ENHANCING ACCESSIBILITY: SIGN LANGUAGE TRANSLATION

Submitted by

ABHIRAMI SASIKUMAR: 65422825001

Guided by

Mrs. Arathi Chandran R.I

Assistant Professor

CHRIST NAGAR COLLEGE

MARANALLOOR, THIRUVANANTHAPURAM

A CMI Educational Institution | Affiliated to the University of Kerala

PROJECT REPORT

SUBMITTED ON PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF MASTER OF COMPUTER SCIENCE DEGREE OF THE UNIVERSITY OF KERALA

2024

CHRIST NAGAR COLLEGE

MARANALLOOR, THIRUVANANTHAPURAM

A CMI Educational Institution | Affiliated to the University of Kerala

CERTIFICATE

CERTIFIED THAT THIS PROJECT REPORT TITLED **"ENHANCING ACCESSIBILITY: SIGN LANGUAGE TRANSLATION"** IS A BONAFIDE RECORD OF THE MAJOR PROJECT WORK DONE BY **ABHIRAMI SASIKUMAR (Reg No: 65422825001)** UNDER THE SUPERVISION AND GUIDANCE, AT THE DEPARTMENT OF COMPUTER SCIENCE AND APPLICATIONS, CHRIST NAGAR COLLEGE TOWARDS THE PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF **MASTER OF COMPUTER SCIENCE** OF THE UNIVERSITY OF KERALA.

PRINCIPAL **HEAD OF THE DEPARTMENT** **INTERNAL GUIDE**

EXTERNAL EXAMINER

ACKNOWLEDGEMENTS

I am using this opportunity to express my sincere gratitude to everyone who supported me throughout the MSc Computer Science Major Project (Phase II). I first like to thank the **Almighty** for showering the blessings upon me.

I am extremely thankful to our Manager & Director, **Rev. Fr. Cyriac Madathil CMI**, for his blessing and inspiration.

Next, I would like to thank our Principal, **Dr. Jolly Jacob**, for her support and encouragement.

I express my sincere gratitude towards The Head of the Department of Computer Science and Applications, **Mrs. Sunitha S Nair** for providing an opportunity for undertaking this project.

I would also like to thank my project guide **Mrs. Arathi Chandran R.I**, Assistant Professor, for all her support throughout the project's completion.

I also take the privilege to extend my sincere thanks to the faculty members and all staff of Christ Nagar College for their valuable suggestions throughout the Major Project (Phase II) duration. Finally, I would like to express my profound thanks to all my friends for their support. I thank all the people mentioned above for their aspiring guidance, invaluably constructive criticism, and friendly advice during the project work. I am sincerely grateful to them for sharing their truthful and illuminating views on several issues related to the project.

WITH GRATITUDE

ABHIRAMI SASIKUMAR

ABSTRACT

A vital component of modern society, accessibility ensures inclusion and equal opportunities for those with disabilities. This paper investigates the creation of a sign translation system to improve accessibility for those who are dumb, blind, or deaf. The main objective is finding the best model for accurate and efficient sign language translation, using developments in deep learning (DL) and neural network technology.

Datasets in sign language are gathered, and different machine learning (ML) and deep learning architectures—such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs)—are implemented as part of the research technique. Performance criteria including accuracy, speed, and resource efficiency are used to train and assess these models.

The report also discusses the difficulties in translating sign language for people with disabilities. The study also discusses the difficulties in translating sign language for those who are visually or verbally impaired. To meet the needs of people who are blind or deaf, methods for integrating tactile input and audio information into the translation system are investigated.

The study's findings demonstrate the effectiveness of various machine learning and neural network models in translating sign language while taking accessibility needs and practical applications into account. The knowledge gathered from this study advances the development of technology meant to increase inclusivity and accessibility for people with impairments.

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Communication is the cornerstone of human interaction, serving as the conduit through which ideas are exchanged, relationships are formed, and societal bonds are strengthened. However, for individuals who are deaf, hard of hearing, or blind, traditional modes of communication often present formidable barriers, impeding their ability to fully engage with the world around them. Despite the strides made in technology, existing solutions for sign language translation remain fraught with limitations that hinder widespread accessibility and inclusivity.

Current sign language translation technologies are beset by a myriad of challenges, ranging from intrusive hardware requirements to constraints imposed by ambient lighting conditions and resolution specifications. Moreover, these technologies often fall short in their ability to capture the nuances of sign language, limiting their utility to translating individual signs rather than complete sentences or expressions. As a result, individuals with sensory impairments are confronted with obstacles that hinder their ability to communicate effectively in everyday scenarios, thereby exacerbating feelings of isolation and exclusion. In response to these challenges, the field of communication accessibility is witnessing a transformative shift fueled by advancements in technology. Emerging methodologies, such as bidirectional deep recurrent neural networks and convolutional neural networks, hold immense promise for enhancing the translation of American Sign Language (ASL) across various linguistic levels. By leveraging the power of artificial intelligence and machine learning, researchers are pioneering new approaches to bridge the communication divide and foster greater inclusivity for individuals with sensory impairments.

Furthermore, the convergence of Virtual Voice Assistants and Hand Gesture Recognition technologies represents a groundbreaking opportunity to empower individuals with speech/hearing impairments. By seamlessly integrating these technologies into everyday

communication devices, individuals can interact with digital gadgets and navigate the virtual landscape with unprecedented ease and efficiency. This synergistic fusion of cutting-edge technologies not only enhances accessibility but also opens a world of possibilities for individuals with sensory impairments, enabling them to engage more fully with their surroundings and participate actively in social interactions.

Against this backdrop, this project seeks to address the multifaceted challenges of communication accessibility faced by individuals who are deaf or hard of hearing. By developing a comprehensive system and user-friendly interface, this endeavor aims to empower deaf individuals to communicate effectively in diverse settings, thereby fostering greater connectivity and inclusivity. Through a holistic approach that encompasses technological innovation, user-centric design, and inclusive practices, this project endeavors to break down communication barriers and create a more equitable and inclusive society for all individuals, regardless of their sensory abilities.

Furthermore, the project's emphasis on research that combines Virtual Voice Assistants and Hand Gesture Recognition technologies represents a pioneering effort to revolutionize communication accessibility for individuals with speech/hearing impairments. By harnessing the synergistic capabilities of these technologies, the project aims to empower individuals to interact with digital gadgets and navigate the virtual landscape with unprecedented ease and efficiency. Through the development of a seamless system and intuitive interface, the project seeks to bridge the communication gap and empower individuals with sensory impairments to communicate effectively in diverse settings. Moreover, the project's focus on developing a comprehensive system and interface that enables deaf individuals to make use of voice-automated virtual assistants with the assistance of Sign Language represents a significant step forward in enhancing communication accessibility. By leveraging the power of sign language as a means of communication, the project aims to empower deaf individuals to interact with

digital gadgets and communicate with the outside world more effectively. Through a multidimensional approach that encompasses technological innovation, user-centric design, and inclusive practices, the project seeks to break down communication barriers and foster greater connectivity and inclusivity for individuals with sensory impairments.

Through collaborative research, innovative solutions, and a commitment to inclusivity, the project seeks to pave the way for a future where communication barriers are a thing of the past, and individuals with sensory impairments can fully participate in all aspects of life.

1.2 RESEARCH QUESTIONS

Training and optimizing Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) networks for real-time American Sign Language (ASL) gesture recognition and translation requires robust datasets and fine-tuning of hyperparameters to enhance accuracy and processing speed. These networks, known for their ability to handle sequential data, offer comparative advantages over other neural architectures by effectively capturing temporal dependencies essential for interpreting ASL gestures. The MediaPipe framework, pivotal for real-time hand gesture capture and processing, must be evaluated for effectiveness across various lighting and environmental conditions, with potential modifications to enhance its ASL recognition performance. Best practices for integrating text-to-speech technology include optimizing the clarity and accuracy of audio output to ensure comprehensibility for visually impaired users. The proposed system's impact on inclusivity and participation in social and professional settings can be significant, fostering better communication and integration for individuals with sensory impairments. Addressing these aspects comprehensively ensures the system's practical applicability and user-centric design, promoting greater inclusivity and accessibility.

1. How can RNN and LSTM networks be trained and optimized for real-time ASL gesture recognition and translation?
2. What are the comparative advantages of RNN and LSTM networks over other neural network architectures in handling sequential data for ASL translation?
3. How effective is the MediaPipe framework in capturing and processing hand gestures in various lighting and environmental conditions?
4. What are the best practices for integrating text-to-speech technology to ensure clear and accurate audio output of translated ASL content?

5. How does the system influence the inclusivity and participation of these individuals in various social and professional settings?
6. What modifications or enhancements are necessary to improve MediaPipe's performance in ASL gesture recognition?

1.3 OBJECTIVES

- Develop and train RNN and LSTM models specifically tailored to interpret the complex and dynamic nature of ASL gestures. Continuously refine the models through iterative training with diverse datasets to improve recognition accuracy and robustness across different users and settings.
- Implement algorithms that can effectively capture the temporal dependencies and nuances between successive gestures to form meaningful sentences. Validate the coherence and grammatical correctness of the generated text through rigorous testing and user feedback.
- Integrate MediaPipe's advanced hand tracking and pose estimation capabilities to streamline the initial gesture detection phase. Customize and fine-tune MediaPipe components to align with the specific requirements of ASL recognition, ensuring high accuracy and efficiency.
- Develop a feature to convert the translated text into audio output using text-to-speech technology to cater to individuals who are blind or visually impaired.
- Select and integrate high-quality text-to-speech engines that can produce natural and intelligible speech from the translated text. Ensure that the audio output supports multiple languages and dialects to cater to a broader audience and enhance usability.
- Develop a robust video processing pipeline that can handle high frame rates and varying resolutions to ensure smooth real-time translation. Implement adaptive techniques to manage different lighting conditions and background scenarios, maintaining consistent performance.
- Design an intuitive interface with clear visual and auditory feedback to guide users through translation. Include accessibility features like customizable displays, voice commands, and screen reader support for enhanced usability.

CHAPTER 2

LITERATURE SURVEY

2.1 LITERATURE REVIEW

The study and evaluation of sign language recognition systems entail a meticulous examination of research papers and journals to gauge the current landscape of methodologies, advancements, and challenges within the field. Through a systematic review of existing literature, researchers acquire invaluable insights into the diverse array of approaches, techniques, and algorithms utilized in sign language recognition. This comprehensive review spans a broad spectrum of topics, encompassing the utilization of machine learning and computer vision algorithms, such as convolutional neural networks (CNNs) and other architectural models, for the recognition of sign language gestures. Additionally, researchers explore the integration of multimodal inputs, such as depth sensors and wearable devices, aimed at augmenting the performance of these systems.

- The paper "Convolutional Neural Network Hand Gesture Recognition for American Sign Language (2021)"[1] presents a novel method that makes use of CNNs to recognize hand movements in American Sign Language (ASL). The study highlights how crucial image preprocessing methods are for improving the CNN model's validation accuracy, including skin models and Gaussian mixture models. Additionally, they go over the drawbacks of using ImageNet weights for transfer learning with ASL data and stress the necessity of a CNN architecture with a reasonable depth to get good results.

  To accurately recognize ASL gestures, the paper's suggested solution combines deep learning techniques with image preprocessing to extract features from gesture datasets using CNNs. The method described in the research focuses on real-time prediction accuracy and efficiency by using a multilayer perceptron model for hand gesture identification. All things considered, the paper offers insightful information about using CNNs for ASL gesture recognition and highlights the significance of tailored CNN

architectures and image preprocessing methods for best results. With possible uses in embedded systems and mobile devices, the findings aid in the development of portable, real-time sign language recognition systems.

- The paper "Hand Gesture Recognition and Voice, Text Conversion using CNN and ANN (2022)"[2] presents a detailed implementation and testing process for a system aimed at recognizing hand gestures and converting them into voice and text. Pre-processing the hand gesture dataset, modeling and training gesture pictures using CNN and ANN algorithms, webcam image extraction and feature identification, and speech and text translation of hand gestures are the four main implementation phases that are outlined in the document. The system's user interface is demonstrated with pictures of different hand movements that were taken with the webcam and used to display the results. The technology is important because it can help blind and deaf people communicate more successfully by translating hand gestures into text and speech output. Given the better accuracy of the CNN model, the conclusion emphasizes the system's preference, affordability, and usefulness in helping people with impairments communicate more effectively. Overall, the study offers a thorough review of a potential system that makes use of cutting-edge technologies to increase inclusivity and accessibility for those who struggle with communication.
- The paper “Gesture based Real-Time Sign Language Recognition System (2022)” [3] focuses on recognizing all 26 alphabets from A-Z in real-time, with voice output using the gTTS library. The combination of Convolutional Neural Network (CNN) and Histogram Equalization (HE) is utilized to achieve this goal. This approach signifies a step towards developing a comprehensive and inclusive sign language recognition system that can cater to a wider range of communication needs. Overall, the paper provides a glimpse into the advancements in sign language recognition technology and

the potential impact it can have on improving communication for individuals with hearing impairments. The integration of deep learning techniques, real-time processing, and voice output capabilities showcases the continuous evolution of assistive technologies in enhancing accessibility and inclusivity for all individuals.

- The paper "Hand Gesture Recognition to Facilitate Tasks for the Disabled (2022)" [4] presents a technique for recognizing gestures using Python libraries like Opencv and Numpy. The methodology involves capturing live gestures, differentiating between gestures based on parameters like area ratio and convexity defects, and using Opencv functions for recognition. Additionally, the system displays text messages and vocalizes gestures using audio files. Overall, the paper provides valuable insights into the development of a gesture recognition system for disabled individuals, emphasizing the use of technology to enhance communication and accessibility. The methodology, results, and potential applications discussed in the paper contribute to the advancement of assistive technologies for the disabled community.
- The paper "Hand Detection and Gesture Recognition in Complex Backgrounds (2023)"[5] presents a Convolutional Neural Network (CNN) based system for hand detection and gesture recognition, focusing on segmenting hands from complex backgrounds using OpenCV libraries. The proposed system is trained with binary images to eliminate the background and classify gestures based on extracted features. The paper introduces a novel approach to hand detection and gesture recognition by addressing the challenge of complex backgrounds, which is a common issue in real-world applications. By utilizing the VGG16 CNN architecture and OpenCV libraries, the system offers a practical solution for real-time computer vision tasks. The use of binary images for training and background elimination is a key methodological innovation that simplifies the classification process and enhances system performance.

The integration of OpenCV methods for background elimination and feature extraction showcases a comprehensive approach to hand detection and gesture recognition.

- The paper "Convolution Neural Network based Hand Gesture Recognition System (2023)" [6] presents a detailed study on utilizing Convolutional Neural Networks (CNNs) for accurately identifying hand gestures, particularly in the context of sign language recognition and human-computer interaction. The methodology section outlines the use of transfer learning with a pretrained VGG16 model for training a CNN-based classifier to recognize hand shapes. The process of dataset creation, model training, and gesture recognition workflow are explained in detail. The conclusion emphasizes the effectiveness of the proposed CNN-based system for recognizing dynamic hand gestures and the importance of data augmentation in enhancing classification accuracy. The paper suggests that the system offers higher accuracy and addresses issues of overfitting and lighting scenarios compared to existing systems.

2.2 COMPARISONS AND RESULTS

SL NO:	NAME OF THE PAPER	METHOD	ACCURACY	ADVANTAGES	DISADVANTAGES
1	Convolutional Neural Network Hand Gesture Recognition for American Sign Language	Image preprocessing techniques, CNN.	TA: 91.3% VA: 86.30%	Reduce the computational and storage complexity, A good set of training hyperparameters.	Limited to the specific dataset, computationally intensive, sensitive to noise, background clutter, or occlusions.
2	Hand Gesture Recognition and voice, text conversion using CNN and ANN	CNN, ANN	ANN: 86%, CNN: 98%	Reduction of hardware components as a simple and practical technique to establish human-computer interaction.	Lighting conditions, background noise, and hand positioning. Processing hand gestures in real-time for immediate voice and text conversion may pose challenges in terms of latency and responsiveness, especially in dynamic or fast-paced interaction scenarios.
3	Gesture based Real-Time Sign Language Recognition System	CNN, Histogram Equalization	87.8%	Highly efficient in predicting gestures, An original dataset is created for training and testing.	Limited gesture recognition, Environment factors, Processing speed, User adaptability
4	Hand Gesture Recognition to Facilitate Tasks for the Disabled	Image processing, OpenCV, Flex Sensor, Classification	95%	Useful for the entertainment of them as the program.	Limited Gesture Vocabulary, Hardware Limitations, Real-Time Processing Challenges.
5	Hand Detection and Gesture Recognition in Complex Backgrounds	Binary image training, VGG16 CNN, Classification	90%	Efficient feature Extraction, Uses a single camera.	Complex Background Limitations, Sensitivity to Lighting Conditions, Computational Complexity.
6	Convolution Neural Network based Hand Gesture Recognition System	2D CNN, VGG16	99.08%	Efficient approach for recognizing dynamic hand gestures, Prevent overfitting.	Limited Dataset, Complexity of Gestures, Overfitting.

Table 2.1.1: Comparison Of Different Papers Review.

2.3 CONCLUSION OF THE LITERATURE REVIEW

The survey of recent research in sign language and gesture recognition reveals diverse methodologies aimed at improving accessibility and communication for individuals with disabilities. Notable approaches include Convolutional Neural Networks (CNNs), Artificial Neural Networks (ANNs), and computer vision techniques. The first paper, "Convolutional Neural Network Hand Gesture Recognition for American Sign Language (2021)," highlights the role of image preprocessing and tailored CNN architectures in achieving high prediction accuracy for ASL gestures. The second paper, "Hand Gesture Recognition and Voice, Text Conversion using CNN and ANN (2022)," details a system that converts recognized gestures into voice and text, emphasizing affordability and practical application.

"Gesture-based Real-Time Sign Language Recognition System (2022)" focuses on real-time recognition of the ASL alphabet using CNN and Histogram Equalization, demonstrating significant technological advancements. The fourth study, "Hand Gesture Recognition to Facilitate Tasks for the Disabled (2022)," utilizes Opencv and Numpy to enhance communication for the disabled. Another notable work, "Hand Detection and Gesture Recognition in Complex Backgrounds (2023)," tackles the challenge of recognizing gestures in complex environments using the VGG16 CNN architecture. Lastly, the "Convolution Neural Network based Hand Gesture Recognition System (2023)" paper examines dynamic gesture recognition, addressing overfitting and lighting issues. These studies collectively advance sign language recognition technology, showcasing innovative approaches to enhance accessibility for individuals with disabilities.

CHAPTER 3

PROPOSED SYSTEM

3.1 INTRODUCTION TO DEEP LEARNING

Deep learning neural networks represent a significant advancement in machine learning, drawing inspiration from the structure and function of the human brain. These networks are characterized by multiple layers of interconnected neurons, where each neuron processes input data, applies weights and biases, and passes results to subsequent layers. Key to their operation are activation functions like ReLU, sigmoid, and tanh, which introduce non-linearity and enable the network to learn complex patterns. Various types of deep learning networks serve specific purposes: feedforward neural networks (FNNs) handle straightforward data flows, convolutional neural networks (CNNs) excel in image and video recognition by learning spatial hierarchies, recurrent neural networks (RNNs) process sequential data with connections that can retain information over time, and long short-term memory networks (LSTMs) are specialized RNNs capable of learning long-term dependencies. Applications span diverse fields including computer vision for image analysis and autonomous vehicles, natural language processing for translation and sentiment analysis, healthcare for medical image analysis and personalized medicine, finance for fraud detection and algorithmic trading, and gaming for AI development and strategy.

Recurrent Neural Networks (RNNs) and Long Short-Term Memory networks (LSTMs) are specialized architectures in deep learning for processing sequential data. RNNs maintain a memory of previous inputs to handle sequences effectively, making them suitable for tasks like speech recognition and language modeling. LSTMs improve upon RNNs by introducing gated mechanisms that better capture and remember long-term dependencies in data. They are essential for applications like natural language processing, time series analysis, and sequence generation, advancing AI's ability to understand and predict patterns.

3.2 PROPOSED SYSTEM

The proposed system aims to enhance accessibility for individuals who use American Sign Language (ASL) by developing a sign language translation system using Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) networks, implemented with the assistance of the MediaPipe framework. This system will facilitate seamless communication between deaf or hard-of-hearing individuals who use ASL and those who do not understand sign language.

The system's primary functionality is to translate ASL gestures captured through a camera feed into text in sentence format. This involves utilizing RNN and LSTM networks, which are well-suited for sequential data processing tasks like sign language recognition. These networks will analyze the sequence of hand gestures and movements captured by the camera and generate corresponding textual representations of the signed sentences.

Moreover, to further enhance accessibility, the system will incorporate a feature to convert the translated text into audio output for individuals who are blind or visually impaired. This conversion will enable users with visual impairments to comprehend the translated ASL content through auditory means, thereby ensuring inclusivity for a wider range of users.

The implementation of the proposed system will leverage the MediaPipe framework, which provides a comprehensive set of tools and libraries for building various multimedia processing applications. MediaPipe offers pre-trained machine learning models and components specifically designed for tasks like hand gesture recognition and pose estimation, which are essential for interpreting ASL gestures accurately.

In operation, the system will capture real-time video input of ASL gestures using a camera-equipped device, such as a smartphone or a webcam. The captured video feed will be processed

by the MediaPipe framework, which will extract and analyze the hand gestures using RNN and LSTM networks. The recognized gestures will then be translated into text in sentence format.

Finally, the translated text will be converted into audio output using text-to-speech technology, enabling individuals with visual impairments to perceive the translated ASL content audibly. This comprehensive approach ensures that the proposed system enhances accessibility by providing multiple means of communication for both deaf and blind individuals, thereby promoting inclusivity in communication and interaction.

3.2.1 Existing System

The existing system introduces an innovative approach to tackle the complexities of hand detection and gesture recognition, particularly in environments with challenging backgrounds, using Convolutional Neural Networks (CNNs). In today's era of touchless interfaces, where minimizing physical contact is crucial, gesture recognition plays a pivotal role in enhancing user experience and enabling cleaner interactions with computer systems. The primary goal of the system is to establish efficient communication channels between users and computers by accurately identifying hand configurations, orientations, movements, and spatial locations to interpret specific gestures.

Central to the system is the adoption of the VGG16 Convolutional Neural Network architecture, renowned for its effectiveness in extracting and classifying image features. VGG16 is structured with multiple layers of trainable filters that excel in capturing intricate patterns and details from input images. Leveraging this architecture enables the system to extract significant and relevant features from images of hand gestures, which is critical for precise gesture classification and recognition accuracy.

Furthermore, alongside the VGG16 architecture, the system integrates advanced background cancellation techniques facilitated by the OpenCV (Open Source Computer Vision) libraries. Background cancellation is essential to isolate hand gestures from complex and cluttered

environments where background elements may interfere with accurate recognition. OpenCV provides a comprehensive set of tools and algorithms for preprocessing image inputs, enabling the system to effectively remove unwanted background elements. This preprocessing step enhances the system's robustness by ensuring that the focus remains squarely on the hand gestures themselves, despite varying environmental conditions.

By combining the robust feature extraction capabilities of VGG16 with sophisticated background cancellation techniques from OpenCV, the system aims to deliver reliable and accurate gesture recognition capabilities. This approach not only enhances usability in touchless interaction scenarios but also contributes to improving overall user experience by facilitating intuitive and hygienic interactions with computing systems.

3.2.2 Proposed System Architecture

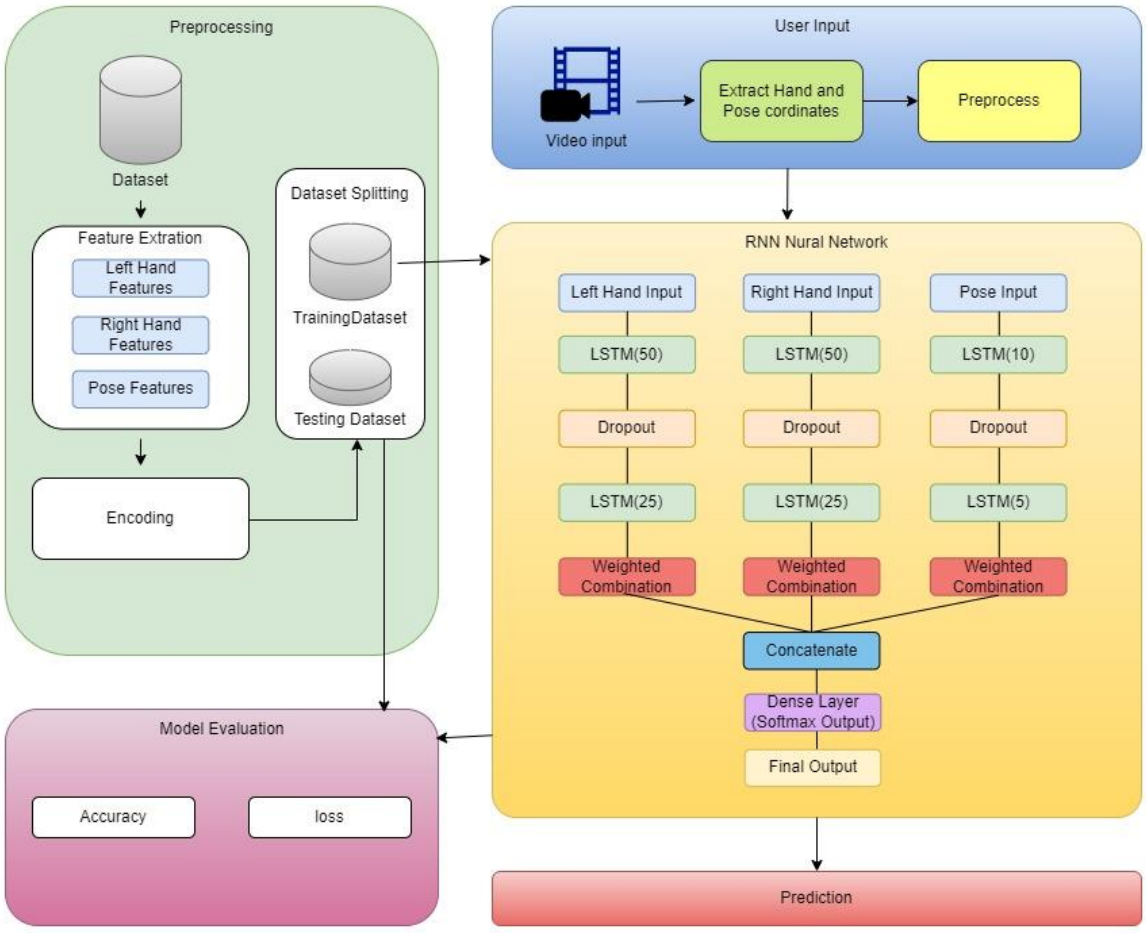

Figure 3.1: *Architecture of the proposed system*

The architecture presented outlines a comprehensive framework for hand and pose recognition using a Recurrent Neural Network (RNN) model. The process begins with preprocessing the video input. The video input is subjected to coordinate extraction for the left hand, right hand, and overall pose. This step involves identifying and recording the key points of the hands and pose from each frame of the video, which are then transformed into a usable dataset. The dataset is subsequently split into training and testing sets to ensure that the model can be evaluated effectively.

Following the preprocessing phase, feature extraction is carried out, where specific features are derived from the coordinates of the left hand, right hand, and pose. These extracted features

serve as the primary input for the RNN model. The combined dataset, now consisting of left hand features, right hand features, and pose features, is encoded if necessary to standardize the input data. This encoding process is crucial for the compatibility of the data with the neural network.

In the model building phase, the architecture defines three separate input pathways: one for each type of feature (left hand, right hand, and pose). Each pathway begins with an input layer followed by a series of Long Short-Term Memory (LSTM) layers designed to capture temporal dependencies within the data. For the left hand and right hand inputs, the model incorporates two LSTM layers with 50 and 25 units respectively, interspersed with dropout layers to prevent overfitting by randomly setting a fraction of input units to zero during training. The pose input follows a similar structure but uses LSTM layers with 10 and 5 units respectively due to the presumably lesser complexity of the pose data. Each pathway concludes with a weighted combination layer that merges the information captured by the LSTM layers.

The outputs from these three pathways are then concatenated into a single vector, effectively combining the learned features from the left hand, right hand, and pose inputs. This concatenated vector is then fed into a dense layer with a softmax activation function, which produces the final output of the model. The softmax layer converts the output into a probability distribution over the predefined classes, enabling the model to make predictions about the input data.

For the evaluation of the model, metrics such as accuracy and precision are employed to measure its performance on the testing dataset. Accuracy indicates the overall correctness of the model's predictions, while precision measures the proportion of true positive predictions among all positive predictions made by the model. This evaluation step is critical to ensure that the model generalizes well to unseen data and does not overfit the training set.

Finally, the trained model can be used for prediction on new video inputs. By processing new video data through the same preprocessing, feature extraction, and encoding steps, the model can predict the hand and pose configurations presented in the new data, providing valuable insights for applications such as gesture recognition, human-computer interaction, and motion analysis.

This architecture effectively integrates video preprocessing, feature extraction, and a sophisticated RNN model to achieve robust hand and pose recognition, demonstrating a well-structured approach to handling complex temporal data in a neural network framework.

Proposed System Flowchart

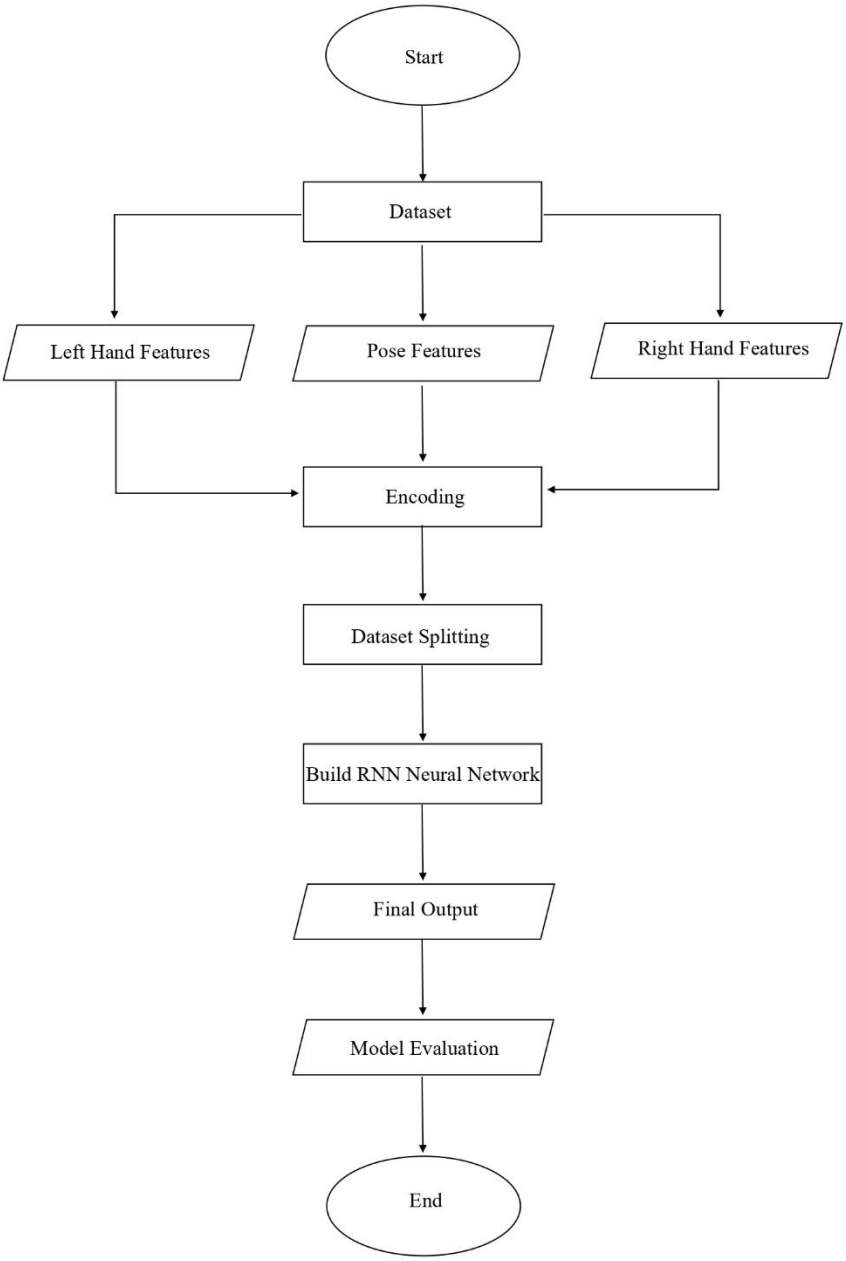

Figure 3.2: *Flowchart of Model Training*

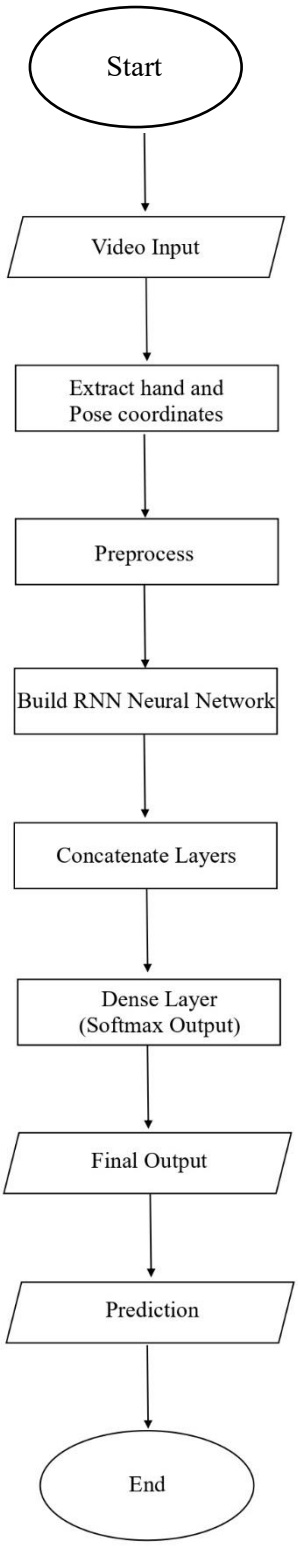

Figure 3.3: *Workflow of the system*

3.3 MODULE DESCRIPTION

1. **Real-time Transition:** The Real-time Transition module enables sign language translation in real-time by utilizing a camera feed. It captures sign language gestures and translates them into text or spoken language, facilitating immediate communication between sign language users and those who may not understand sign language.

   Input: Real-time video feed from a camera capturing hand gestures and body movements in real-time. This is often used in applications requiring live interaction, such as sign language interpretation or gesture-based control systems.

   Process:

   - **Real-Time Coordinate Extraction:** As the camera captures live video, the system continuously processes each frame to extract key points for the left hand, right hand, and body pose in real time. The extraction of key points must be efficient to ensure that the system keeps up with the live stream without delays.
   - **Real-Time Feature Extraction:** From the extracted key points, features are derived instantaneously for each frame. These features represent the current hand and pose configuration, allowing the system to recognize gestures as they occur.
   - **Sequence Encoding:** As with video input, the live stream provides sequential data. The extracted features are encoded and organized into a time series format to preserve the temporal structure of the gestures. This enables the system to understand ongoing movements in real time.
   - **Real-Time RNN Processing:** The sequential data is passed through the LSTM layers of the RNN model. The LSTM layers continuously update their internal states as new frames are processed, capturing the dynamic nature of the live gestures.

- **Real-Time Classification:** The model continuously generates predictions for the gestures or poses being performed. The final dense layer with softmax activation provides a live prediction of the current gesture or pose based on the most recent set of frames.

Output: The system outputs real-time predictions for the gesture or pose being performed, updating continuously as new frames are captured in text and audio format.

2. **Video Upload:** The Video Upload module allows users to upload videos containing sign language content. It processes the video, recognizing and translating sign language gestures, making it accessible to a broader audience.

   Input: A sequence of frames that captures hand gestures and body poses over time. The video typically shows a continuous set of movements that form a gesture or series of gestures.

   Process:

   - **Frame-by-Frame Coordinate Extraction:** The video is processed frame by frame, and key points for the left hand, right hand, and body pose are extracted from each frame.
   - **Temporal Feature Extraction:** For each frame, features are derived from the hand and pose coordinates. These features represent the movement and positioning of hands and body over time.
   - **Sequence Encoding:** The extracted features are arranged sequentially, forming a time series of gestures or poses. This sequential data is encoded to be compatible with the RNN model. Encoding helps to maintain the temporal order of the features, ensuring that the movement trajectory is preserved.

- **RNN Processing (LSTM Layers):** The sequence of features is passed through the LSTM layers, which capture the temporal dependencies and patterns in the hand movements and body postures. The LSTM layers model how hand and pose configurations change over time, allowing the network to understand complex gestures that unfold across multiple frames.
- **Classification:** The final layer of the model (a dense layer with softmax activation) converts the learned features into class predictions, providing the most likely gesture or pose corresponding to the video sequence.

Output: The system outputs the predicted class for the hand gesture or pose represented in the image in text and audio format.

3. **Image Upload:** The Image module is designed to predict the individual sign language letters or characters displayed in a single frame or image. This module aids in the recognition and interpretation of static sign language signs or gestures.

   Input: A single static image that captures a hand gesture or pose. This could be an image of a person performing a specific sign or gesture with their hands or body.

   Process:

   - Coordinate Extraction: The system processes the image to extract the key points for the left hand, right hand, and body pose. This involves identifying the coordinates of joints, hand positions, and body postures using techniques such as key point detection (e.g., OpenPose or Mediapipe).
   - Feature Extraction: From the extracted coordinates, features such as hand shape, finger orientation, and overall body posture are derived.
   - Encoding: The extracted features are encoded to ensure that the data is in a format compatible with the RNN architecture. Encoding could involve normalizing the coordinates or converting them into feature vectors.

- Model Prediction: Although RNNs are typically used for temporal data, in the case of a single image, the architecture can still be used to predict the gesture or pose based on static features. The extracted features are passed through the LSTM layers, which help to model any sequential aspects of the gesture within the image (e.g., overlapping hands or layered movements).
- Classification: The processed features are fed into a dense layer with a softmax activation function, producing a prediction for the gesture or pose present in the image.

Output: The system outputs the predicted class for the hand gesture or pose represented in the image in text and audio format.

3.4 ALGORITHM

Model Training Algorithm:

Step 1: Start: Initialize the process for model training using dataset features.

Step 2: Dataset: Load the dataset containing hand and pose features. Also, Ensure the dataset is in a suitable format for processing (e.g., CSV, JSON).

Step 3: Left-Hand Features: Extract features specific to the left hand from the dataset (coordinates of key points). Ensure the extracted features are consistent and well-organized.

Step 4: Pose Features: Extract overall body pose features from the dataset (coordinates of joints) and ensure the extracted features represent the body's posture accurately.

Step 5: Right-Hand Features: Extract features specific to the right hand from the dataset (coordinates of key points) and ensures the extracted features are consistent and well-organized.

Step 6: Encoding: Encode the extracted features into a numerical format suitable for neural network input.

Step 7: Dataset Splitting: Split the dataset into training and testing sets.

Step 8: Build RNN Neural Network: Define the architecture of the Recurrent Neural Network (RNN) using the encoded features.

Step 9: Final Output: Obtain the final processed output from the RNN and ensure the output is ready for evaluation.

Step 10: Model Evaluation: Evaluate the model's performance using the testing set. Calculate metrics such as accuracy and loss. Then, Analyze the results to determine the model's effectiveness.

Step 11: End: Conclude the process.

System Workflow Algorithm:

Step 1: Initialization: Start the process for video-based hand and pose recognition.

Step 2: Input Video: Capture video frames from the input source (Camera/Upload) and Ensure the video is in a suitable format for processing.

Step 3: Extract Hand and Pose Coordinates: Apply a pose estimation algorithm (MediaPipe) to detect hand landmarks and body pose. Extract the coordinates (x, y) for key points of hands and body.

Step 4: Preprocess: Normalize the coordinates to a consistent scale (e.g., between 0 and 1) and perform any necessary data augmentation (e.g., rotation, scaling) to enhance the dataset.

Step 5: Build RNN Neural Network: Define the architecture of the Recurrent Neural Network (RNN), including the number of layers and units. Choose the appropriate RNN type (LSTM).

Step 6: Concatenate Layers: Combine outputs from multiple layers, concatenating outputs from different time steps. Also, Ensure the concatenated data is in the correct format for the next layer.

Step 7: Dense Layer: Add a fully connected (dense) layer to the RNN. And use the softmax activation function to convert the network's output into a probability distribution over possible class.

Step 8: Final Output: Obtain the final processed output from the dense layer and ensure the output is ready for prediction.

Step 9: Prediction: Use the final output to make predictions (classifying gestures or poses).

Step 10: End: Conclude the process and display the predictions.

3.5 USE CASE DIAGRAM

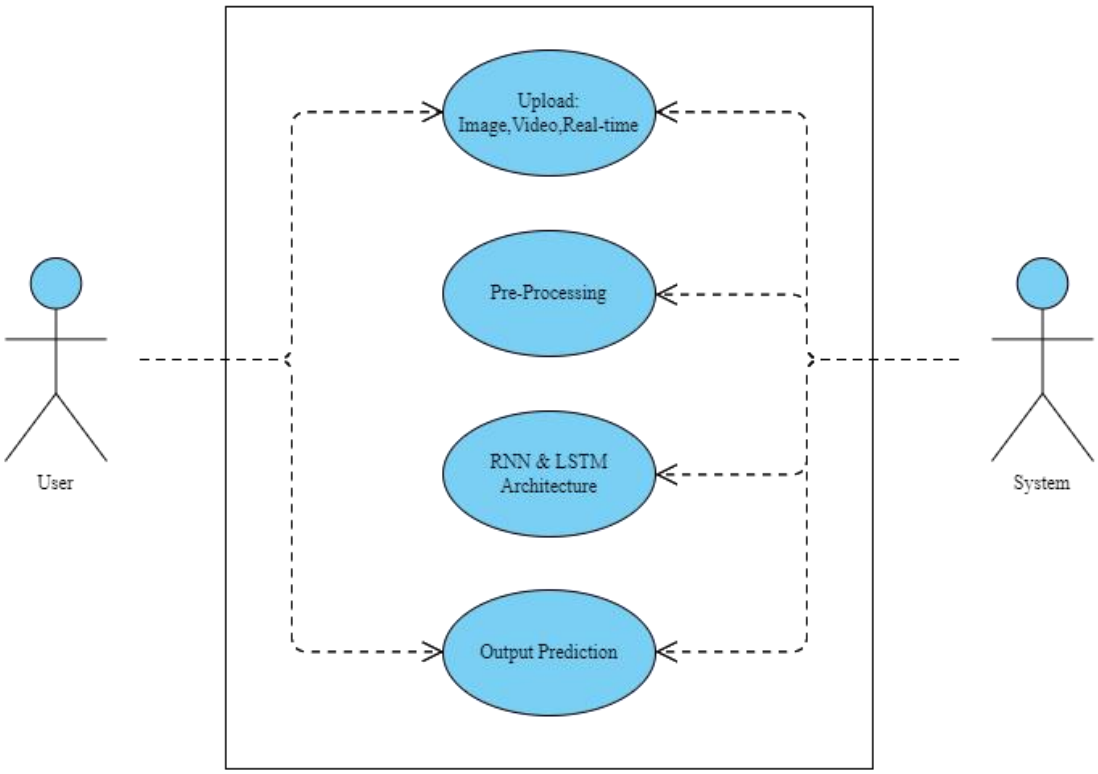

Figure 3.5: *Use case diagram of proposed sign language translation system*

CHAPTER 4

RESULTS AND DISCUSSION

4.1 RESULT ANALYSIS

The proposed system for hand and pose recognition leverages a Recurrent Neural Network (RNN) model with Long Short-Term Memory (LSTM) layers to process and analyze video inputs effectively. The system begins with preprocessing the video input, which involves extracting coordinates for the left hand, right hand, and overall pose from each frame. These coordinates are then transformed into a dataset that is split into training and testing sets for model evaluation.

In the feature extraction phase, the coordinates of the left hand, right hand, and pose are used to derive specific features, which are then encoded if necessary to standardize the input data. This ensures compatibility with the RNN model. The RNN model architecture includes three distinct input pathways—one for each type of feature (left hand, right hand, and pose). Each pathway comprises LSTM layers designed to capture temporal dependencies within the data. For the left hand and right hand features, the model uses two LSTM layers with 50 and 25 units, respectively, while the pose features are processed through LSTM layers with 10 and 5 units, reflecting the relatively simpler nature of pose data. Dropout layers are integrated to prevent overfitting by randomly deactivating a portion of input units during training.

The outputs from these three pathways are concatenated into a single vector and fed into a dense layer with a softmax activation function. The softmax function, expressed as

$$\sigma(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

where (x_i) denotes the output for class (i), converts the dense layer's output into a probability distribution over predefined classes. This enables the model to make predictions about the input data based on the computed probabilities.

The model's performance is assessed using accuracy and precision metrics. Accuracy is calculated with the formula: $\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \times 100\%$

With an accuracy of 97.25%, the system demonstrates high performance in correctly predicting hand and pose configurations. Precision measures the proportion of true positive predictions among all positive predictions, providing additional insight into the model's effectiveness.

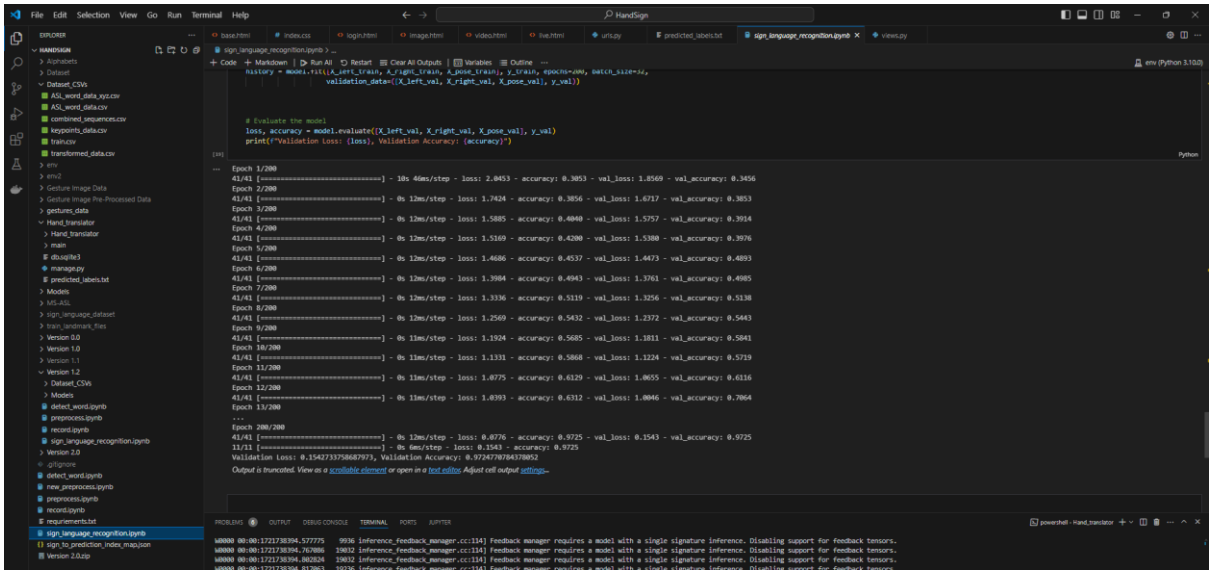

Figure 4.1: *Accuracy and score analysis*

The model was trained over 200 epochs to evaluate its performance. Throughout the training process, key metrics such as loss and accuracy were monitored. Initially, during the first epoch, the model showed a training loss of 2.0453 and an accuracy of 30.51%. The validation loss was recorded at 1.6771 with a validation accuracy of 38.53%, reflecting the early stages of the model's learning where predictions were still largely incorrect.

As training progressed, both the loss and accuracy steadily improved, signifying the model's increasing ability to learn from the data. By the 200th epoch, the training loss had reduced significantly to 0.0776, while the training accuracy reached 97.25%. These values indicate that the model had learned to make highly accurate predictions on the training data. The validation metrics also improved in parallel, with a final validation loss of 0.1543 and validation accuracy of 97.25%. This close alignment between training and validation performance suggests that the model generalizes well to unseen data, effectively avoiding overfitting.

The loss function used during training likely followed the categorical cross-entropy formula, which measures the error between the true labels and predicted probabilities. Accuracy was calculated as the ratio of correct predictions to the total number of predictions. The final results demonstrate that the model performs efficiently in recognizing sign language, achieving a balance between high accuracy and low error on both training and validation datasets.

Method	Validation loss	Accuracy
RNN Model	0.1542	0.9725

Figure 4.2: *Model accuracy and loss*

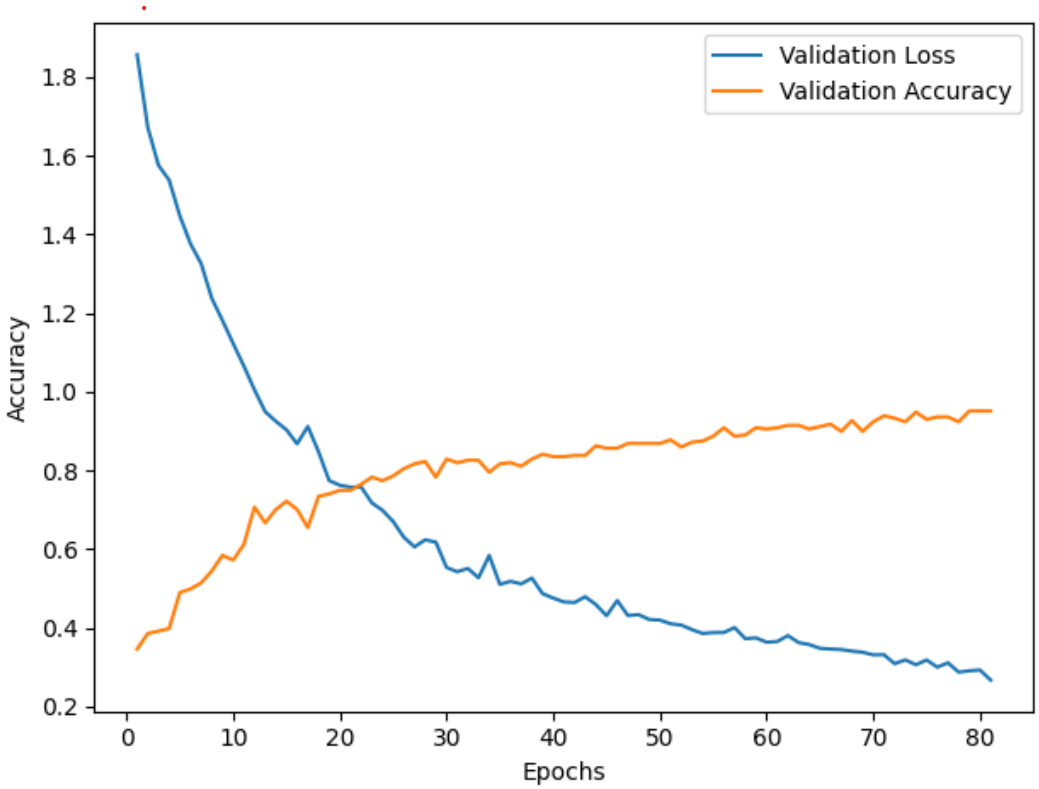

Figure 4.3: *Validation loss and validation accuracy graph*

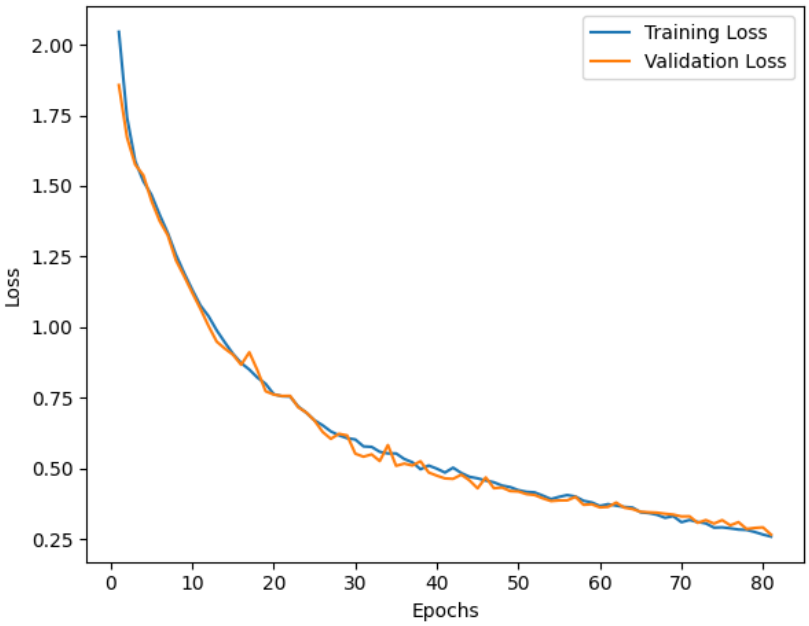

Figure 4.4: *Training loss and validation loss graph*

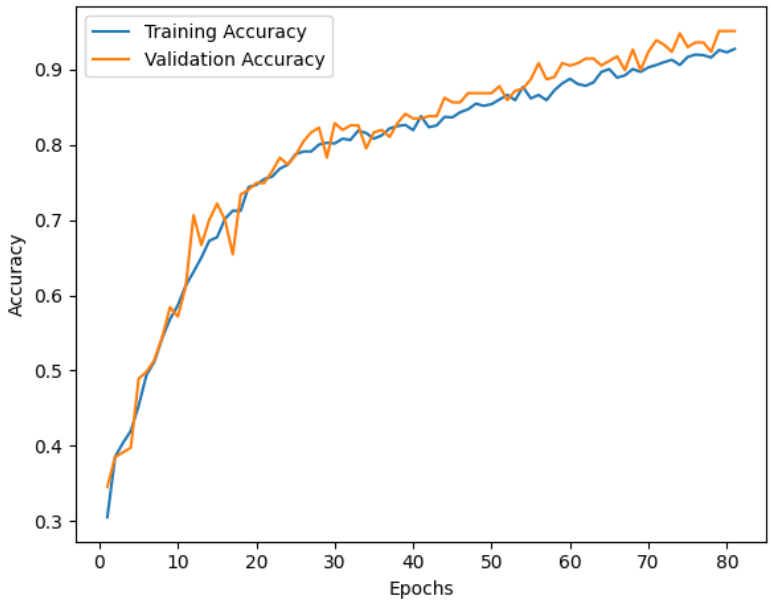

Figure 4.5: *Training accuracy and validation accuracy graph*

4.2 ADVANTAGES AND LIMITATIONS

Advantages:

The proposed system offers several significant advantages that enhance its effectiveness and usability. One of the most notable benefits is its high accuracy rate of 97.25%, which ensures reliable and consistent translation of ASL gestures into both text and audio formats. This high accuracy is critical for effective communication, as it minimizes the likelihood of misinterpretation and facilitates clearer interactions between ASL users and non-users.

In terms of inclusivity, the system excels by providing both text and audio outputs. This dual-output capability makes the system accessible to a diverse range of users, including those who are deaf, hard-of-hearing, or visually impaired. By supporting multiple forms of communication, the system fosters greater inclusivity and ensures that individuals with different needs can effectively engage with the content.

The system's real-time processing capability is another significant advantage. By capturing and interpreting ASL gestures in real-time, the system enables immediate and seamless communication. This feature is particularly beneficial in dynamic environments where timely interaction is crucial, such as in educational settings, customer service scenarios, or emergency situations.

The integration of advanced technologies, such as Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks, enhances the system's ability to handle complex sequential data. These technologies are well-suited for recognizing and interpreting the temporal aspects of ASL gestures. Additionally, the MediaPipe framework contributes to the system's robustness by providing sophisticated tools for hand gesture recognition and pose estimation, ensuring accurate and reliable feature extraction from the video input.

The system's versatility is another key advantage. It can be applied in various domains, including education, healthcare, customer service, and public services. This broad applicability enhances communication and interaction in diverse contexts, making the system a valuable tool for numerous applications.

Moreover, the system is designed to be user-friendly and accessible, with implementation possible on commonly available devices such as smartphones and webcams. This ease of use ensures that the system can be readily adopted by users without requiring specialized equipment or extensive technical knowledge.

Limitations:

Despite its strengths, the system has several limitations that may impact its performance and usability. One major limitation is its dependence on the quality of the camera feed. The accuracy and reliability of the system can be significantly affected by poor lighting conditions, low resolution, and physical obstructions that interfere with the camera's view. Ensuring optimal camera conditions is crucial for maintaining the system's effectiveness.

Another limitation is that the system is currently tailored specifically for ASL. This specialization means that the system may not support other sign languages without additional training and adaptation. Expanding the system to accommodate different sign languages would require further development and customization.

The complexity of certain ASL gestures poses another challenge. Some gestures may involve subtle nuances or intricate movements that can be difficult for the system to accurately recognize and interpret. Improving the system's ability to handle such complex gestures may require enhanced algorithms or additional training data.

Real-time processing, while beneficial, also presents constraints. The system requires significant computational power to process video input and perform gesture recognition in

real-time. This requirement may limit performance on lower-end devices or in environments with limited computing resources.

The quality and naturalness of the audio output generated by the text-to-speech technology can vary. Variations in the effectiveness of text-to-speech systems may impact the clarity and comprehensibility of the audio output, potentially affecting user experience.

Additionally, although the system is designed to be user-friendly, there may still be a learning curve for new users. Familiarizing oneself with the system's features and functionalities may take time, particularly for users who are not accustomed to similar technologies.

Lastly, despite the high accuracy rate, there is always a potential for misinterpretation of gestures. Factors such as noisy environments or rapid signing can introduce errors, affecting the system's reliability in diverse scenarios. Addressing these challenges is essential for improving the system's robustness and overall effectiveness.

4.3FUTURE SCOPE

The future scope of the proposed ASL translation system is vast and multifaceted, aiming to significantly enhance its functionality, usability, and inclusivity. A critical area for development is the expansion of vocabulary and contextual understanding. This involves continuously updating the system to recognize a broader range of ASL gestures, including regional variations and idiomatic expressions, and developing algorithms to better understand the context of conversations, thereby improving translation accuracy in various scenarios. Additionally, multimodal integration will play a vital role by incorporating facial recognition and body language analysis to capture the full spectrum of ASL communication, which includes facial expressions and body postures. Integrating environmental and contextual cues will further enhance understanding and translation accuracy.

User customization and personalization are also crucial for improving the system's accuracy and user experience. Allowing users to create personalized profiles that store their specific signing styles and preferences will significantly enhance recognition accuracy. Implementing adaptive learning algorithms that continuously learn and adapt to individual users' signing patterns over time will further improve the system's effectiveness. Moreover, real-time collaboration and communication features, such as enabling live, bi-directional translation between ASL and spoken language during conversations, video calls, and meetings, will facilitate seamless interactions. Multi-user support will also be essential, allowing multiple users to engage simultaneously in group conversations and collaborative work environments.

Integration with assistive technologies will expand the system's accessibility. Incorporating haptic feedback devices will provide tactile responses for deaf-blind users, enhancing their

ability to receive information. Utilizing augmented reality (AR) glasses to display real-time translated text overlaid on the user's field of view will provide a seamless visual experience, improving communication for users with different needs. Cross-lingual and cross-cultural expansion is another significant area, extending the system to support translation between ASL and other sign languages globally, such as Indian Sign Language (ISL), British Sign Language (BSL), and Japanese Sign Language (JSL). Developing culturally sensitive translation models that consider cultural nuances and variations in sign language usage will ensure inclusivity and accuracy.

Educational applications offer another promising avenue for development. Creating interactive tools and applications for learning ASL, using the translation system to provide instant feedback and guidance, will enhance ASL education. Enabling virtual classrooms where deaf and hearing students can interact seamlessly, with real-time translation facilitating communication, will further promote inclusive education. By pursuing these future developments, the proposed ASL translation system can significantly enhance its impact, making communication more accessible and inclusive for individuals with hearing and visual impairments across various aspects of life.

CHAPTER 5

CONCLUSION

The proposed ASL translation system represents a significant advancement in accessibility technology, aiming to bridge the communication gap between individuals who use American Sign Language (ASL) and those who do not. By leveraging Recurrent Neural Networks (RNN) and Long Short-Term Memory (LSTM) networks, in conjunction with the MediaPipe framework, the system accurately translates ASL gestures captured via a camera into both text and audio formats. This dual functionality ensures inclusivity for a broad range of users, including those who are deaf, hard-of-hearing, or visually impaired.

The system's high accuracy rate of 97.25% underscores its robust performance and reliability in practical, real-world scenarios. Its ability to process ASL gestures in real-time further enhances its effectiveness, making it a valuable tool for various applications, including education, healthcare, customer service, and public services.

Despite its numerous advantages, the system has certain limitations, such as dependency on camera feed quality, potential complexity in gesture recognition, and the need for significant computational power for real-time processing. Addressing these challenges in future iterations could further enhance the system's usability and performance.

In conclusion, the ASL translation system offers a comprehensive solution to facilitate seamless communication between ASL users and non-users, promoting greater inclusivity and accessibility. By continuing to refine and expand its capabilities, this system has the potential to make a substantial impact on improving communication and interaction for diverse user groups.

REFERENCES AND APPENDIX

APPENDIX A- Screenshot of The Application

WELCOME PAGE

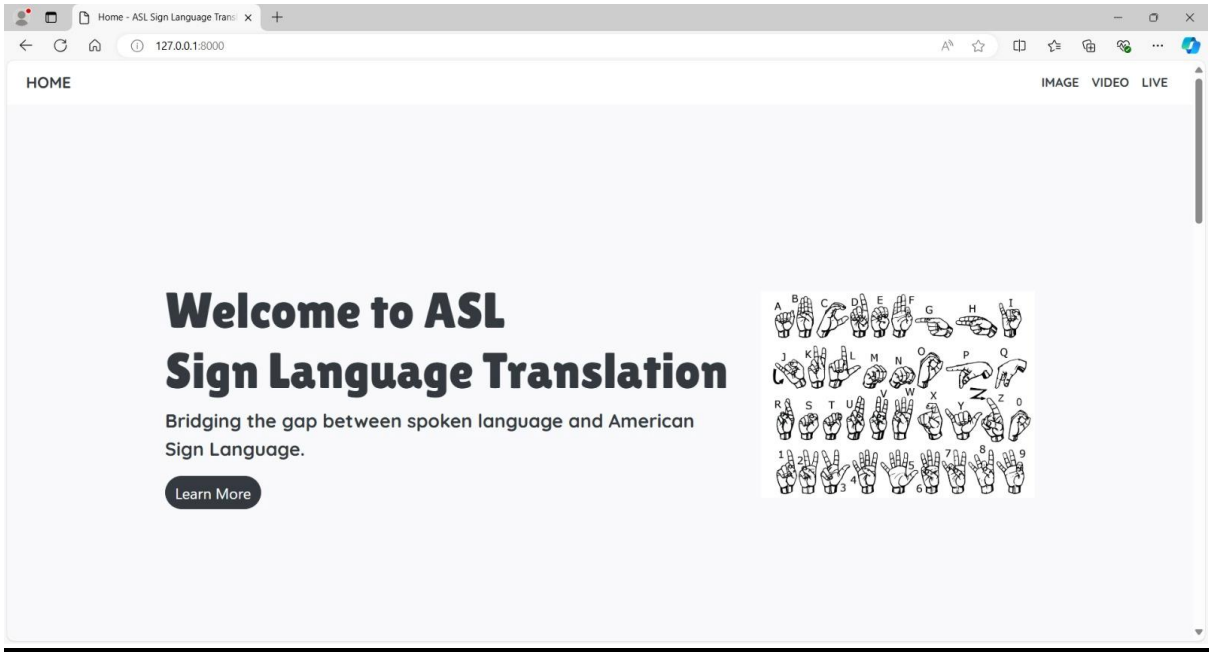

Figure A.1: *Welcome Page*

HOME PAGE

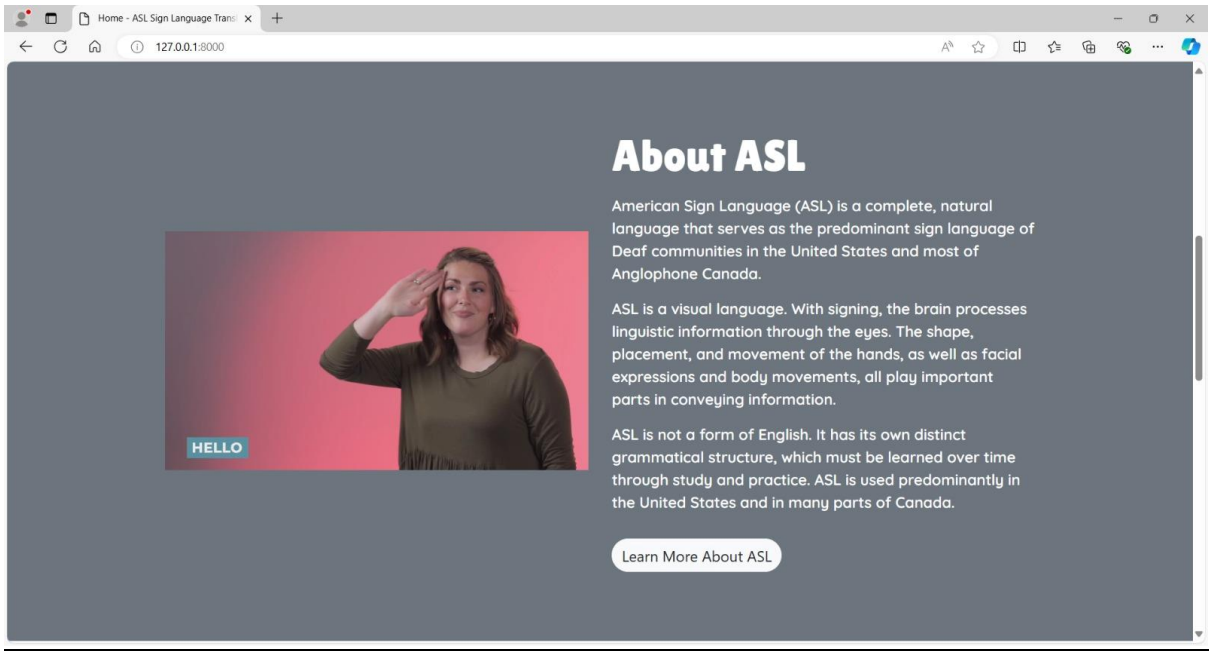

Figure A.2.1: *Home Page*

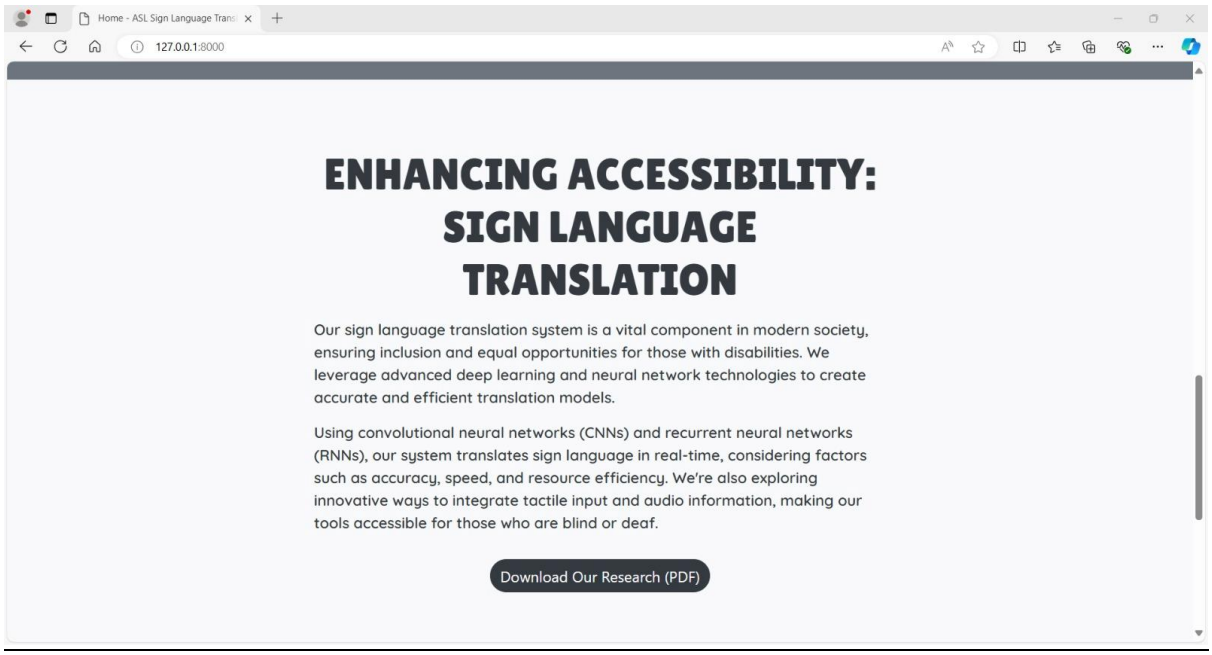

Figure A.2.2: *Home Page*

VIDEO UPLOAD

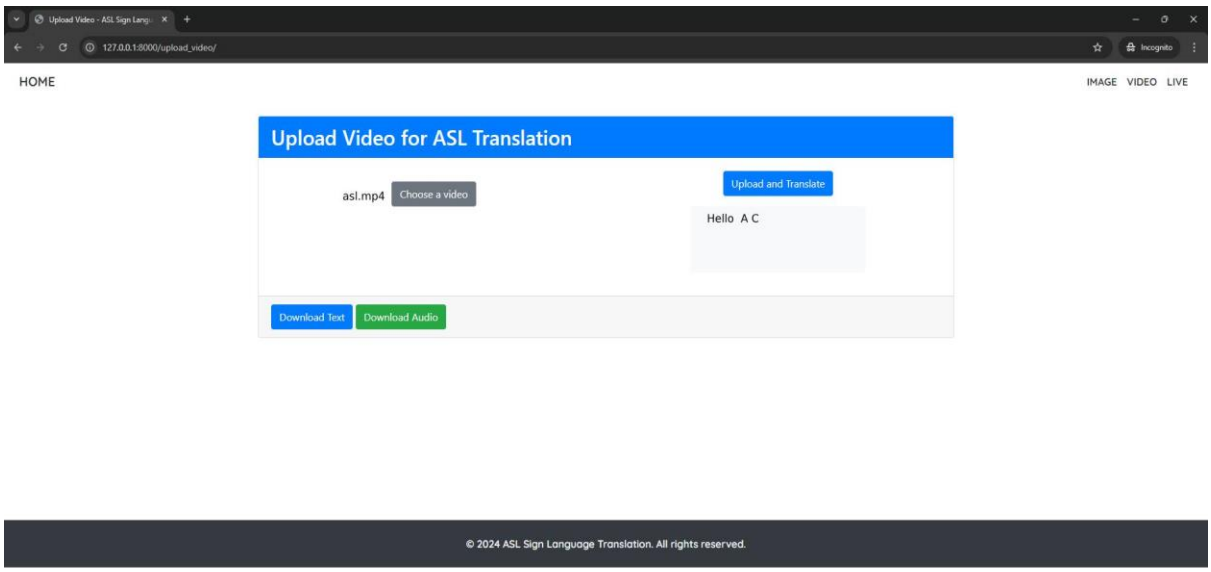

Figure A.3: *Video Upload*

IMAGE UPLOAD

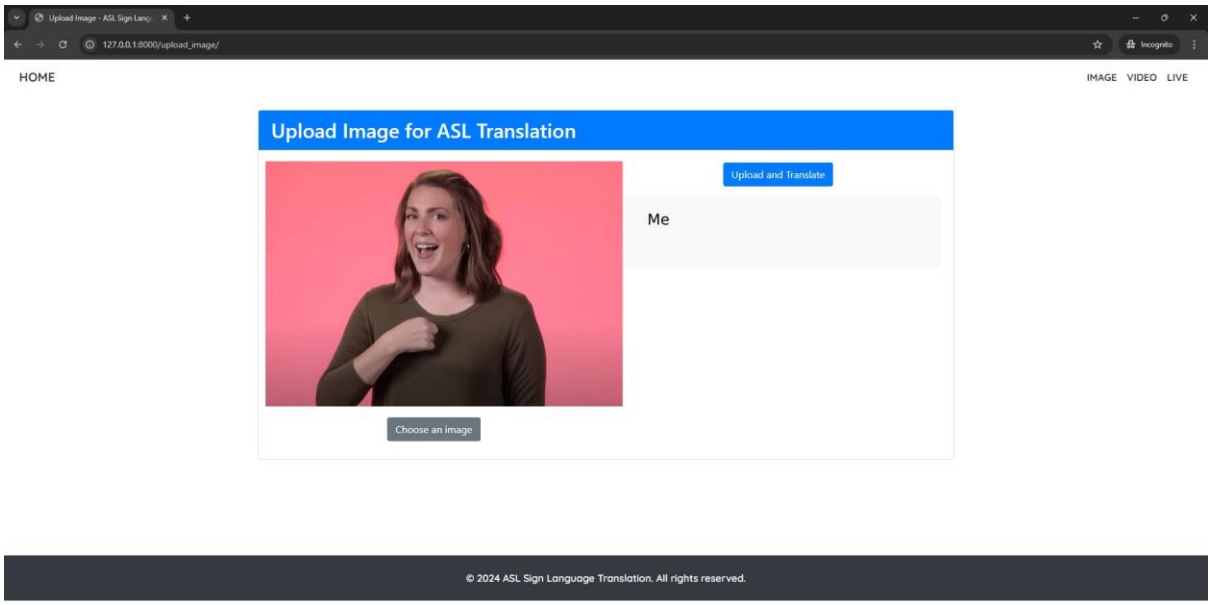

Figure A.4: *Image Upload*

LIVE

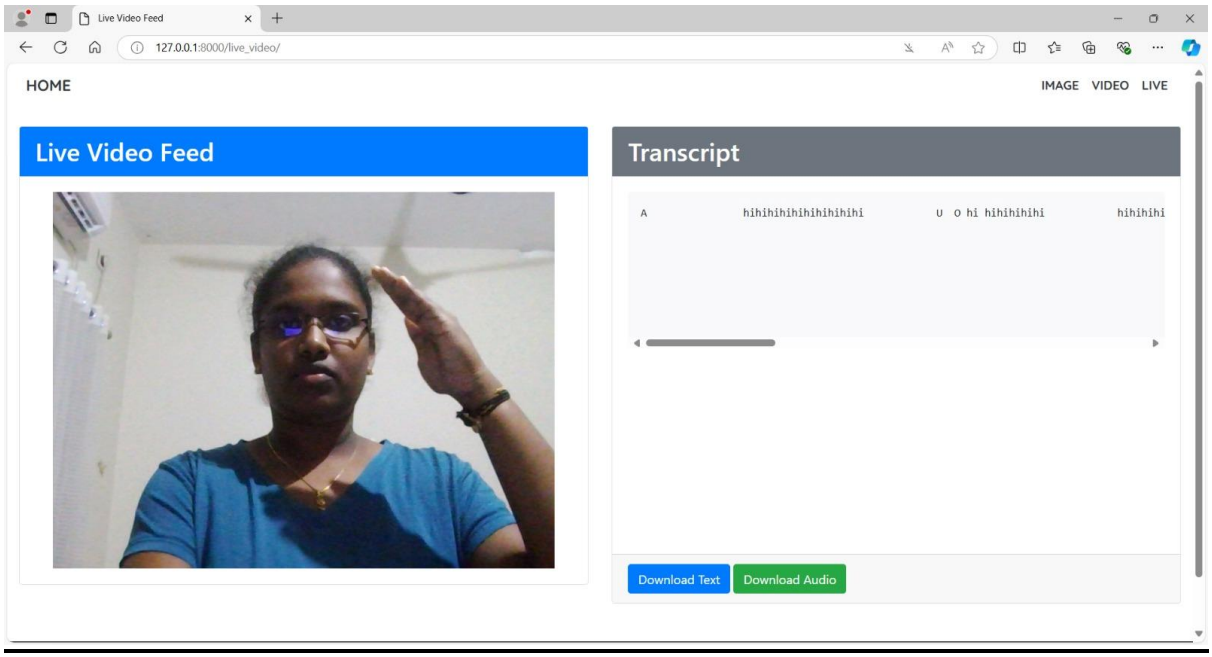

Figure A.5: *Live*

APPENDIX B- Sample Code

Function.py

```
import pandas as pd

import numpy as np

import tensorflow as tf

import cv2

import joblib

import mediapipe as mp

from tensorflow.keras.preprocessing.sequence import pad_sequences

import matplotlib.pyplot as plt

from keras.initializers import Orthogonal

import pickle

from django.conf import settings

import cv2

import numpy as np

import time

model1_path = settings.MODEL1_PATH

model1_encoder = settings.MODEL1_ENCODER

custom_objects = {'Orthogonal': Orthogonal}

loaded_model = tf.keras.models.load_model(model1_path, custom_objects=custom_objects)
```

```
mp_drawing = mp.solutions.drawing_utils

mp_hands = mp.solutions.hands

mp_pose = mp.solutions.pose

mp_face_mesh = mp.solutions.face_mesh

def extract_landmarks_mediapipe(frame):

with mp_hands.Hands(static_image_mode=False, max_num_hands=2,
min_detection_confidence=0.5) as hands:

with mp_pose.Pose(min_detection_confidence=0.5, min_tracking_confidence=0.5) as pose:

with mp_face_mesh.FaceMesh(min_detection_confidence=0.5,
min_tracking_confidence=0.5) as face_mesh:

# Convert BGR to RGB

frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

# Process hand landmarks

hands_results = hands.process(frame_rgb)

left_hand_landmarks, right_hand_landmarks = [],[]

if hands_results.multi_hand_landmarks:

for hand_landmarks, handedness in zip(hands_results.multi_hand_landmarks,
hands_results.multi_handedness):

if handedness.classification[0].label == 'Left':

left_hand_landmarks = hand_landmarks
```

```
elif handedness.classification[0].label == 'Right':

right_hand_landmarks = hand_landmarks

# Process pose landmarks

pose_results = pose.process(frame_rgb)

pose_landmarks = pose_results.pose_landmarks

# Process face landmarks

face_results = face_mesh.process(frame_rgb)

face_landmarks = face_results.multi_face_landmarks

return left_hand_landmarks, right_hand_landmarks, pose_landmarks, face_landmarks

max_face_index = 467

max_left_hand_index = 20

max_right_hand_index = 20

max_pose_index = 32

face_columns = [f"face_{i}" for i in range(max_face_index + 1)]

left_hand_columns = [f"left_hand_{i}" for i in range(max_left_hand_index + 1)]

right_hand_columns = [f"right_hand_{i}" for i in range(max_right_hand_index + 1)]

pose_columns = [f"pose_{i}" for i in range(max_pose_index + 1)]

header =      [f"{col}_{coord}" for col in face_columns for coord in ['x', 'y']] + \

[f"{col}_{coord}" for col in left_hand_columns for coord in ['x', 'y']] + \

[f"{col}_{coord}" for col in right_hand_columns for coord in ['x', 'y']] + \
```

```
[f"{col}_{coord}" for col in pose_columns for coord in ['x', 'y']]

def landmarks_to_df(left_hand_landmarks, right_hand_landmarks, pose_landmarks,
face_landmarks, header):
# Initialize dictionaries to store landmark data
landmarks_data = {}
# Process face landmarks
if face_landmarks:
for i, landmark_list in enumerate(face_landmarks):
for j, lm in enumerate(landmark_list.landmark):
landmarks_data[f"face_{j}_x"] = lm.x
landmarks_data[f"face_{j}_y"] = lm.y
# Fill missing face landmarks with zeros
for j in range(len(landmark_list.landmark), max_face_index + 1):
landmarks_data[f"face_{j}_x"] = 0.0
landmarks_data[f"face_{j}_y"] = 0.0
else:
# Fill all face landmarks with zeros if face_landmarks is None
for j in range(max_face_index + 1):
landmarks_data[f"face_{j}_x"] = 0.0
landmarks_data[f"face_{j}_y"] = 0.0
```

```
# Process left hand landmarks

if left_hand_landmarks:

for i, lm in enumerate(left_hand_landmarks.landmark):

landmarks_data[f"left_hand_{i}_x"] = lm.x

landmarks_data[f"left_hand_{i}_y"] = lm.y

# Fill missing left hand landmarks with zeros

for i in range(len(left_hand_landmarks.landmark), max_left_hand_index + 1):

landmarks_data[f"left_hand_{i}_x"] = 0.0

landmarks_data[f"left_hand_{i}_y"] = 0.0

else:

# Fill all left hand landmarks with zeros if left_hand_landmarks is None

for i in range(max_left_hand_index + 1):

landmarks_data[f"left_hand_{i}_x"] = 0.0

landmarks_data[f"left_hand_{i}_y"] = 0.0

# Process right hand landmarks

if right_hand_landmarks:

for i, lm in enumerate(right_hand_landmarks.landmark):

landmarks_data[f"right_hand_{i}_x"] = lm.x

landmarks_data[f"right_hand_{i}_y"] = lm.y

# Fill missing right hand landmarks with zeros
```

```
for i in range(len(right_hand_landmarks.landmark), max_right_hand_index + 1):

landmarks_data[f"right_hand_{i}_x"] = 0.0

landmarks_data[f"right_hand_{i}_y"] = 0.0

else:

# Fill all right hand landmarks with zeros if right_hand_landmarks is None

for i in range(max_right_hand_index + 1):

landmarks_data[f"right_hand_{i}_x"] = 0.0

landmarks_data[f"right_hand_{i}_y"] = 0.0

# Process pose landmarks

if pose_landmarks:

for i, lm in enumerate(pose_landmarks.landmark):

landmarks_data[f"pose_{i}_x"] = lm.x

landmarks_data[f"pose_{i}_y"] = lm.y

# Fill missing pose landmarks with zeros

for i in range(len(pose_landmarks.landmark), max_pose_index + 1):

landmarks_data[f"pose_{i}_x"] = 0.0

landmarks_data[f"pose_{i}_y"] = 0.0

else:

# Fill all pose landmarks with zeros if pose_landmarks is None

for i in range(max_pose_index + 1):
```

```
landmarks_data[f"pose_{i}_x"] = 0.0

landmarks_data[f"pose_{i}_y"] = 0.0

# Create DataFrame from extracted landmark data

df = pd.DataFrame([landmarks_data], columns=header)

return df

def preprocess_landmarks(df):

left_hand_columns = [col for col in df.columns if col.startswith('left_hand')]

right_hand_columns = [col for col in df.columns if col.startswith('right_hand')]

pose_columns = [col for col in df.columns if col.startswith('pose')]

# Ensure the data is in the correct shape (number_of_samples, number_of_frames,

number_of_features_per_frame)

def reshape_data(df, columns, num_frames):

data = df[columns].values

num_samples = len(df) // num_frames

data = data.reshape(num_samples, num_frames, len(columns))

return data

# Assuming num_frames is known

num_frames = 1 # This should be the length of the time series

left_hand_data = reshape_data(df, left_hand_columns, num_frames)

right_hand_data = reshape_data(df, right_hand_columns, num_frames)
```

```
pose_data = reshape_data(df, pose_columns, num_frames)

return left_hand_data, right_hand_data, pose_data

with open(model1_encoder, 'rb') as file:

label_encoder = pickle.load(file)

camera = cv2.VideoCapture(0)

def generate_frames():

prev_time = time.time()

last_prediction_time = time.time()

no_prediction_counter = 0

while True:

success, frame = camera.read()

if not success:

print('not success')

break

else:

frame = cv2.flip(frame, 1)  # Flip the frame horizontally

# Extract landmarks using MediaPipe

left_hand, right_hand, pose, face = extract_landmarks_mediapipe(frame)

if current_time - prev_time >= 1:  # Process frame every second

prev_time = current_time
```

```
if left_hand or right_hand:

# Preprocess landmarks

df = landmarks_to_df(left_hand, right_hand, pose, face, header)

left_hand_input, right_hand_input, pose_input = preprocess_landmarks(df)

# Make prediction

prediction = loaded_model.predict([left_hand_input, right_hand_input, pose_input])

if not np.isnan(prediction).any():

predicted_class_index = np.argmax(prediction, axis=1)

predicted_class_label = label_encoder.inverse_transform(predicted_class_index)

confidence = prediction[0, predicted_class_index][0]

if confidence > 0.95:

no_prediction_counter = 0

last_prediction_time = current_time

label = predicted_class_label[0]  # Assuming single prediction per frame

print(f"Predicted sign: {label} with confidence {confidence:.2f}")

with open('predicted_labels.txt', 'a') as file:

file.write(label)

else:

print("Prediction contains NaN values")

else:
```

```
if current_time - last_prediction_time >= 3:

no_prediction_counter += 1

last_prediction_time = current_time

with open('predicted_labels.txt', 'a') as file:

file.write(' ')

ret, buffer = cv2.imencode('.jpg', frame)

frame = buffer.tobytes()

yield (b'--frame\r\n'

b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
```

View.py

```
from django.shortcuts import render

from main.functions import generate_frames

from django.http import StreamingHttpResponse

import logging

import time

from django.http import HttpResponse

from gtts import gTTS

import tempfile

def home(request):

return render(request, 'main/index.html')
```

```
def login(request):

# Implement this view

return render(request, 'main/login.html')

def upload_image(request):

# Implement this view

return render(request, 'main/image.html')

def process_image(request):

# Implement this view

pass

def upload_video(request):

# Implement this view

return render(request, 'main/video.html')

def process_video(request):

# Implement this view

pass

def live_video(request):

# Implement this view

return render(request, 'main/live.html')

def video_feed(request):
```

```
return StreamingHttpResponse(generate_frames(), content_type='multipart/x-mixed-replace;
boundary=frame')

logger = logging.getLogger(__name__)

def stream_predictions(request):

def event_stream():

last_sent_data = ""

while True:

try:

with open('predicted_labels.txt', 'r') as file:

data = file.read().replace('[', '').replace(']', '').replace("'", "").replace('"', '').strip()

if data and data != last_sent_data:

last_sent_data = data

yield f"data: {data}\n\n"

except Exception as e:

logger.error(f"Error reading file: {e}")

time.sleep(1)  # Adjust the sleep duration as needed

return StreamingHttpResponse(event_stream(), content_type='text/event-stream')

def download_txt(request):

# Logic to generate or retrieve the text content

content = "This is the transcript content."
```

```
response = HttpResponse(content, content_type='text/plain')

response['Content-Disposition'] = 'attachment; filename="transcript.txt"'

return response

def download_audio(request):

# Get the text content (you might want to pass this from the front-end or retrieve it from a
database)

text_content = "This is the transcript content that will be converted to speech."

# Create a gTTS object

tts = gTTS(text=text_content, lang='en')

# Create a temporary file

with tempfile.NamedTemporaryFile(delete=False, suffix='.mp3') as fp:

# Save the speech to the temporary file

tts.save(fp.name)

# Reopen the file and create the response

with open(fp.name, 'rb') as audio_file:

response = HttpResponse(audio_file.read(), content_type="audio/mpeg")

response['Content-Disposition'] = 'attachment; filename="transcript_audio.mp3"'

# Delete the temporary file

os.unlink(fp.name)

return response
```

APPENDIX C- ANTI-PLAGIARISM STATEMENT

I certify that this project report is my work and is based on my study and research and that I have acknowledged all materials and sources used in its preparation, whether they be books, articles, reports, lecture notes, or any other kind of document, electronic or personal communication. I also certify that this report has not previously been submitted for assessment in any other unit, except where specific permission has been granted from all unit coordinators involved, or at any other time in this unit, and that I have not copied in part or whole or otherwise plagiarized the work of other students and persons.

Name: Date:

Signature:

REFERENCES

[1] Shruti Chavan, Xinrui Yu and Jafar Saniie Embedded Computing and Signal Processing Research Laboratory "*Convolutional Neural Network Hand Gesture Recognition for American Sign Language*" (2021) (http://ecasp.ece.iit.edu/).

[2] Surekha P, Pranavi Duggirala, Niharika Vitta, Venkata Surya Saranya Ambadipudi and Teja Sree Desani, Computer Science Engineering Gokaraju Rangaraju Institute of Engineering and Technology Hyderabad, India, "*Hand Gesture Recognition and voice, text conversion using CNN and ANN*" (2022).

[3] Tiya Ann Siby, Sonam Pal, Jessica and Shamanth Nagaraju, Department of Computer Science & Engineering School of Engineering & Technology Christ (Deemed-to-be University) Bengaluru, India. "*Gesture based Real-Time Sign Language Recognition System*" (2022).

[4] Swati Nitnaware, Department of Electronics And Telecommunication, Yeshwantrao Chavan College of Engineering Nagpur and Ashutosh Bagde, Jawaharlal Nehru Medical College, Datta Meghe Institute of Medical Univ Sciences, Sawangi (M), Wardha, "*Hand Gesture Recognition to Facilitate Tasks for the Disabled*" (2022).

[5] S.Gnanapriya, K.Rahimunnisa, M.Sowmiya, P.Deepika and S.Praveena Rachel Kamala, Department of Information Technology, Easwari Engineering College, Anna University, India, "*Hand Detection and Gesture Recognition in Complex Backgrounds*" (2023)

[6] Shalvee Meshram and Roshan Singh Dept. of ESE, National Institute of Electronics, and Information Technology Aurangabad, Maharashtra, India and Prashant Pal, Shashank Kumar Singh Scientist B National Institute of Electronics and Information Technology Aurangabad, Maharashtra, "*Convolution Neural Network based Hand Gesture Recognition System*" (2023)

[7] Maria Papatsimouli, Konstantinos-Filippos Kollias, Lazaros Lazaridis, George Maraslidis, Herakles Michailidis, Panagiotis Sarigiannidis and George F. Fragulis Department of Electrical and Computer Engineering University of Western Macedonia, Kozani, Greece, "*Real Time Sign Language Translation Systems: A review study*".

[8] Dongxu Li, Cristian Rodriguez Opazo, Xin Yu, Hongdong Li, "*Word-level Deep Sign Language Recognition from Video: A New Large-scale Dataset and Methods Comparison*" (2019).

[9] Necati Cihan Camgoz, Simon Hadfield, Oscar Koller, Hermann Ney, Richard Bowden "*Neural Sign Language Translation*" (2018).

[10] Zeyu Liang, Huailing Li and Jianping Chai, "*Sign Language Translation: A Survey of Approaches and Techniques*".

[11] JOUR Johnny, Swapna, Nirmala, Jaya, "*Sign Language Translator Using Machine Learning*".

[12] Ming Jin Cheok1 · Zaid Omar1 · Mohamed Hisham Jaward2,"*A review of hand gesture and sign language recognition techniques*", Int. J. Mach. Learn. & Cyber. © Springer-Verlag GmbH Germany 2017.

[13] Soeb Hussain and Rupal Saxena, Xie Han, Jameel Ahmed Khan, Hyunchul Shin, "*Hand Gesture Recognition Using Deep Learning*" , International SoC Design Conference (ISOCC), 2017.

[14] Neel Kamal Bhagat, Vishnusai Y, Rathna G N, "*Indian Sign Language Gesture Recognition using Image Processing and Deep Learning*", Digital Image Computing: Techniques and Applications (DICTA)IEEE, 2019.

[15] N. A. Ahmad, "*A Globally Convergent Stochastic Pairwise Conjugate Gradient-Based Algorithm for Adaptive Filtering*," in IEEE Signal Processing Letters, vol. 15, pp. 914-917, 2008, doi: 10.1109/LSP.2008.2005437.

[16] S. Mitra and T. Acharya, "*Gesture Recognition: A Survey*," in IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 37, no. 3, pp. 311-324, May 2007, doi: 10.1109/TSMCC.2007.893280.

[17] N. H. Dardas and N. D. Georganas, "*Real-Time Hand Gesture Detection and Recognition Using Bag-of-Features and Support Vector Machine Techniques*," in IEEE Transactions on Instrumentation and Measurement, vol. 60, no. 11, pp. 3592-3607, Nov. 2011, doi: 10.1109/TIM.2011.2161140.

[18] P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree and J. Kautz, "*Online Detection and Classification of Dynamic Hand Gestures with Recurrent 3D Convolutional Neural Networks*," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 4207-4215, doi: 10.1109/CVPR.2016.456.

[19] P. Molchanov, S. Gupta, K. Kim and J. Kautz, "*Hand gesture recognition with 3D convolutional neural networks*," 2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2015, pp. 1-7, doi: 10.1109/CVPRW.2015.7301342

[20] S. K. Shareef, I. V. S. L. Haritha, Y. L. Prasanna and G. K. Kumar, "*Deep Learning Based Hand Gesture Translation System*," 2021 5th International Conference on Trends in Electronics and Informatics (ICOEI), 2021, pp. 1531-1534, doi: 10.1109/ICOEI51242.2021.9452947.

[21] S. Hussain, R. Saxena, X. Han, J. A. Khan and H. Shin, "*Hand gesture recognition using deep learning*," 2017 International SoC Design Conference (ISOCC), 2017, pp. 48-49, doi: 10.1109/ISOCC.2017.8368821.

[22] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar and L. Fei-Fei, "*Large-Scale Video Classification with Convolutional Neural Networks,*" 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1725-1732, doi: 10.1109/CVPR.2014.223.

[23] Jie Huang, Wengang Zhou, Houqiang Li and Weiping Li, "*Sign Language Recognition using 3D convolutional neural networks*," 2015 IEEE International Conference on Multimedia and Expo (ICME), 2015, pp. 1-6, doi: 10.1109/ICME.2015.7177428.

[24] V. de Oliveira Silva, F. de Barros Vidal and A. R. Soares Romariz, "*Human Action Recognition Based on a Two-stream Convolutional Network Classifier*," 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), 2017, pp. 774-778, doi: 10.1109/ICMLA.2017.00-64.

[25] P. Y. Simard, D. Steinkraus and J. C. Platt, "*Best practices for convolutional neural networks applied to visual document analysis*," Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings., 2003, pp. 958-963, doi: 10.1109/ICDAR.2003.1227801.

[26] S. Tayeb et al., "*Toward data quality analytics in signature verification using a convolutional neural network*," 2017 IEEE International Conference on Big Data (Big Data), 2017, pp. 2644- 2651, doi: 10.1109/BigData.2017.8258225.