hearing-impaired with improved cross-modal communication.	Problem with the transition between signs while performing a sign sentence

Tabel 2.1 Literature Review

Based on our research, we will utilize CNN and LSTM models to build an enhanced sign language recognition system. The CNN model will extract features from sign language videos, while the LSTM model will capture temporal dependencies between frames. We will integrate Speech API to enable speech recognition and translation. Blender will be employed to create realistic 3D animations of sign language gestures, empowering individuals with hearing disabilities and fostering inclusivity in society.

CHAPTER 3

Research Methodology

3.1 Introduction

In the preceding chapters, we explored the realm of sign language translation, uncovering the foundational tools and methods that drive this revolutionary field. Now, as we step into Chapter 3, we embark on a journey deep into the core of our graduation project. This chapter unveils the intricate research methodology we've employed to attain our primary objectives. It functions as a compass, steering us towards the creation of an innovative sign language translator. This creation draws its strength from the fusion of deep learning, computer vision, Natural Language Processing (NLP), and the PyAudio library.

The research methodology gets even more interesting with LSTM and CNN methods. LSTM helps us understand patterns over time, and CNN lets us dive into visual information. By blending these techniques with others.

Just as a map guides explorers through unknown places, Chapter 3 guides you through our project's core. In Section 3.2, we lay out the main plan and what we need. In Section 3.3, we explain how everything comes together in stages. And in Section 3.4, we wrap it up, summarizing what we've shared.

3.1.1 Use Case Diagram.

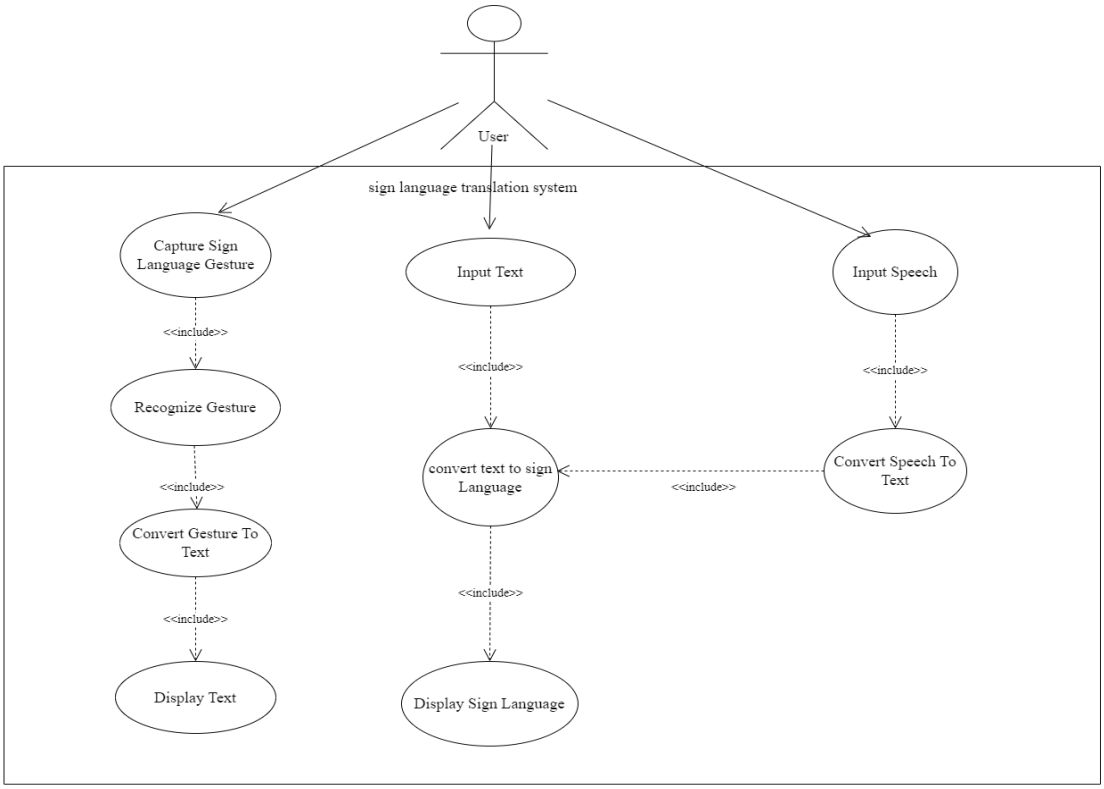

Figure 3.1 Use Case Diagram

Describes the tasks that Sign Language Translator will perform, Figure 3.1 shows a complex language processing system that can handle input from web cameras, text, and speech. It includes interconnected nodes that represent different components of the system. The system receives language data from various sources like web cameras, text inputs, or speech inputs. The data flows through language processing tools that analyze and process it, regardless of the input type. For example, image recognition modules handle visual data, speech recognition modules handle spoken language, and natural language understanding modules handle the output of the speech recognition API and textual data. The system can include an animated representation of sign language, which takes textual or speech input and processes it to generate

an animated representation of the signs. The gesture recognition component analyzes visual data, such as hand movements captured by the web camera input system, to identify and recognize specific gestures associated with sign language. After gesture recognition, the processed data continues to move through the system, passing through components that further refine and analyze it. Finally, the output systems take the processed language data and generate meaningful outputs based on the input type. The diagram showcases the complexity of the language processing system and its ability to handle different input types, including the animated representation of sign language. The interconnected components work together to process and analyze language data, enabling the system to understand and generate human language in various forms.

3.1.2 Class Diagram

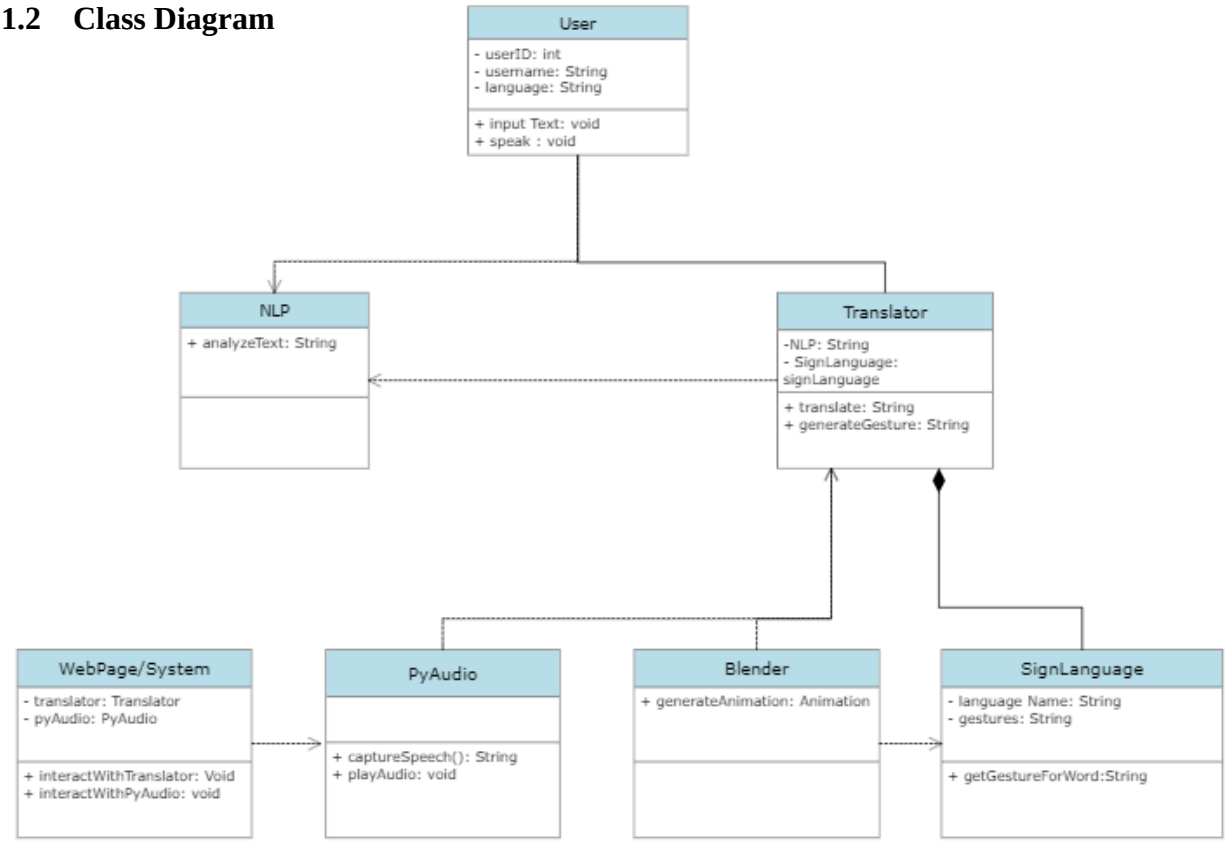

Figure 3.2 Class Diagram

In Figure 3.2, a class diagram illustrates the architecture of a Sign Language Translation system, a tool designed to bridge communication gaps. The system is composed of several classes, each with a unique role, 'User' This is the interface for individuals to interact with the system. It stores information such as the username and handles text inputs and speech. 'Translator' This class uses functions to map words to sign language gestures. 'SignLanguage' This class holds the visual expressions of sign language, including mappings from gesture to words, which are essential for effective translation. 'NLP' Specializing in text analysis, this class employs natural language processing techniques to ensure accurate interpretation of user inputs. 'Blender' This class is responsible for generating animations that convert text into dynamic visual representations, enhancing both understanding and learning. 'WebPage/System' This class serves as the interface where users input text and receive translations and animated visualizations, integrating various communication methods. Each class is interconnected, working together to provide a seamless sign language translation experience. The system is robust and adaptable, capable of handling a variety of inputs and producing accurate, understandable outputs. It's a powerful tool for bridging communication gaps and making sign language more accessible.

3.1.3 Flowchart of the Sign Language Recognition Process

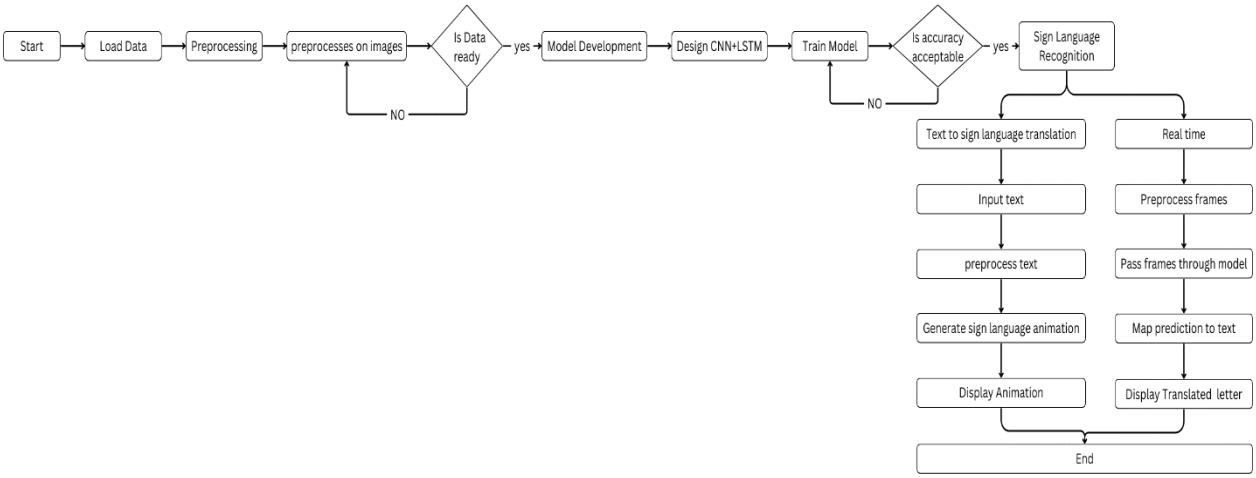

Figure 3.3 Model workflow

In Figure 3.3 the workflow represents a process for developing a real-time sign language recognition system. It begins with loading and preprocessing image data, which is then used to train a Convolutional Neural Network Long Short-Term Memory (CNN-LSTM) model. The model is trained until it reaches an acceptable level of accuracy. Once the model is ready, it is used for real-time sign language recognition. This involves translating text to sign language and vice versa. For text to sign language translation, the input text is preprocessed and used to generate a sign language animation. For sign language to text translation, video frames are preprocessed and passed through the model, which then maps the predictions into text.

3.1.4 Flowchart of Input Handling and Processing in GUI

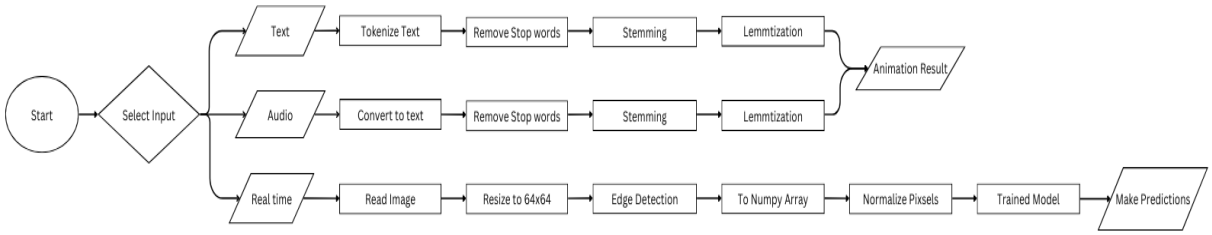

Figure 3.4 Input workflow

In Figure 3.4 The workflow outlines an overall process for generating an animation result from different types of input data. For text input, the process involves tokenizing the text, removing stop words, and applying stemming and lemmatization. If the input is audio, it's first converted to text and then undergoes the same processing as text input lastly it is converted into an animation. For real-time or image input, the image is resized, edge detection is applied, and it's converted into a normalized NumPy array. This array is then fed into a trained model to make predictions.

3.2 Overview of the proposed SLTS

To achieve the research goals mentioned in Chapter one (refer to 1.5), the proposed approach's flow is presented in this section. Figure 3.5 illustrates the overall stages of the proposed sign language translator and shows the relationship between the three stages and their roles in achieving the research's main objectives.

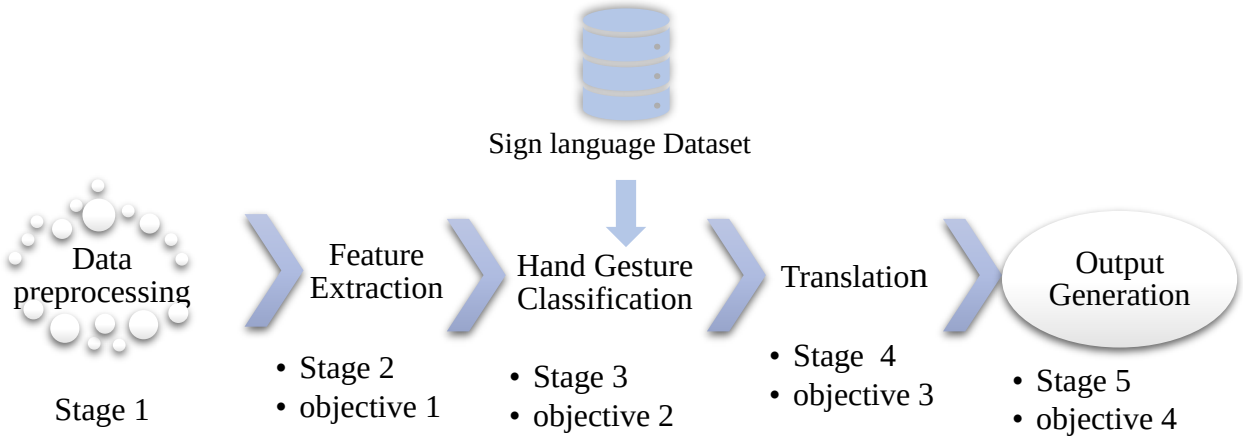

Figure 3.5 General Architecture of Sign Language Translator

As shown in Figure 3.5, a breakdown of the sign language translator steps, which is made up of five stages. We'll start our discussion with the first stage, known as preprocessing:

3.2.1 Stage One (Data Preprocessing)

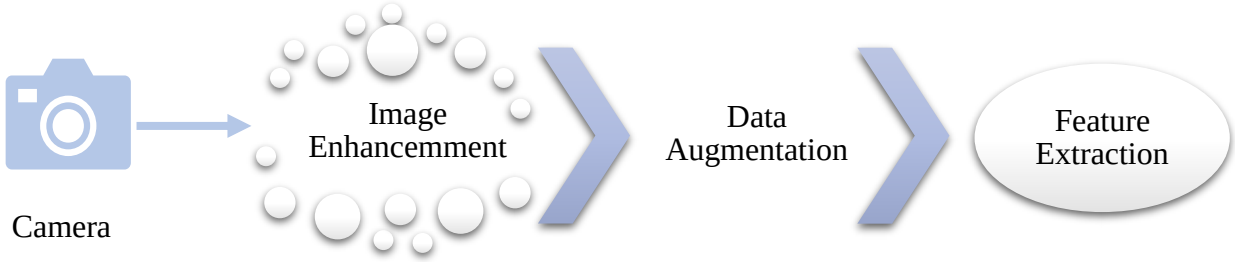

Figure 3.6 Preprocessing Stage Steps

3.2.1(a) Image Enhancement

Image enhancement is a way of improving an image's visual appeal. Techniques like contrast stretching, noise reduction, and edge detection can help to reveal hidden details and make it easier for machine vision systems to extract relevant data from images.

There are many different methods and techniques available for image enhancement, and the best way to utilize them will depend on the specific qualities of the image and the intended results, by these methods we can maximize the accuracy and performance of computer vision systems:

1. Grayscale conversion**:** Converting an image to grayscale eliminates color information and represents the image using shades of gray. This simplifies the image data, reduces computational complexity, and enhances visualization. Grayscale conversion reduces the image to its essential pixel intensities.
2. Normalization: Normalizing the pixel values of an image involves scaling them to a specific range, such as 0 to 1. This ensures consistency in pixel intensity values across all images, making the image easier to process. Normalization can improve the performance of subsequent processing and classification algorithms by ensuring that no range of pixel intensities dominates others.
3. Image standardization: Image resizing, and standardization involve resizing the input image to a standardized size. This ensures consistency in dimensions across all images, which is important for further processing and classification. Standardizing the image size helps in comparing and analyzing the signs accurately, as all images will have the same spatial resolution.
4. Noise reduction: Applying noise reduction techniques, such as Gaussian blur, helps to reduce random variations or ‘noise’ in pixel values that could hinder sign recognition. Noise reduction

enhances the clarity of the image and improves the accuracy of subsequent processing steps by reducing the impact of pixel intensity variations that are not relevant to the sign shapes.

5. Edge detection: Edge detection techniques, such as adaptive thresholding, are used to identify and highlight the boundaries of the signs in an image. By calculating different threshold values for different regions of the image, it adapts to varying lighting conditions and contrasts. This helps in distinguishing the signs from the background, making them easier to identify and recognize.

3.2.1(b) Data Augmentation

The last step is a process of artificially increasing the amount of data by generating new data points from existing data. This technique is useful when there is limited data available and can improve the performance of machine learning models. In sign recognition, data augmentation involves creating modified copies of the dataset using existing data. This technique can be used to artificially expand the size of a training set and prevent overfitting. Techniques for data augmentation include geometric transformations and color space transformations.

3.2.2 Stage Two (Feature Extraction)

Feature extraction is a crucial stage in sign recognition that involves identifying important features or attributes of the data. This is performed using Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks in the model. The CNN layers act as automatic feature extractors. They learn to identify important features such as edges, shapes, and textures in sign language images during training when the model weights are updated to minimize the loss function. After the convolutional layers, the model is reshaped and passed through a bidirectional

LSTM layer. Contrary to analyzing sequences of frames, this LSTM layer in this model is used to process the spatial features extracted by the CNN layers from individual images. This can be particularly useful for recognizing signs that are defined by specific spatial configurations of hand and finger movements. The extracted features are then used for sign classification or recognition. The goal of feature extraction in this context is to convert raw image data into a form that can be processed by the machine learning model, reducing dimensionality and computational complexity. By utilizing these techniques, we can improve the accuracy and performance of machine learning models in sign recognition.

3.2.3 Stage three (Hand Gesture Classification)

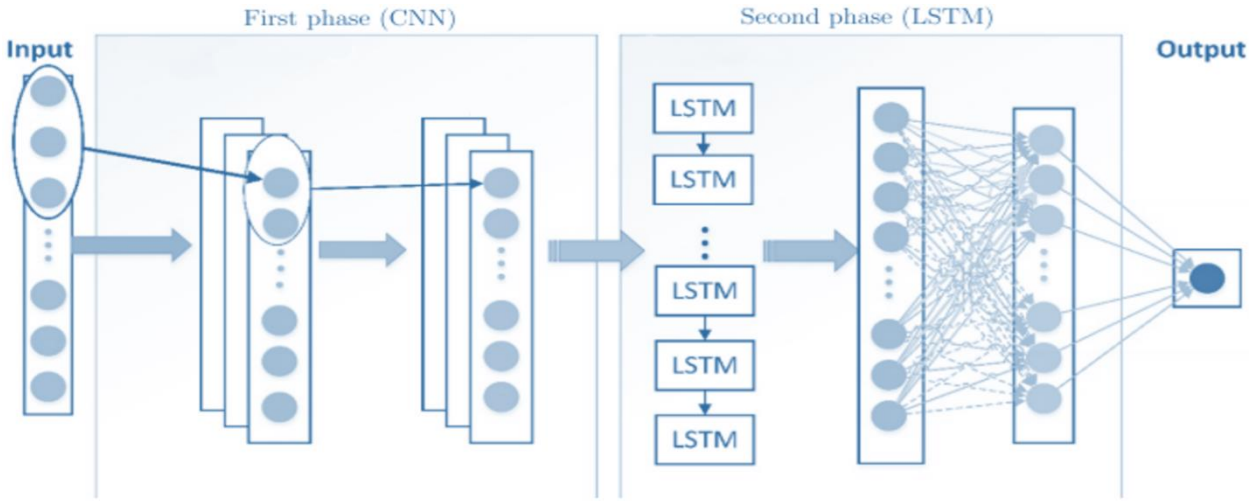

Figure 3.7 Hybrid CNN-LSTM neural network internal architecture

This is the final stage where the actual prediction of the hand gesture is made. This is handled by the dense layer in the model where it outputs the probabilities of each class (hand gestures) and the class with the highest probability is chosen as the prediction.

By utilizing these techniques, we can improve the accuracy and performance of machine learning models in sign recognition.

3.2.4 Stage Four (Translation)

One thing to keep in mind is that sign language translation is not just about converting visual gestures into text or speech. It is also about preserving the cultural and linguistic nuances of sign language. Sign languages have their own grammar, syntax, and vocabulary. Sign language translation is still a developing field, and there are challenges that need to be addressed, such as the accuracy of translation and the availability of data. However, the potential benefits of sign language translation are significant. It has the power to bridge communication gaps and foster understanding and inclusivity between sign language users and speakers. In the translation stage of sign language communication, the system can either produce output in text and speech or take input speech using PyAudio for speech recognition. The input speech is processed using speech recognition techniques to convert it into text. This text can then be further processed using Natural Language Processing (NLP) techniques to understand the meaning and context of the input. This technology has the potential to break down barriers and promote inclusivity in communication. Overall, sign language translation is a complex and evolving field that requires careful consideration of linguistic and cultural aspects. By leveraging advancements in speech recognition, NLP, and synthesis technologies, we can strive towards accurate and meaningful translation in sign language communication.

3.2.5 Stage Five (Output Generation)

And lastly the model can convert spoken language into text or use text to create animations of sign language gestures. This model has the potential to make communication more accessible and inclusive for everyone, and it can be used by people who are deaf or hard of hearing, and people who are learning sign language.

3.3 Requirements of the Sign Language Translating System

Based on the objectives mentioned in chapter one, the following requirements must be validating the thesis's findings:

1. the data set must contain a collection of images and videos that have different lighting conditions and other various cases.

2. SLTS should recognize hand gestures and interpret them into text and vice versa.

3. SLTS should involve converting both text and speech, transcribing spoken words into text, and then synchronizing them with a 3D animated sign gesture.

4. SLTS will use CNN and LSTM models. The CNN model will capture features of gestures while LSTM will capture temporal dependencies**.**

3.4 Summary

In this chapter, we've looked closely at the methods and key parts that help us reach our main goal. We've put a lot of effort into improving methods, specially making the most of LSTM and CNN techniques. We've used Natural Language Processing, PyAudio. This mix of techniques has made our understanding deeper and opened a door for more improvement. This journey shows how bringing different techniques together can help achieve a common goal.

Chapter Four
Project Design & Implementation

4.1 Introduction

This chapter presents the design and implementation of the proposed Sign language recognition system. Section 4.2 presents the project design and 4.3 discusses the implementation and experiment environments.

4.2 The GUI Design

Figure 4.1 shows the entry screen which appear upon launching the application, users encounter the sign-in page, where the users are prompted to input their credentials, specifically a username and password. Once the user enters this information, they click the "Login " button.

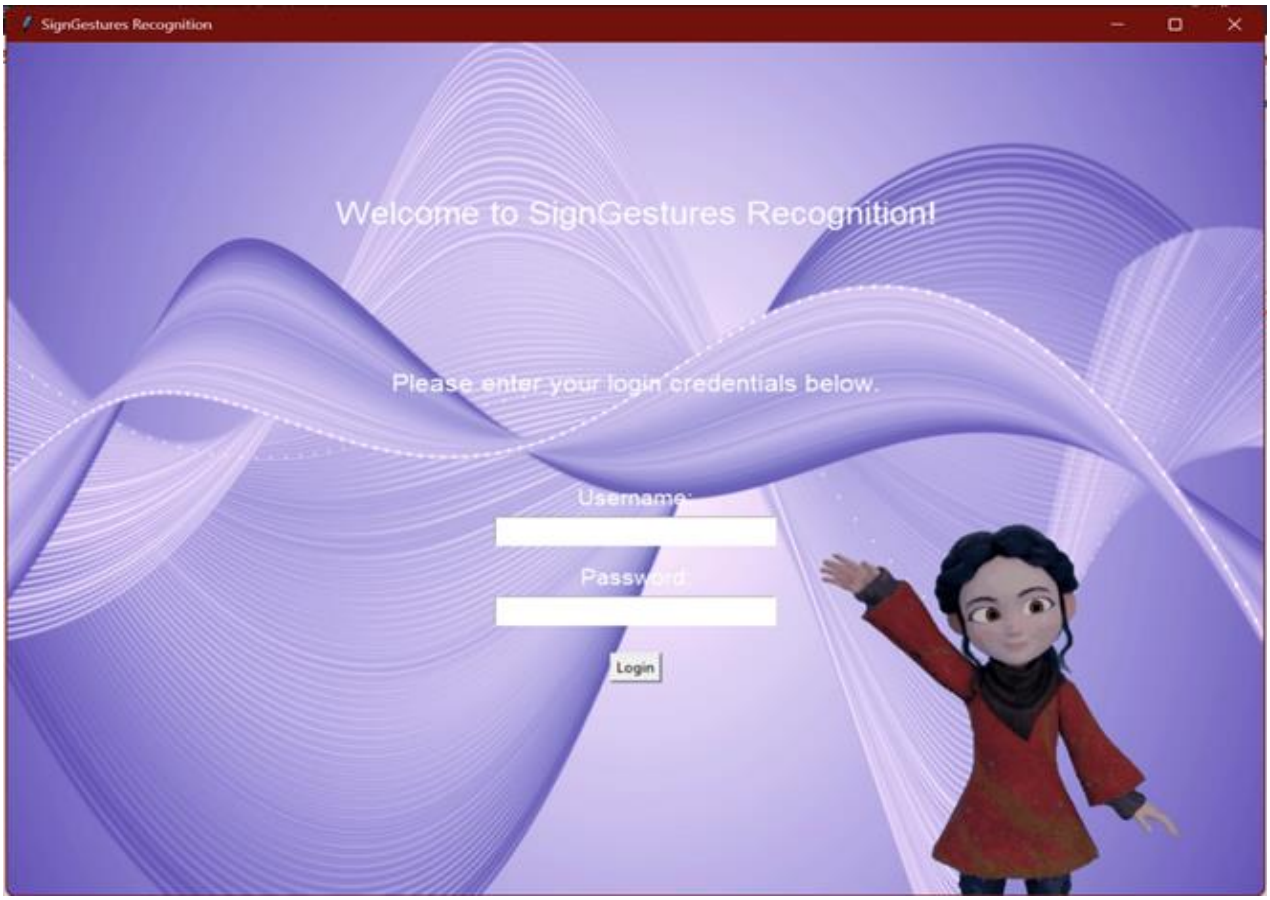

Figure 4.1 Login screen

When successful Logging in, the Figure 4.2 displays the Input choices for the user to choose between text, audio, or real time input. The user proceeds by clicking on one of these options based on what they want to use.

Figure 4.2 Input Options

In First case, that the user choses the text input, the application displays a page where the user can input text. Following the text input, the user clicks a Submit button. The text processed based on the predefined text processing steps. Subsequently the application displays preview the recognized sign language symbols as previewed in figure 4.3.

Figure 4.3 Input text

If user selects audio as an input option, the application displays a page where the user can start recording. The user interacts with the audio input interface and clicks a stop record button. The recorded audio is processed based on the predefined audio processing to preview the recognized sign language symbols as previewed in figure 4.4.

Figure 4.4 Input audio

As for the last option the user can use real time input, the application displays a page where the user's camera will start, and it starts predicting which character is being previewed. The photo is processed based on the predefined photo processing steps; the application displays the results which is previewed in figure 4.5 which make this user interface easy to use by both signers and non-signer solving an issue that (**(Saquib, 2017))** has in his system.

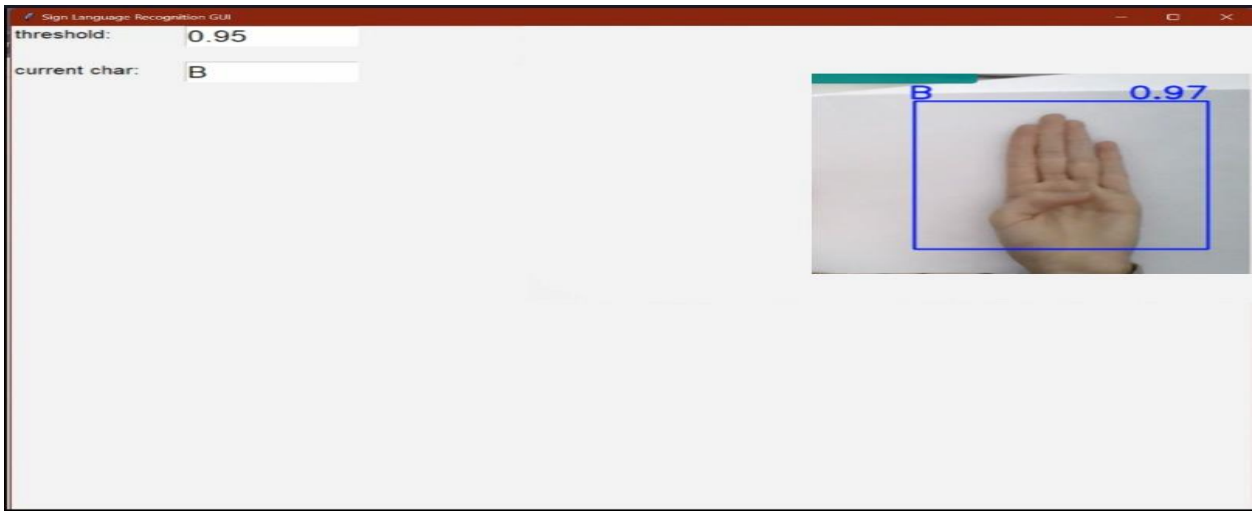

Figure 4.5 Real time Input

4.3 Project implementation

4.3.1 Data Collection

The primary source of data for this project was the compiled dataset of American Sign Language (ASL) called the ASL Alphabet from Kaggle The dataset is comprised of 87,000 images which are 200x200 pixels. There are 29 total classes, each with 3000 images, 26 for the letters A-Z and 3 for space, delete and nothing we excluded space, delete and nothing. This data is solely of the user Akash gesturing in ASL, with the images taken from his laptop's webcam. These photos were then cropped, rescaled, and labelled for use.

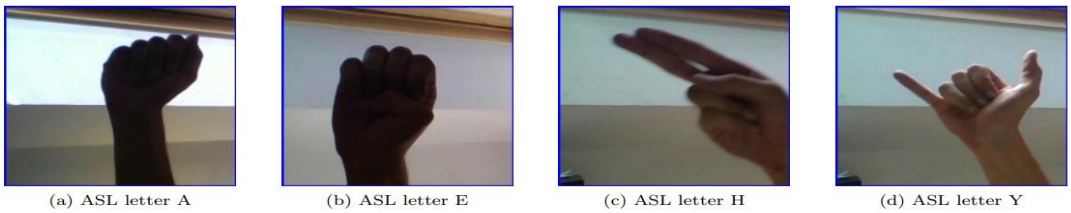

Figure 4.6 Examples of images from the Kaggle dataset used for training.

4.3.2 Data Pre-processing

Data preprocessing was done using the cv2 library an image processing library, Keras and Sklearn, For Image enhancement using multiple steps from changing to gray scale, noise reduction, and for highlighting edges.

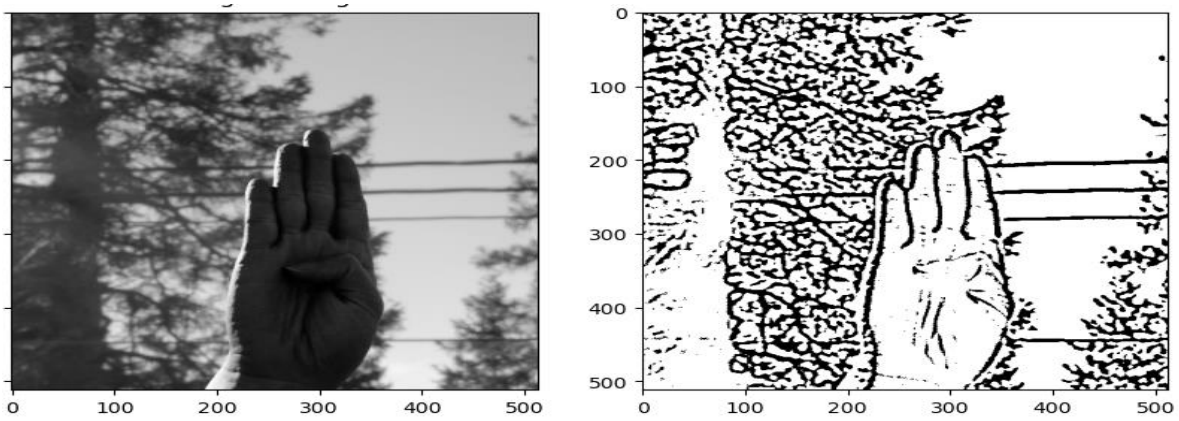

Figure 4.7 Examples of images after preprocessing

4.3.3 Deep learning model

4.3.3.(A) Overall Structure

The model employed in this classification task is a deep learning model capable of classifying 26 distinct classes of alphabets. It starts with an input layer, which is a 2D convolution layer with 64 filters of size (3,3), and it accepts an image of size (64,64) with a single channel. Following this, the model includes a series of hidden layers. These layers consist of ReLU activation functions, batch normalization, 2D convolution layers with a varying number of filters (128 and 256), max pooling layers with a pool size of (2,2), and dropout layers with a dropout rate of 0.25. After these layers, the model applies a reshape operation to the input to convert it into a 2D tensor. The reshaped output is then passed through a bidirectional LSTM layer with 100 units. The model concludes with two dense layers: the first with 512 units and a ReLU activation function, and the second with 26 units and a SoftMax activation function. The use of dropout and batch normalization is an attempt to prevent overfitting. The ReLU activation function is employed to introduce non-linearity into the model. The LSTM layer is designed to process the spatial features extracted by the CNN layers from individual images, which is beneficial if the data exhibits a sequential nature. The final SoftMax activation function ensures that the output can be interpreted as probabilities for each of the 26 classes. The class with the highest probability is selected as the model's prediction. This complexity of the model enabled us to recognize letter that are similar which was one the obstacles that **(Rivera-Acosta et al., 2021)** faced in his system.

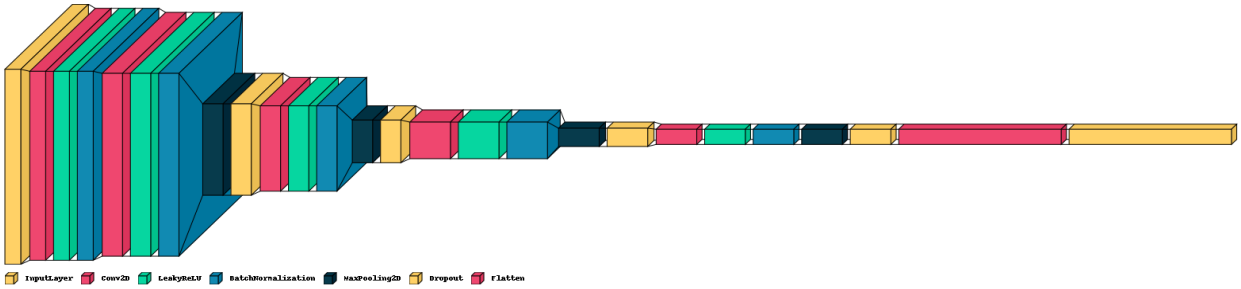

Figure 4.8 Model Architecture

4.3.3.(B) Model Performance

Our models were trained using the Adam optimizer and Cross Entropy Loss. The Adam optimizer is recognized for its rapid convergence compared to Stochastic Gradient Descent (SGD). We trained three distinct models: one using only a Convolutional Neural Network (CNN), the second combining a CNN and a Gated Recurrent Unit (GRU), and the last one integrating a CNN and a Long Short-Term Memory (LSTM) network. The models took a significantly longer time to train. We attribute this to the challenging nature of our classification task, which included background elements in the images and lower resolution, making training more difficult.

Although we had a substantial dataset to work with, comprising 3,000 samples for each of the 26 classes, we encountered challenges when processing the images into NumPy arrays. Our personal computers took considerable time to load all the images, necessitating the use of smaller datasets. We tested the impact of increasing the data available to our models and found a correlation between the number of samples per class and model accuracy. Models trained on less data tended to overfit quickly, likely due to the small number of training samples leading to poor generalization and learning of the sample space. However, increasing our dataset to the full number of samples per class resulted in improved model performance, with a peak validation accuracy of 83.3% in the 10th epoch.

Table 4.9 shows the result that was obtained throughout the training processes.

Model	Avg. Validation Accuracy
CNN	0.80
CNN +GRU	0.82
CNN+LSTM	0.83

Table 4.9 Results table

To determine which model was better we verified on a test set comprised of images from the dataset. The performance of the models on the test set is shown in Figure 4.9. We see that the model trained using CNN and LSTM was the best one.

For the CNN model the training accuracy that it achieved was 80% while the testing accuracy was 80% which is previewed in figure 4.10.

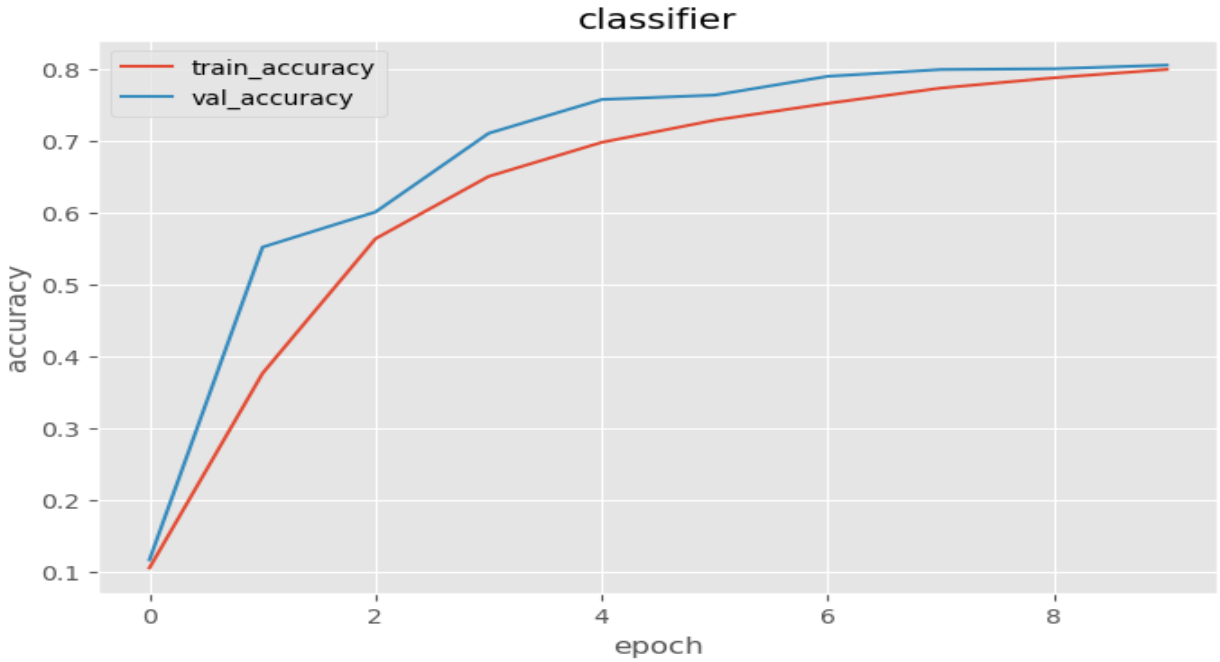

Figure 4.10 Training and Validation Accuracy (CNN)

The same steps were repeated but this time the model consisted of CNN +GRU and the training accuracy that it achieved was 72% while the testing accuracy was 82% which is previewed in figure 4.11.

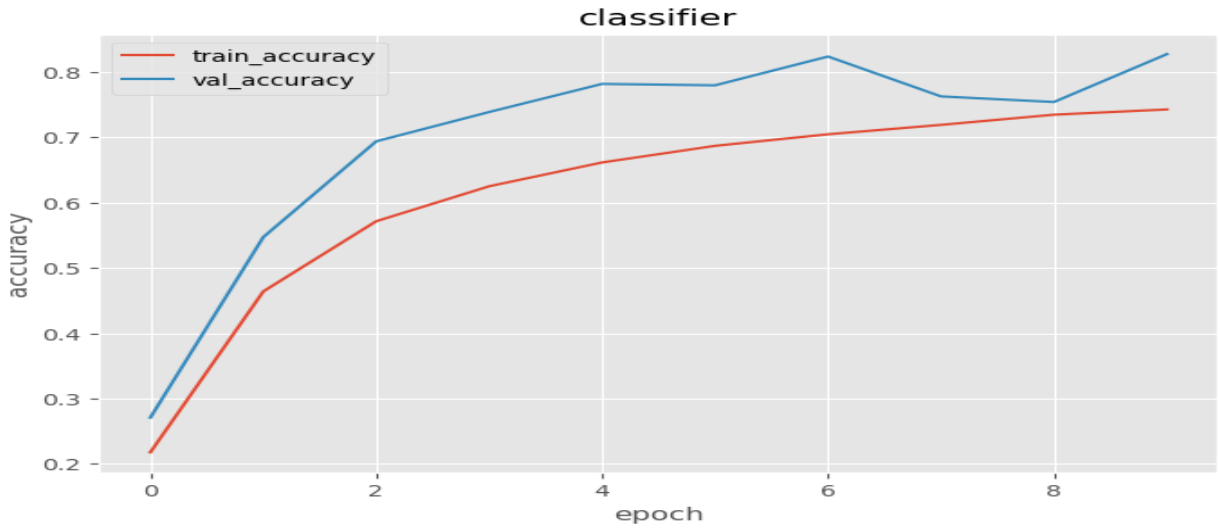

Figure 4.11 Training and Validation Accuracy (CNN) + (GRU)

The same steps were repeated but this time the model consisted of (CNN) + (LSTN) and the training accuracy that it achieved was 80.62% while the testing accuracy was 83.16% which is previewed in figure 4.12.

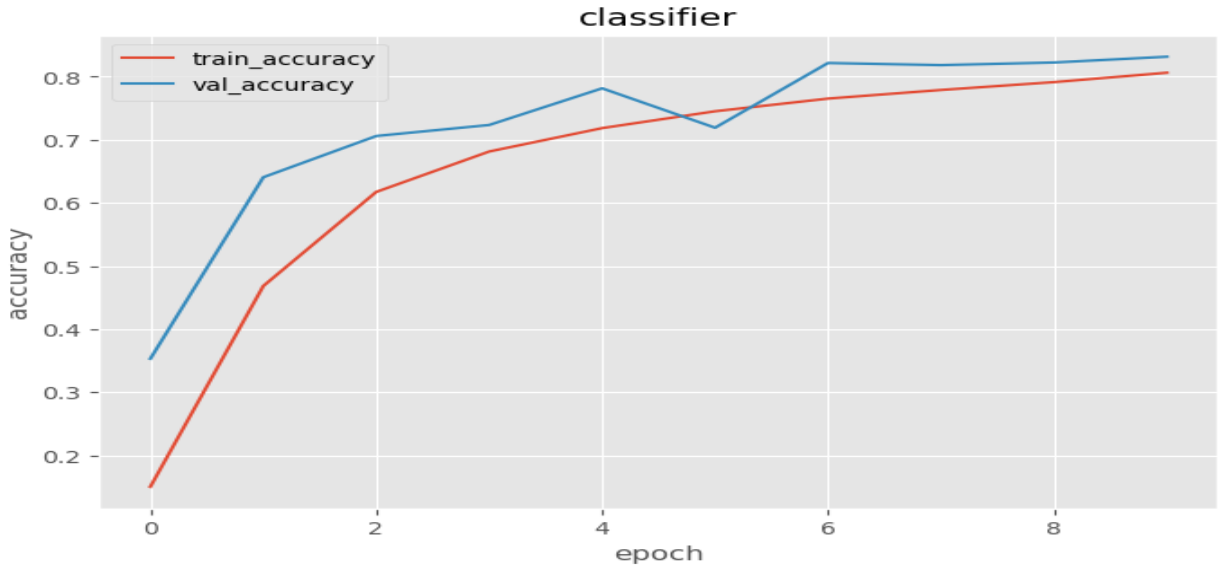

Figure 4.12 Training and Validation Accuracy (CNN) + (LSTM)

After selecting the final model, we proceeded with the testing phase. During this phase, we used several metrics to evaluate the model's performance. These metrics included:

1. **Loss**: This is a measure of how well our model is performing. It represents the summation of errors in our model. If the errors are high, the loss will be high, indicating that the model is not performing well. Conversely, the lower the loss, the better our model is performing. The loss is calculated using a loss or cost function. There are several different cost functions we used Cross-Entropy also finding the accuracy on a testing dataset which it achieved 84.45% and the loss was 0.48.

```
    def evaluate_model(self,classifier):
        train_images, val_images, test_images, train_labels, val_labels, test_labels = self.load_images()
        results = classifier.evaluate(test_images, test_labels)
        print("Loss of the model is - ", results[0])
        print("Accuracy of the model is - ", results[1]*100, "%")
```

```
1 data_gatherer.evaluate_model(classifier)
```

```
Loading images from folder A has started.
Loading images from folder B has started.
Loading images from folder C has started.
Loading images from folder D has started.
Loading images from folder E has started.
Loading images from folder F has started.
Loading images from folder G has started.
Loading images from folder H has started.
Loading images from folder I has started.
Loading images from folder J has started.
Loading images from folder K has started.
Loading images from folder L has started.
Loading images from folder M has started.
Loading images from folder N has started.
Loading images from folder O has started.
Loading images from folder P has started.
Loading images from folder Q has started.
Loading images from folder R has started.
Loading images from folder S has started.
Loading images from folder T has started.
Loading images from folder U has started.
Loading images from folder V has started.
Loading images from folder W has started.
Loading images from folder X has started.
Loading images from folder Y has started.
Loading images from folder Z has started.
317/317 [==============================] - 35s 111ms/step - loss: 0.4877 - accuracy: 0.8446
Loss of the model is -  0.4877445697784424
Accuracy of the model is -  84.45759415626526 %
```

Figure 4.13 Testing accuracy and loss for the model

2. **Confusion Matrix**: This is a table that summarizes the performance of a machine learning model on a set of test data. It provides more insight into not only the performance of a predictive model, but also which classes are being predicted correctly and which incorrectly. The confusion matrix includes:

    - True Positives (TP): These are the instances where the model correctly predicts the positive class.
    - False Positives (FP): These are the instances where the model incorrectly predicts the positive class.
    - True Negatives (TN): These are the instances where the model correctly predicts the negative class.
    - False Negatives (FN): These are the instances where the model incorrectly predicts the negative class.

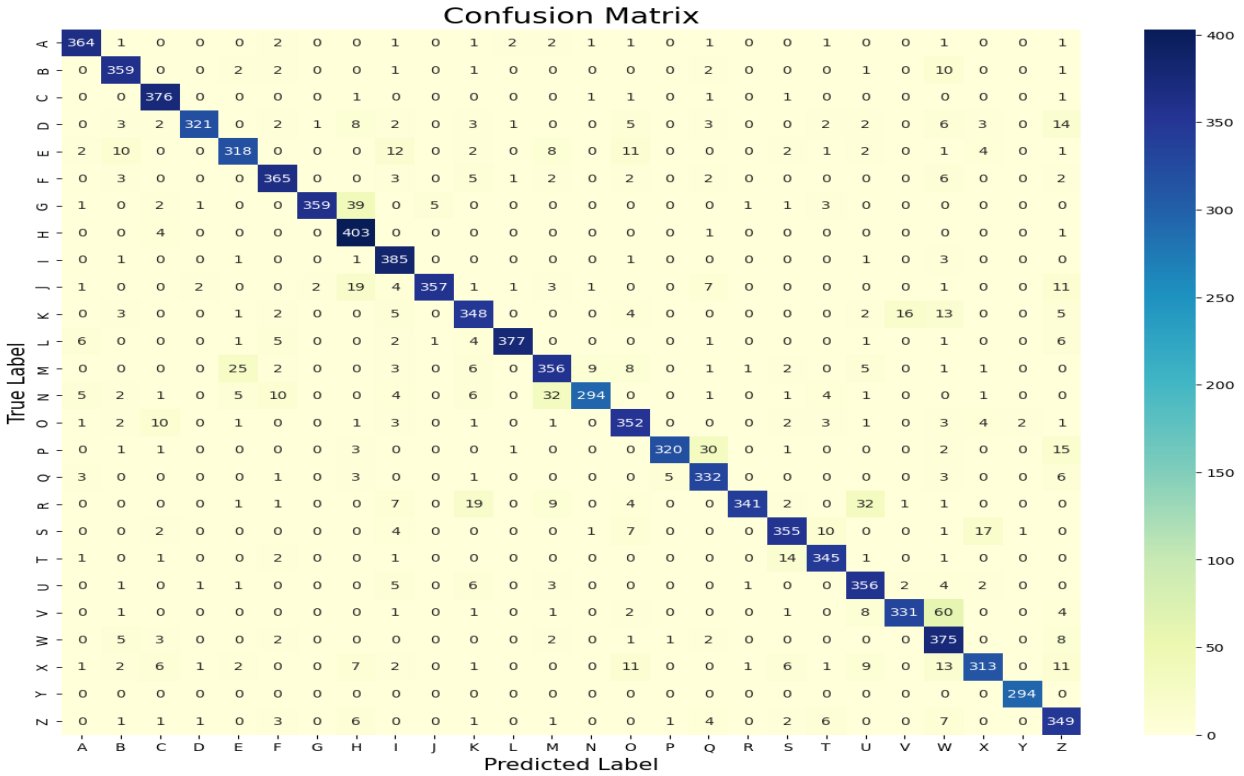

Figure 4.14 Confusion Matrix for the Model

3. **Classification Report**: This report includes several metrics that assess the quality of the model:

    - Precision: This is the percentage of correct positive predictions relative to total positive predictions. It is calculated as

$$precision = \frac{TP}{TP + FP}$$

    - Recall: This is the percentage of correct positive predictions relative to total actual positives. It is calculated as

$$Recall = \frac{TP}{TP + FN}$$

    - F1 Score: This is a weighted harmonic mean of precision and recall. The closer to 1, the better the model. It is calculated as

$$F1\ Score = 2 * \frac{Precision * recall}{Precision + recall}$$

	precision	recall	f1-score	support
0	0.71	0.91	0.80	370
1	0.68	0.96	0.80	370
2	0.88	0.89	0.89	399
3	0.97	0.82	0.89	375
4	0.76	0.87	0.81	414
5	0.94	0.83	0.88	426
6	0.90	0.92	0.91	382
7	0.90	0.94	0.92	435
8	0.69	0.92	0.79	400
9	0.78	0.87	0.83	417
10	0.94	0.85	0.89	397
11	0.97	0.85	0.91	378
12	0.79	0.57	0.66	399
13	0.57	0.89	0.69	367
14	0.97	0.77	0.86	357
15	0.96	0.80	0.87	395
16	0.91	0.91	0.91	385
17	0.85	0.81	0.83	365
18	0.82	0.82	0.82	393
19	0.97	0.84	0.90	409
20	0.93	0.71	0.80	408
21	0.95	0.77	0.85	378
22	0.65	0.90	0.76	361
23	0.90	0.73	0.81	379
24	0.94	0.86	0.90	396
25	0.98	0.78	0.87	385
accuracy			0.84	10140
macro avg	0.86	0.84	0.84	10140
weighted avg	0.86	0.84	0.84	10140

Figure 4.15 Classification Report for Final Model

4.3.3.(C) Testing

Testing the model on a different dataset that has a few challenging images to predict from then the one that it was trained on to see how the model would perform

Here is a sample from those images that predicted them correctly which is seen in Figure 4.16.

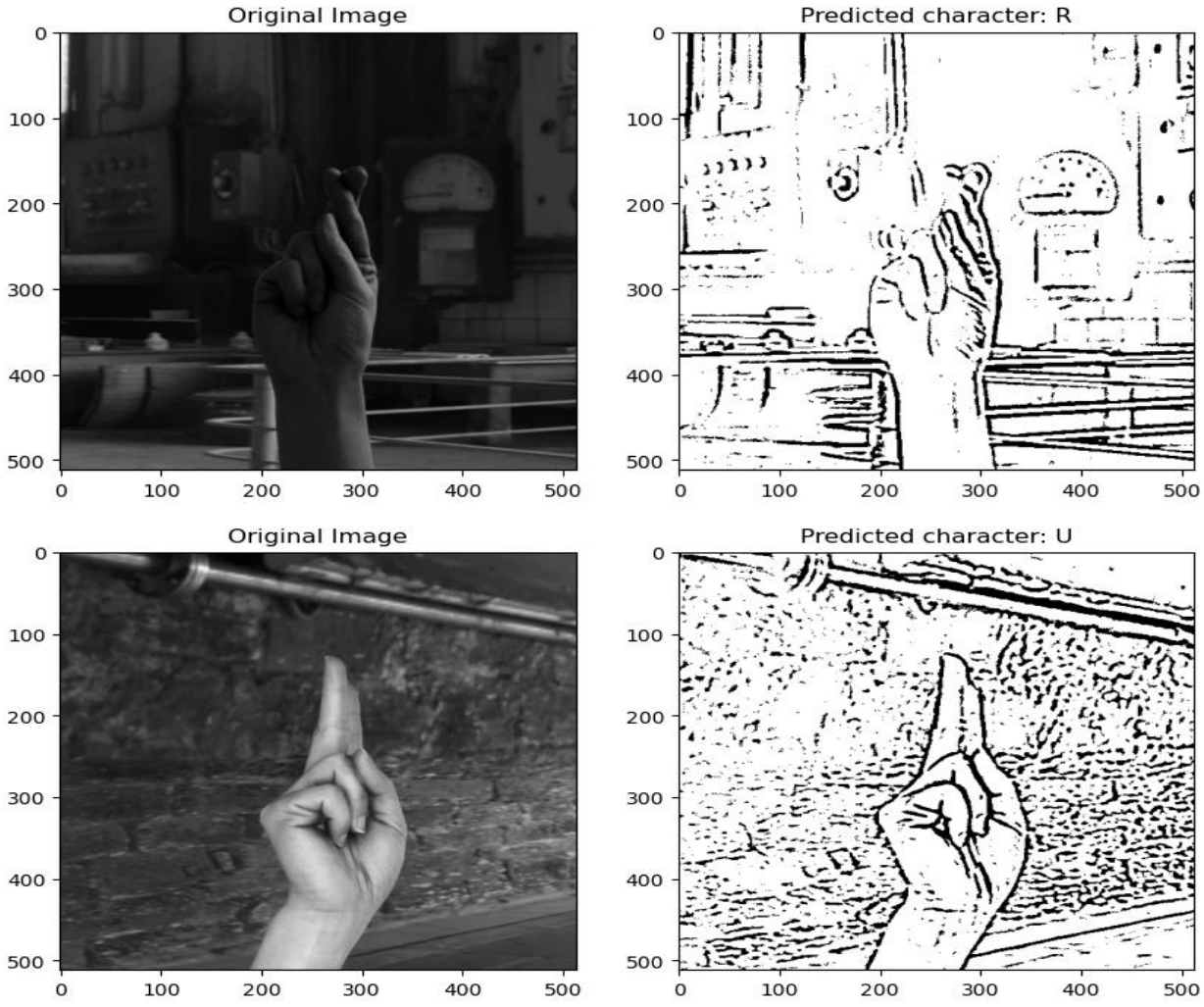

Figure 4.16 Testing Model on Images for R and U

Chapter Five

Conclusion and Future Work

5.1 Conclusion

Wrapping up, we've put together a solid strategy to boost sign language recognition by blending deep learning, computer vision, and language processing tools. By using CNN and LSTM models, along with NLP and PyAudio for speech, we've seen some great progress in turning hand gestures into text and the other way around. The smooth pairing of spoken words with 3D sign gestures shows us that we can help people who use sign language communicate more easily with those who use spoken language. As we look to the future, this project sets the stage for more working in the future, where we'll focus on making the system even more accurate, adding to our dataset, and looking at how this can be used in real-time to make a big difference in the world of sign language translation and communication access.

5.2 Future Work

For the future work, we've identified several areas for enhancement. First off, we plan to refine our preprocessing strategies to better adapt to various backgrounds. This involves expanding our dataset to cover a broader range of lighting conditions, hand gestures, and environmental factors. Next, we aim to implement feature selection on the NumPy array. This will allow our system to identify and utilize the most effective features for prediction, thereby improving accuracy. We also intend to optimize our model for faster response times in real-time classification. This will enhance the system's performance and make it more practical for real-world use. Lastly, we plan to incorporate more languages into our system. This will make our tool more inclusive and useful to a wider audience. In summary, with a focus on improving accuracy, expanding the dataset, and enhancing real-time performance. Our goal is to make significant strides in the field of sign language translation and communication accessibility.

References:

[1] Saquib Nazmus (2017). Sign language recognition using data gloves. University of Bangladesh

[2] Razieh Rastgoo, Kourosh Kiani, Sergio Escalera (2021). Sign Language Recognition: A Deep Survey

[3] Miguel Rivera-Acosta, Juan Manuel Ruiz-Varela, Susana Ortega-Cisneros, Jorge Rivera Ramón Parra-Michel and Pedro Mejia-Alvarez (2021). Spelling Correction Real-Time American Sign Language Alphabet Translation System Based on YOLO Network and LSTM

[4] Natarajan B, Rajalakshmi E, Elakkiya R, Ketan Kotecha, Ajith Abraham, Lubna Abdelkareim Gabralla, and, Subramaniyaswamy V(2022).Development of an End-to-End Deep Learning Framework for Sign Language Recognition, Translation, and Video Generation

[5] M. Kavya Mounika, B. Hema Latha, M. Sai Mounika, J. Sumanth (2022). SPEECH/TEXT TO SIGN LANGUAGE CONVERTOR USING NLP

[6] Debashis Das Chakladar,Kumar, Pradeep,Mandal, Shubham,Roy, Partha Pratim,Iwamura, Masakazu (2021) 3D Avatar Approach for Continuous Sign Movement Using Speech/Text

[7]- Koller, O., Ney, H., & Bowden, R. (2019). Deep Sign: Hybrid CNN-HMM for Continuous Sign Language Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence

[8] - Starner, T., Weaver, J., & Pentland, A. (2000). Real-time American Sign Language Recognition from Video Using Hidden Markov Models. IEEE Transactions on Pattern Analysis and Machine Intelligence.

[9]- Camgoz, N. C., Hadfield, S., & Bowden, R. (2018). Neural Sign Language Translation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. - Zhu, Y., Yang, J