A Mini Project report on

**BIDIRECTIONAL SIGN LANGUAGE COMMUNICATION SYSTEM**

A documentation submitted in partial fulfillment of the academic requirement for the award of degree of

## BACHELOR OF ENGINEERING

in

## ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

by

## SYED IBRAHIM ALI (1604-22-748-010)

## MOHAMMED MUDASIR AHMED (1604-22-748-012)

## OZAIR ALI (1604-22-748-014)

Under the guidance of

### Mrs. NISHAT AFZA

Assistant Professor, CS&AI Dept., MJCET



## COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE DEPARTMENT

MUFFAKHAM JAH COLLEGE OF ENGINEERING AND TECHNOLOGY

(Affiliated to Osmania University) Hyderabad.

2024 – 2025

|  |  |
| --- | --- |
|  | **MUFFAKHAM JAH COLLEGE OF ENGINEERING AND TECHNOLOGY** |
| **Mount Pleasant, 8-2-249 to 267, Road No.3, Banjara Hills,**  **Hyderabad-500 034, Telangana State, India.** |

## CERTIFICATE

This is to certify that the mini project report on **“BIDIRECTIONAL SIGN LANGUAGE COMMUNICATION SYSTEM”** is a bonafide work carried out by **SYED IBRAHIM ALI (1604-22-748-010), MOHAMMED MUDASIR AHMED (1604-22-748-012)** and **OZAIR ALI (1604-22-748-014)** in the partial fulfillment of the requirements for the award of the B.E. CSE(AI&ML) in MUFFAKHAM JAH COLLEGE OF ENGINEERING AND TECHNOLOGY, Hyderabad for the academic year 2024-2025.

|  |  |
| --- | --- |
| **Internal Guide** | **Head of the Department** |
| **Mrs. Nishat Afza** | **Prof. Uma N. Dulhare** |
| **Assistant Professor,** | **Professor & Head,** |
| **CS&AI Dept., MJCET,** | **CS&AI Dept., MJCET,** |
| **Hyderabad** | **Hyderabad** |

**External Examiner**

## DECLARATION

We hereby declare that the work entitled “**BIDIRECTIONAL SIGN LANGUAGE COMMUNICATION SYSTEM”** developed under the supervision of **Mrs. NISHAT AFZA, Assistant Professor, CS&AI Department** and submitted to **MUFFAKHAM JAH COLLEGE OF ENGINEERING AND TECHNOLOGY** is original and has not been submitted in part or while for under graduation degree to any other university.

**Syed Ibrahim Ali**

**(1604-22-748-010)**

**Mohammed Mudasir Ahmed**

**(1604-22-748-012)**

**Ozair Ali**

**(1604-22-748-014)**

## ACKNOWLEDGEMENT

Our sincere thanks and gratitude to **Prof. UMA N. DULHARE, Professor & Head of the Department** for all the timely support and valuable suggestions during the period of my mini project.

We are extremely thankful to Project Review Committee Members, Dept. of CS&AI for their encouragement and support throughout the project.

We are extremely thankful and indebted to our internal guide**, Mrs. NISHAT AFZA, Assistant Professor, Department of CS&AI** for her exemplary guidance and encouragement. Her trust and support inspired us in the most important moments of making right decisions and we are glad to work under her supervision.

We would like to thank all the faculty and staff of department who helped us directly or indirectly in completing the project work. Last but not least, we would like to thank my family members and friends for their co-operation in completing the project.

**Syed Ibrahim Ali**

**(1604-22-748-010)**

**Mohammed Mudasir Ahmed**

**(1604-22-748-012)**

**Ozair Ali**

**(1604-22-748-014)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| TABLE OF CONTENTS | | | | | | |
| **CONTENT** | | | | | | **PAGE NO.** |
|  | | | | | |  |
| ABSTRACT | | | | | | iii |
| LIST OF FIGURES | | | | | | iv |
| **CHAPTERS** | | | | | |  |
| **1.** | **INTRODUCTION** | | | | | **1** |
|  |  | 1.1 Introduction | | | | 1 |
|  |  | 1.2 Objectives | | | | 2 |
|  |  | 1.3 Problem Statement | | | | 3 |
|  |  | 1.4 Scope | | | | 3 |
| **2.** | **LITERATURE SURVEY** | | | | | **4** |
| **3.** | **EXISTING SYSTEM** | | | | | **7** |
|  |  | 3.1 Introduction | | | | 7 |
|  |  | 3.2 Problems with Existing System | | | | 8 |
| **4.** | **PROPOSED SYSTEM** | | | | | **9** |
|  |  | 4.1 Introduction | | | | 9 |
|  |  | 4.2 System Architecture | | | | 10 |
|  |  |  | | 4.2.1 Working Structure of Bidirectional Sign Language Communication System | | 10 |
|  |  |  | | 4.2.2 Working Process | | 10 |
| **5.** | **SYSTEM DESIGN** | | | | | **12** |
|  |  | 5.1 Uml Diagrams | | | | 12 |
|  |  |  | | 5.1.1 Class Diagram | | 12 |
|  |  |  | | 5.1.2 Use Case Diagram | | 13 |
|  |  |  | | 5.1.3 Component Diagram | | 15 |
|  |  |  | | 5.1.4 Sequence Diagram | | 16 |
|  |  |  | | 5.1.5 Activity Diagram | | 18 |
| **6.** | **METHODOLOGIES** | | | | | **20** |
|  |  | 6.1 Components | | | | 20 |
|  |  | 6.2 Technologies | | | | **22** |
|  |  |  | | | |  |
|  |  |  | | | |  |
|  |  |  | | | |  |
|  |  |  | | | |  |
| **7.** | **IMPLEMENTATION** | | | | | **24** |
|  |  | 7.1 Requirements | | | | 24 |
|  |  |  | | | 7.1.1 Overall Description | 24 |
|  |  |  | | | 7.1.2 Software Requirements | 24 |
|  |  |  | | | 7.1.3 Hardware Requirements | 24 |
|  |  | 7.2 Code Snippets | | | | 25 |
|  |  | 7.3 Execution | | | | 27 |
|  |  |  | | | 7.3.1 Sign Alphabets Detection-To-Speech | 27 |
|  |  |  | | | 7.3.2 Sign Number Detection-To-Speech | 27 |
|  |  |  | | | 7.3.3 Speech-To-Sign | 28 |
|  |  |  | | | 7.3.4 Text-To-Sign | 29 |
| **8.** | **CONCLUSION AND FUTURE ENHANCEMENTS** | | | | | **30** |
|  |  | | 7.1 Conclusion | | | 30 |
|  |  | | 7.2 Future Enhancement | | | 30 |
|  | **REFERENCES** | | | | | **32** |

**ABSTRACT**

Effective communication between deaf and hearing communities is often impeded by language barriers, necessitating innovative solutions. Our project introduces a real-time sign language translation application designed to convert sign language gestures into audible speech, thereby facilitating seamless interaction. The system employs advanced computer vision and machine learning techniques to capture and interpret hand gestures. A camera records the user's hand movements, which are processed by a trained neural network to recognize specific gestures. These gestures are then mapped to corresponding words or phrases and synthesized into audible speech, enabling effective communication between deaf and hearing individuals. Preliminary evaluations demonstrate promising accuracy in both gesture recognition and speech synthesis, highlighting the system's potential across various settings. This technology not only promotes inclusivity and accessibility for individuals with hearing impairments but also offers adaptability to multiple sign languages, broadening its applicability. Future developments may include expanding the gesture database, enhancing recognition accuracy, and incorporating features such as real-time feedback to further improve communication effectiveness. This real-time sign language translation application represents a significant advancement in bridging the communication divide between deaf and hearing communities, fostering a more inclusive society.

**Keywords:** Gesture Recognition, Computer Vision, Deep Learning, MLP Classifier, Hand Landmarks, Real-Time Processing

# LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **NAME OF THE FIGURE** | **PAGE NO.** |
| 1.1 | ISL Numbers and Alphabets | 2 |
| 4.1 | Architecture Diagram | 10 |
| 5.1 | Class Diagram | 13 |
| 5.2 | Use Case Diagram | 14 |
| 5.3 | Component Diagram | 16 |
| 5.4 | Sequence Diagram | 17 |
| 5.5 | Activity Diagram | 19 |
| 6.1 | Bounding Box | 20 |
| 6.2 | Gesture Detection and Hand Segmentation | 21 |
| 6.3 | MLP Classifier | 21 |
| 6.4 | Hand Landmarks | 22 |
| 7.1 | Data Collection | 25 |
| 7.2 | Dataset Processing | 25 |
| 7.3 | Model Training | 25 |
| 7.4 | Sign Recognition | 26 |
| 7.5 | Real Time Sign-to-Speech | 26 |
| 7.6 | Text-to-Sign | 26 |
| 7.7 | Speech-to-Sign | 26 |
| 7.8 | Detected Alphabet P | 27 |
| 7.9 | Detected Alphabet Z | 27 |
| 7.10 | Detected Alphabet U | 27 |
| 7.11 | Detected Numbers 2 and 6 | 28 |
| 7.12 | Speech-To-Sign | 28 |
| 7.13 | Text-To-Sign | 29 |

**CHAPTER 1**

**INTRODUCTION**

## Introduction

Effective communication is a cornerstone of human interaction, yet individuals with hearing and speech impairments often encounter significant barriers due to the reliance on spoken language. Sign language serves as a primary mode of communication for the deaf and hard-of-hearing community, enabling them to convey their thoughts and emotions through structured hand gestures and facial expressions. A substantial communication gap persists between sign language users and those unfamiliar with this visual language, leading to social isolation and misunderstandings. According to World Health Organization (WHO) estimates, there are over 63 million significant auditory impairment sufferers in India, putting the estimated prevalence at 6.3% of the population. Therefore, there is a need for developing a system that bridges the communication gap with non-sign language users.

Sign language recognition is an important research area since there are a lot of challenges in developing an automatic recognition system. Most of the researchers in this area concentrate on the recognition of American Sign Language (ASL) since most of the signs in ASL are single handed and thus, complexity is less. Another attractive feature is that ASL already has a standard database that is available for use. When compared with ASL, Indian Sign Language (ISL) relies on both hands and thus, an ISL recognition system is more complex. The research works carried out by the researchers in the recognition of ISL is very less.

Existing solutions in this domain have explored various approaches, including glove-based devices equipped with sensors to detect hand movements and positions. While effective, these devices can be expensive and cumbersome, limiting their widespread adoption. Vision-based methods, utilizing cameras to capture gestures, offer a more user-friendly and cost-effective alternative. Recent studies have demonstrated the potential of deep learning models in accurately recognizing sign language gestures from video inputs, paving the way for innovative applications in this field.

To address this challenge, our project focuses on developing a real-time sign language-to-speech and speech-to-sign language conversion system. By leveraging advancements in computer vision and machine learning, the system aims to interpret sign language gestures and translate them into audible speech, thereby facilitating seamless communication between sign language users and the broader, non-signing population.

In our project by specifically tailoring our system to ISL, we focus on producing a model which can recognize Fingerspelling based hand gestures. The gestures we aim to train are as given in the image below:



**Fig 1.1** ISL Numbers and Alphabets

## Objectives

The objectives of our project are:

* To develop a real-time system that translates Indian Sign Language(ISL) gestures into audible speech.
* To generate Indian Sign Language(ISL) gestures from text or speech.
* To prepare a working model that is user-friendly, robust and scalable.
  1. **Problem Statement**

Communication barriers between the deaf and hearing communities create significant challenges in daily interactions, education, and employment. Existing Indian Sign Language (ISL) translation systems often suffer from limited gesture datasets, high latency, poor real-time accuracy, dependency on expensive hardware, and lack of bidirectional translation.

Our project aims to develop an efficient, real-time bidirectional ISL communication system using computer vision and deep learning. This system leverages MediaPipe landmarks for hand tracking, MLP classifiers for gesture recognition, and real-time text processing, the system ensures seamless two-way communication between ISL users and non-signers, all without requiring specialized hardware beyond a webcam.

* 1. **Scope**

The project encompasses the development of a real-time system that translates sign language gestures into audible speech, aiming to facilitate effective communication between individuals who use sign language and those who do not. The key components and boundaries of the project are Gesture Recognition, Real-Time Processing, Speech Synthesis, User Interface(UI)

**CHAPTER 2**

**LITERATURE SURVEY**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SNO** | **Title of paper** | **Methodology** | **Results** | **Advantages** | **Drawbacks** |
| **1** | **Sign Language Recognition Using Convolutional Neural Networks (2015)** | This study employs Convolutional Neural Networks (CNNs) to recognize 20 Italian sign language gestures. The system utilizes the Microsoft Kinect sensor to capture depth images, which are then processed by the CNN to identify specific gestures. | This study employs Convolutional Neural Networks (CNNs) to recognize 20 Italian sign language gestures. The system utilizes the Microsoft Kinect sensor to capture depth images, which are then processed by the CNN to identify specific gestures. | Automated feature extraction reduces the need for manual intervention.  High recognition accuracy enhances communication between the Deaf community and the hearing majority. | Limited to recognizing a predefined set of gestures, which may not encompass the full range of sign language expressions. Dependence on specific hardware (Microsoft Kinect) may limit accessibility. |
| **2** | **Indian Sign Language Recognition Using Mediapipe Holistic (2023)** | Employed MediaPipe Holistic for feature extraction, capturing hand, face, and pose landmarks. Implemented Long Short-Term Memory (LSTM) networks to process temporal sequences of gestures. | Demonstrated that CNNs are effective for static sign language recognition, while LSTMs excel in dynamic gesture recognition. | Comprehensive feature extraction from multiple body parts.  LSTM networks effectively handle temporal dependencies in dynamic gestures. | Complexity in integrating multiple body part features.  Potential computational overhead due to processing multiple landmarks. |
| **3** | **Dynamic Sign Language Recognition Using MediaPipe Library and Modified LSTM Method (2023)** | This research utilizes the MediaPipe Holistic library to extract keypoints from video frames, capturing both static and dynamic aspects of sign language gestures. The extracted data is processed using a modified LSTM network to recognize dynamic sign language movements. | The system achieved an average accuracy of 99.4% in recognizing 20 dynamic sign language words, with individual word detection accuracies reaching up to 85%. | High accuracy in recognizing dynamic gestures enhances real-time communication.  Utilization of MediaPipe Holistic allows for comprehensive gesture analysis. | Requires high-quality video input for optimal performance.  Faces challenges in recognizing gestures performed by different individuals due to variability. |
| **4** | **Indian Sign Language Translation Using HMM and Deep Learning (2021)** | Combines Hidden Markov Models (HMM) with deep learning-based feature extraction for ISL translation. Uses an extensive dataset for training. | Achieves higher translation accuracy by incorporating dynamic hand movements. | Addresses both static and dynamic gestures, making it more effective for real-time communication. | Requires high computational resources and dataset expansion for full ISL vocabulary coverage. |
| **5** | **Recognition of Indian Sign Language (ISL) Using Deep Learning Model (2021)** | Employs skin color segmentation and face elimination for hand tracking, followed by SVM-based classification  . | Improves real-time gesture recognition in ISL using a vision-based approach. | Reduces dependency on hardware devices like gloves or depth sensors, making it widely accessible. | Background noise and varying lighting conditions affect segmentation accuracy. |
| **6** | **Sign Language Recognition Using Wearable Electronics: Implementing k-Nearest Neighbors with Dynamic Time Warping and Convolutional Neural Network Algorithms (2020)** | This study develops a wearable device equipped with sensors to capture hand and finger movements. The collected data is analyzed using k-Nearest Neighbors (k-NN) with Dynamic Time Warping (DTW) and Convolutional Neural Network (CNN) algorithms to recognize sign language gestures. | The system demonstrated effective recognition of sign language gestures, with performance metrics indicating the feasibility of wearable electronics in sign language recognition. | Wearable devices offer portability and ease of use.  Combining k-NN, DTW, and CNN algorithms enhances recognition accuracy. | Wearable devices may be uncomfortable for prolonged use. Sensor calibration and maintenance can be challenging. |

**CHAPTER 3**

**EXISTING SYSTEM**

## 3.1 Introduction

Existing systems for sign language recognition have evolved significantly, employing various methodologies to bridge communication gaps between the Deaf and hearing communities[1]. The procedure used in various research papers relating to the project are discussed below:

**Vinay J, Roopashree M** and **Dr. Hema N** have proposed that this study introduced a bidirectional ISL Translator using gesture recognition. The system converts ISL gestures into text/audio and vice versa using Convolutional Neural Networks (CNN) for gesture recognition and Natural Language Processing (NLP) for translation. An animated ISL avatar enhances visualization. The system ensures real-time, multilingual support with minimal latency, promoting accessibility and inclusivity.

**M. Suresh Anand, A. Kumaresan,** and **Dr. N. Mohan Kumar** have proposed that this study developed a vision-based two-way ISL translation system. It captures gestures via a webcam, processes images through binarization and feature extraction, and translates them into text/speech. Pattern recognition techniques match hand movements to a trained database, while text-to-sign mapping converts speech into ISL animations. This eliminates the need for wearable devices, making it more affordable and accessible.

**Mr. P. Jagannadha Varma, S. Amulya** and **T. Roshitha** have proposed that this study develops a real-time ISL translation system using computer vision and deep learning. The system captures gestures via a camera, preprocesses images using edge detection, and classifies signs with Support Vector Machines (SVM). Speech-to-text processing employs Hidden Markov Models (HMM), while text-to-sign conversion uses an ISL animation module. This system enhances accessibility in public services, education, and healthcare.

**Shaheen Aaliya Khan, Jay Patni** and **Maliha Ashfaque Khan** have proposed that this study explored a real-time bidirectional communication system for Indian Sign Language (ISL) using computer vision and deep learning. The system captures hand gestures via a camera, preprocesses images using Gaussian filters for noise removal, and applies OTSU’s method for binary conversion. A Convolutional Neural Network (CNN) model is then trained on processed images to recognize signs and convert them into text or speech. The reverse process involves speech-to-text conversion followed by text-to-sign translation using an ISL animation module, enabling seamless interaction between ISL users and non-signers.

**Prof. Kirti Balsaraf,** and **Prasanna Phadtare** have proposed that this study introduced a bidirectional ISL interpreter utilizing deep learning and image processing techniques. The methodology involves capturing real-time ISL gestures using a camera, preprocessing images with grayscale conversion and edge detection, and extracting features using CNN for sign recognition. The system also converts spoken language to ISL gestures by processing audio inputs through a speech recognition module, followed by text-to-gesture mapping using OpenCV. This approach ensures an efficient, real-time translation between ISL and spoken language, promoting inclusivity.

## 3.2 Problems with Existing System

Existing sign language recognition systems have made significant strides in facilitating communication between the Deaf and hearing communities. However, several challenges persist:

* **Variability in Sign Language:** Sign languages exhibit regional and individual variations, leading to inconsistencies in gesture production. This variability complicates the development of universal recognition systems.
* **Temporal Segmentation Challenges:** Accurately segmenting continuous sign language into individual signs remains a complex task. Errors in segmentation can propagate, affecting the overall recognition accuracy.
* **Limited Datasets:** The scarcity of large, annotated datasets hampers the training of robust models. Many existing datasets are small, lack diversity, or are not representative of real-world scenarios.
* **Computational Demands:** Advanced recognition models often require substantial computational resources, making them less accessible for real-time applications or deployment on resource-constrained devices.
* **Real-Time Processing:** Achieving real-time recognition with high accuracy is challenging, especially in dynamic environments where lighting conditions and backgrounds vary.
* **High Latency & Slow Processing:** Older models (SVM, HMM) are computationally expensive, causing delays.
* **Lack of Standardization:** The absence of standardized sign language datasets and evaluation metrics makes it difficult to compare and benchmark different recognition systems effectively.

**CHAPTER 4**

**PROPOSED SYSTEM**

## Introduction

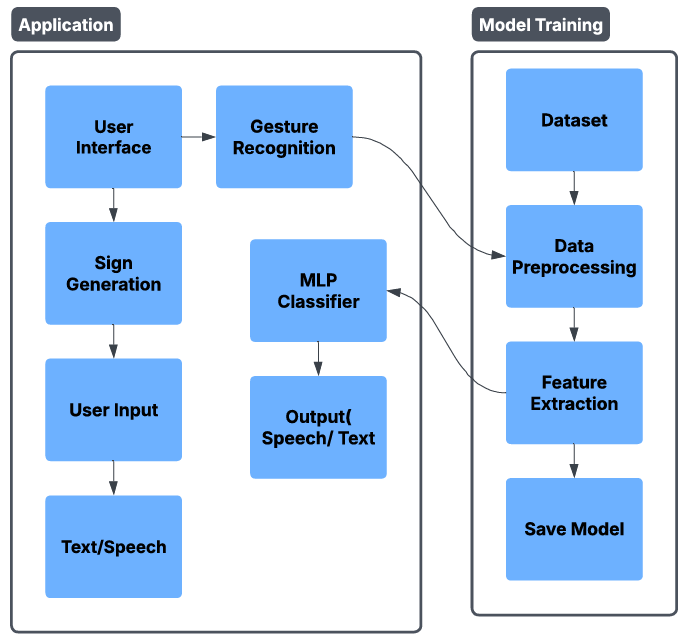
To address the challenges identified in existing sign language recognition systems, we propose a comprehensive system that integrates advanced computer vision techniques, deep learning models, and real-time processing capabilities.

Our proposed system consists of the following:

* The system consists of text-to-sign, speech-to-sign, and sign-to-speech functionalities.
* It leverages computer vision, and deep learning to recognize hand gestures and convert them into text or speech.
* The system integrates MediaPipe Hands for hand tracking, MLP-based classification for sign recognition, and Google TTS for speech output.
* The application is built using Streamlit for an interactive user interface.
* Preprocessing includes landmark extraction, normalization, and data augmentation to improve model accuracy.
* Training is performed using MLP classifiers trained on extracted features.
* The system supports real-time recognition using webcam-based inference models.
* User settings allow customization of voice, speech rate, and volume for better accessibility.

## System Architecture

### 4.2.1 Working Structure of Bidirectional Sign Language communication System



**Fig 4.1** Architecture Diagram

### 4.2.2 Working Process

#### Step – 1: Dataset Creation

The dataset is created by capturing images and videos of sign language gestures for each letter (A-Z) and number (0-9), using MediaPipe for hand landmark extraction.

**Step – 2: Preprocessing**

The collected images and videos are preprocessed to ensure consistency in size, resolution, and color scale. Data augmentation techniques (like rotations, flips, and zooms) are applied to increase dataset diversity.

#### Step – 3: Feature Extraction

Hand gestures are analyzed using MediaPipe to extract 42 or 84 hand landmarks, representing finger and palm positions, which serve as the input features for deep learning models.

#### Step – 4: Pickle file

The Pickle file contains the features along with their labels for training the model.

**Step – 5: Splitting Dataset**

The dataset is split into train set and test set. (80% for training and 20% for testing)

**Step – 6: Model Training**

In this step, the model is trained on labeled dataset, and the necessary algorithms are used to make the machine learning model efficient and increase the accuracy.

**Step – 7: Testing Model**

In this step, the trained model is tested for accuracy on test dataset to prevent overfitting.

**Step – 8: Saving the Model**

If the accuracy score of the trained model is satisfactory, the model is saved as a ‘.p’ file for further use in gesture prediction.

**Step – 9:** **Real-Time Gesture Recognition**

Live video input is captured and preprocessed. The trained MLP Classifiermodel recognizes static gestures, providing text output.

**Step – 10: Output (Text/Speech)**

Recognized gestures are displayed as text on-screen. Using gTTS, the text is converted to speech for auditory output

**CHAPTER 5**

**SYSTEM DESIGN**

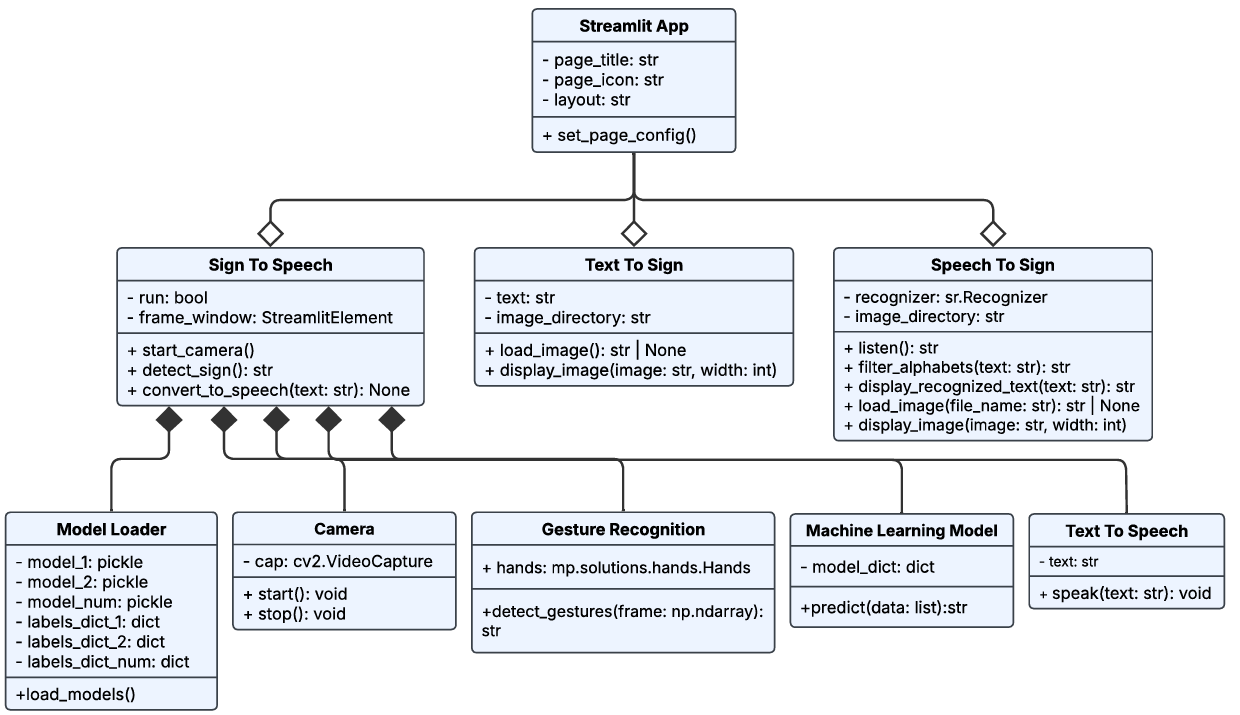
* 1. **UML Diagrams**

**5.1.1 Class Diagram**

The class diagram is a visual representation of the structure and relationships between the classes in the Streamlit application designed for sign language recognition and conversion. It describes the classes, their attributes (variables), and methods (functions), as well as how they interact with each other.

The various classes used in our project along with their functionalities are described as follows:

* **Streamlit App**: This is the main application class that sets up the page configuration with a title and icon. It uses the **set\_page\_config()** method to configure these settings.
* **Sign To Speech**: This class handles the conversion of sign language to speech. It includes methods to start the camera (start\_camera()), detect signs (detect\_sign()), and convert the detected text to speech (convert\_to\_speech(text: str)).
* **Text To Sign**: This class is responsible for converting text to sign language images. It includes methods to load images (load\_image()) and display them (display\_image(image: str, width: int)).
* **Speech To Sign**: This class converts spoken language to sign language. It uses a speech recognizer (sr.Recognizer) to listen to speech (listen()), filter alphabets from the recognized text (filter\_alphabets(text: str)), and display the recognized text and corresponding sign images (display\_recognized\_text(text: str)).
* **Model Loader**: This class loads machine learning models and their corresponding label dictionaries. It includes a method (load\_models()) to load the models and labels.
* **Camera**: This class manages the camera operations. It includes methods to start (start()) and stop (stop()) the camera.
* **Gesture Recognition**: This class is responsible for detecting gestures using mediapipe's hands module (mp.solutions.hands.Hands). It includes a method (detect\_gestures(frame: np.ndarray)) to detect gestures from a video frame.
* **Machine Learning Model**: This class contains a dictionary of models (model\_dict) and a method (predict(data: list)) to make predictions based on input data.
* **Text To Speech**: This class converts text to speech. It includes a method (speak(text: str)) to speak the given text.



**Fig 5.1** Class Diagram

* + 1. **Use Case Diagram**

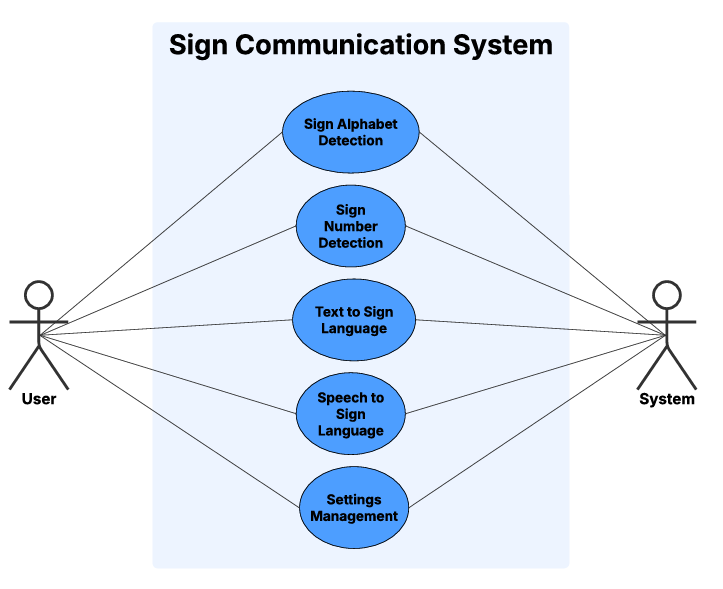
  Use case diagrams are a type of UML (Unified Modeling Language) diagram that represent the functional requirements of a system by showing the interactions between users (actors) and the system.

**Key Components of the Use Case Diagram:**

* **User**:
  + The **User** is the primary actor in this diagram. This represents the person who interacts with the **Sign Communication System**. The user could be someone who wants to use the system for sign language recognition, conversion, or communication.
* **System**:
  + The **System** represents the **Sign Communication System** itself. This is the software application that provides functionalities such as sign language recognition, text-to-sign conversion, speech-to-sign conversion, and sign-to-speech conversion.

**Use Cases:**

* **Sign Alphabet Detection** – Detects alphabets from hand gestures.
* **Sign Number Detection** – Detects numbers from hand gestures.
* **Text to Sign Language** – Converts entered text into sign language images.
* **Speech to Sign Language** – Converts spoken letters ("letterA", "letterB", etc.) into sign language images.
* **Sign to Speech** – Converts sign language gestures into spoken words.
* **Settings Management** – Allows users to configure application settings.



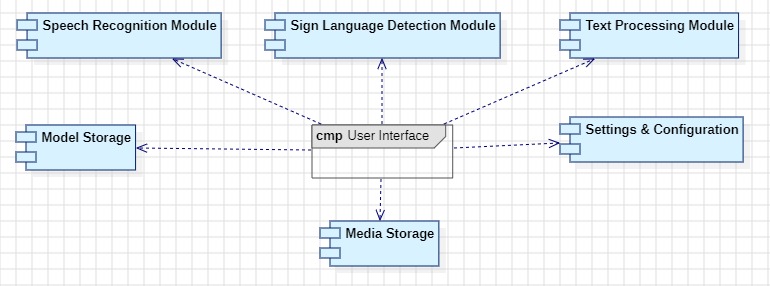
**Fig 5.2** Use Case Diagram

* + 1. **Component Diagram**

A **component diagram**, which is a type of UML (Unified Modeling Language) diagram used to represent the structure and organization of a system's components and their relationships. It is a structural diagram that visualizes the high-level organization of a system by breaking it down into its functional or logical components. It shows how these components interact with each other and with external systems or users. Here, the component diagram represents the components of **Sign Communication System** and its key modules.

**Main Components:**

* **User Interface (UI)**
  + Streamlit-based frontend for user interaction.
  + Displays images, buttons, and text input.
* **Speech Recognition Module**
  + Uses speech\_recognition to convert spoken words into text.
  + Works with the **Speech to Sign Language** module.
* **Sign Language Detection Module**
  + Uses trained ML models for **Sign Alphabet** and **Sign Number Detection**.
  + Detects hand gestures and maps them to corresponding characters.
* **Text Processing Module**
  + Converts typed text into sign language representation.
  + Works with **Text to Sign Language** and **Sign to Speech** features.
* **Model Storage**
  + Stores machine learning models for sign detection.
  + Includes CNN-based and hand landmark-based models.
* **Media Storage**
  + Stores sign images (A-Z, 0-9) for **Text to Sign** and **Speech to Sign** modules.
  + Stores generated speech files (if required) using gTTS.
* **Settings & Configuration**
  + Allows users to configure preferences.
  + Handles language settings, input options, etc.

****

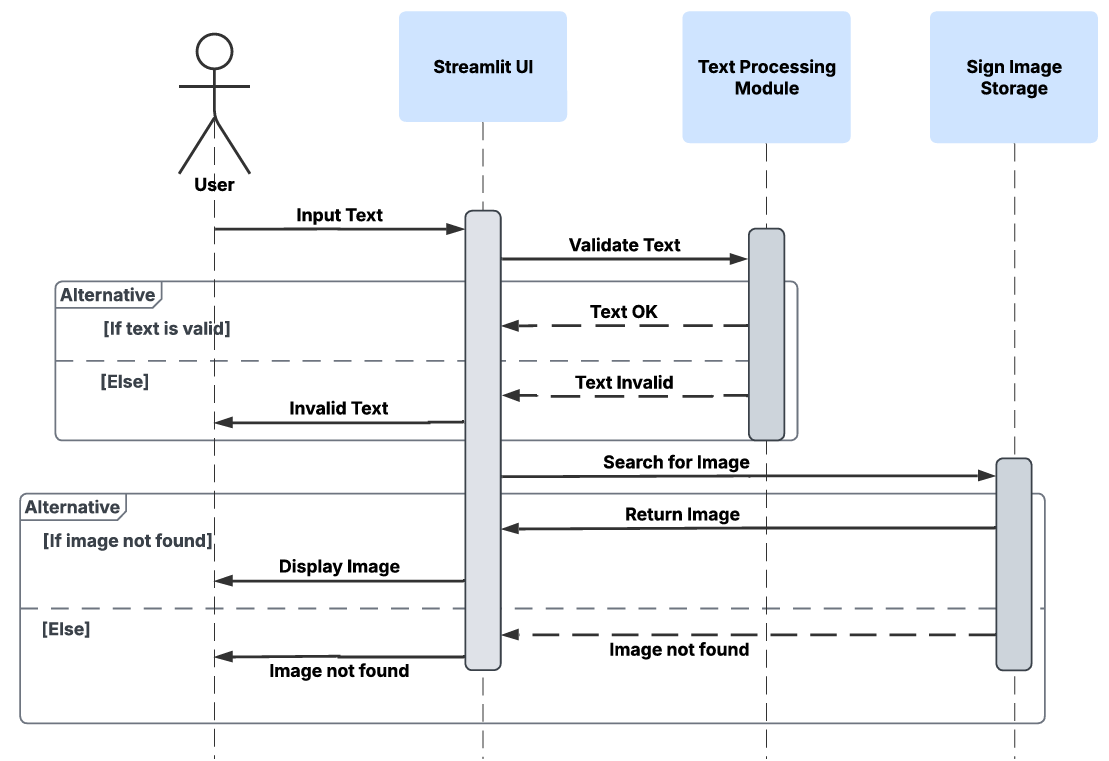
**Fig 5.3** Component Diagram

* + 1. **Sequence Diagram**

A **sequence diagram** is a behavioral diagram that shows the sequence of interactions between objects or components in a system over time. In this context, the sequence diagram illustrates the flow of interactions between the **User** and the **Text Processing Module** in the Streamlit UI application.

**Sequence of Interactions:**

* **User Input**:
  + The user inputs text into the Streamlit UI. This text is intended to be converted into a sign language image.
* **Text Validation**:
  + The **Text Processing Module** validates the input text to ensure it is valid and can be processed.
  + **If the text is valid**, the module proceeds to search for the corresponding sign language image.
  + **If the text is invalid**, the module notifies the user that the input is invalid.
* **Search for Image**:
  + If the text is valid, the **Text Processing Module** searches for the corresponding sign language image in the **Sign Image Storage**.
* **Return Image**:
  + If the image is found, it is returned to the user and displayed in the Streamlit UI.
  + **If the image is not found**, the user is notified that the image could not be retrieved.
* **Display Image**:
  + The valid image is displayed to the user in the Streamlit UI.



**Fig 5.4** Sequence Diagram

* + 1. **Activity Diagram**

An **activity diagram** is a behavioral diagram that represents the flow of activities or processes in a system. It shows the sequence of actions, decision points, and the flow of control from one activity to another. In this context, the activity diagram illustrates the processes involved in recognizing sign gestures and converting speech into sign language animations.

**Key Components in the Activity Diagram:**

* **Perform Sign Recognition**:
  + This process involves capturing and recognizing sign gestures, converting them into text, and then using text-to-speech to produce audible output.
* **Speech-To-Sign**:
  + This process involves capturing speech, converting it into text, and then displaying the corresponding sign language animation.

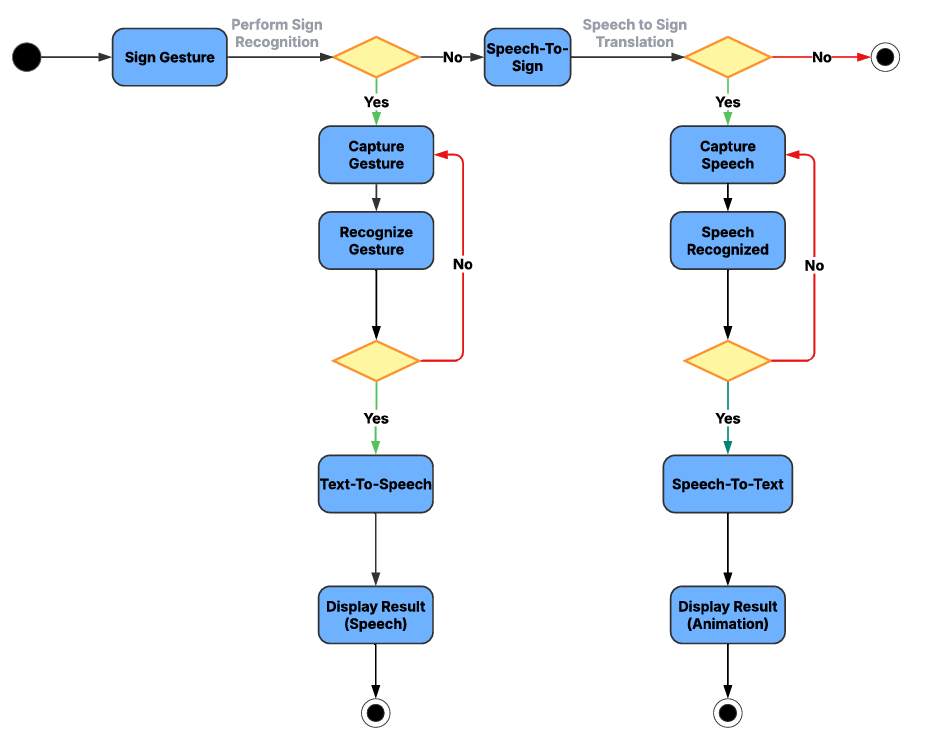
**Flow of Activities:**

**Perform Sign Recognition:**

* **Capture Gesture**: The system captures the user's sign gesture using a camera or other input device.
* **Recognize Gesture**: The system attempts to recognize the captured gesture.
  + **If the gesture is recognized (Yes)**:
    - The system converts the recognized gesture into text.
    - The text is then converted into speech using **Text-To-Speech**.
    - The result (speech) is displayed to the user.
  + **If the gesture is not recognized (No)**:
    - The process ends, and no further action is taken.

**Speech-To-Sign:**

* **Capture Speech**: The system captures the user's speech using a microphone or other input device.
* **Speech Recognized**: The system attempts to recognize the captured speech.
  + **If the speech is recognized (Yes)**:
    - The system converts the recognized speech into text.
    - The text is then converted into a sign language animation.
    - The result (animation) is displayed to the user.
  + **If the speech is not recognized (No)**:
    - The process ends, and no further action is taken.

****

**Fig 5.5** Activity Diagram

**CHAPTER 6**

**METHODOLOGIES**

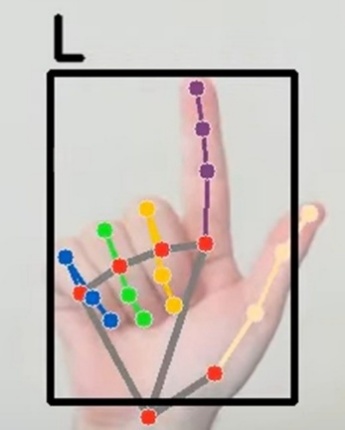
# 6.1 Components

# Bounding Boxes:

Bounding box annotation is a technique used to identify and classify hand gestures in images and video frames. Each bounding box defines attributes after training, including:

* **Hand Gesture Detection**
* **Accuracy Score**
* **Class Prediction**

The bounding boxes assist in isolating the hand in real-time video processing, ensuring accurate feature extraction.

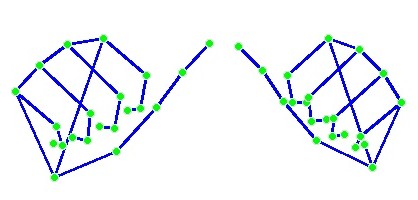


**Fig 6.1** Bounding Box

#### ****Hand Gesture Detection and Segmentation:****

#### Hand gesture segmentation is achieved using MediaPipe Hands, a deep learning-based approach for detecting 21 hand landmarks. The process involves:

* **Preprocessing**: Converting frames to grayscale and resizing for consistency.
* **Feature Extraction**: Capturing **(x, y) coordinates** of hand landmarks.
* **Data Augmentation**: Adding minor noise to improve model robustness.



**Fig 6.2** Gesture Detection and Hand Segmentation

#### ****Multi-Layer Perceptron (MLP) Classifier:****

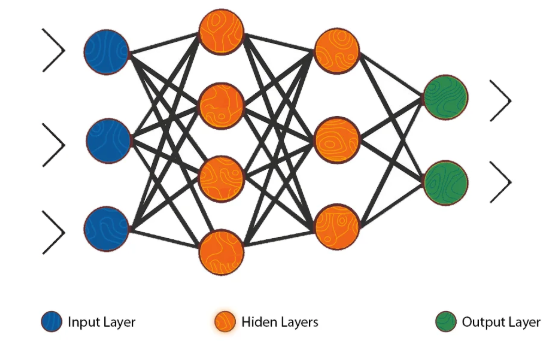
#### The MLP classifier is used for hand gesture classification. It consists of:

* **Input Layer**: Hand landmark features.
* **Hidden Layers**: Two layers with 128 and 64 neurons, using **ReLU activation**.
* **Output Layer**: Softmax activation for class prediction.

Two versions of the classifier are used:

* **Single-Hand Model (42 Features)**: Recognizes one-hand gestures.
* **Dual-Hand Model (84 Features)**: Recognizes two-hand gestures.

Both models are trained with **80% training and 20% testing data**, achieving high accuracy for **sign language recognition**.



**Figure 6.3** MLP Classifier

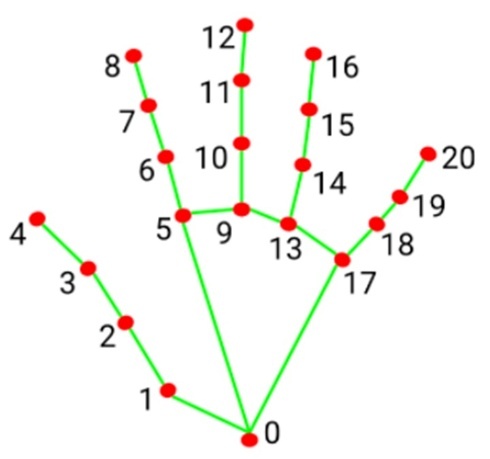
## 6.2 Technologies

## 

* **Python**

Python is the primary programming language used for developing the entire project. It provides extensive libraries for machine learning, image processing, and speech recognition. Its simplicity and powerful ecosystem make it an ideal choice for implementing AI models, integrating APIs, and building an interactive web-based application using Streamlit. It integrates well with libraries such as **OpenCV, MediaPipe, TensorFlow, and Streamlit** for implementing hand gesture recognition.

* **Streamlit**  
   Streamlit is used to create the web interface for the project, allowing users to interact with sign language recognition, speech-to-text, and text-to-sign functionalities. It simplifies UI development by enabling rapid deployment with minimal code. Features like real-time updates, image display, and audio playback enhance the overall user experience.
* **MediaPipe**  
   MediaPipe is utilized for real-time hand tracking and landmark extraction. It detects hand movements, extracts key points from the fingers, and provides structured data for classification. This eliminates the need for manual image feature extraction and allows efficient sign language recognition in real-time applications.



**Fig 6.4** Hand Landmarks

* **Speech Recognition (Google API & pyttsx3)**

The Speech Recognition library enables speech-to-text conversion for the speech-to-sign feature. It captures audio input, processes it using Google's Speech-to-Text API, and extracts relevant words. The recognized text is then mapped to corresponding sign language gestures, allowing for a seamless voice-controlled sign translation system. gTTS (Google Text-to-Speech).

* **gTTS(GoogleText-to-Speech)**  
   Google Text-to-Speech (gTTS) converts recognized text into speech for the sign-to-speech feature. It generates natural-sounding audio output, enabling users to listen to the translated text. This feature enhances communication accessibility by allowing sign language users to convert their signs into spoken language effortlessly.
* **OpenCV**  
   OpenCV (Open Source Computer Vision Library) is used for image processing tasks, such as capturing frames from a webcam, detecting hands, and preprocessing images before feeding them into the model. It ensures smooth real-time hand tracking and gesture recognition by optimizing frame analysis and feature extraction.
* **NumPy&Pandas**  
   NumPy and Pandas are used for handling datasets and preprocessing data before training the machine learning model. NumPy enables efficient numerical operations on extracted features, while Pandas helps structure and manage datasets, making it easier to clean, analyze, and prepare data for model training.
* **Matplotlib&Seaborn**  
   These libraries are used for visualizing the dataset and analyzing model performance. Matplotlib creates plots to display training accuracy, loss, and dataset distributions, while Seaborn helps generate heatmaps and correlation matrices, allowing a better understanding of the model’s behavior and improvement areas.
* **Pickle**

Pickle is used to save and load trained machine learning models, preventing the need for retraining the model every time the application runs. By serializing the trained model, the application can quickly load it and perform sign language recognition without additional computational overhead.

**CHAPTER 7**

**IMPLEMENTATION**

## 7.1 Requirements

### 7.1.1 Overall Description

Defining system requirements is crucial for ensuring our sign language recognition system meets user needs and functions effectively. These requirements guide development, streamline decision-making, and help manage scope, reducing project risks. The system must support real-time text-to-sign, speech-to-sign, and sign-to-text conversion with high accuracy. It should integrate computer vision (MediaPipe Hands), deep learning (MLP classifiers), and Google TTS for seamless translation. Performance, usability, and security are key, ensuring low latency, an intuitive Streamlit interface, and secure data handling. Clear requirements also enhance team collaboration, minimize errors, and provide a reference for validating that the final product meets all specified criteria.

### Software Requirements

* **Python (3.7 or above)** – Programming language used for implementation
* **Operating system:** Windows 10/11, Linux (Ubuntu), or macOS
* **Dependencies:** OpenCV, MediaPipe, scikit-learn, **Google Text-to-Speech (gTTS), Streamlit, SpeechRecognition API, pickle, Matplotlib, Seaborn, NumPy & Pandas**
* **Web Browser** (Google Chrome, Mozilla Firefox, Edge)- Required for running Streamlit application.

### Hardware Requirements

## Laptop/PC

## Processor: Intel i5/i7 or higher

## RAM: 8GB or higher

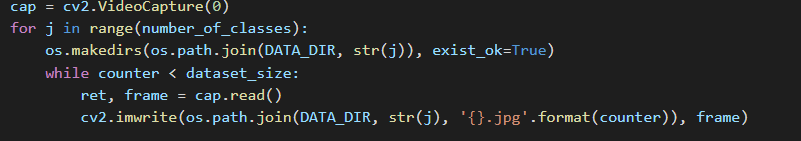
## Camera: Built-in or external webcam for real-time sign detection.

## Code Snippets

The implementation involves multiple scripts that handle different aspects of the project. Below are the key snippets demonstrating major functionalities:

**Data Collection**

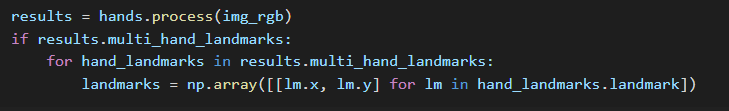
The script collect\_imgs2.py captures images from the webcam, organizing them into class-labeled directories. Each class represents a unique ISL gesture.



**Fig: 7.1** Data Collection

**Dataset Preprocessing**

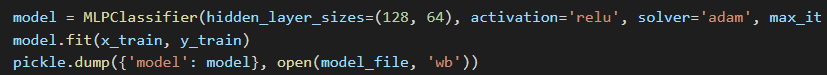
The create\_dataset2.py script extracts **21 hand landmarks** using MediaPipe and normalizes them before saving the dataset.



**Fig: 7.2** Dataset Processing

**Model Training**

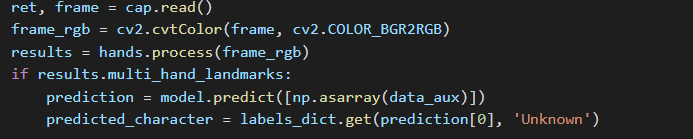
The train\_classifier2.py script trains an MLP classifier on the extracted hand landmarks.



**Fig: 7.3** Model Training

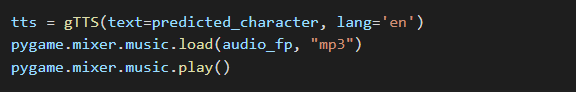
**Real-time Sign Recognition**

The inference\_classifier.py script detects hand landmarks in real-time and classifies the sign.



**Fig: 7.4** Sign Recognition

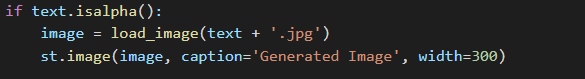
**Detected Sign-to-Speech Conversion**

The 1\_🅰️ Sign Alphabet Recognition.py and 2\_🔢 Sign Number Recognition.py scripts recognize ISL gestures and convert them into speech using gTTS.

**Fig: 7.5** Real Time Sign-to-Speech

**Text-to-Sign Conversion**

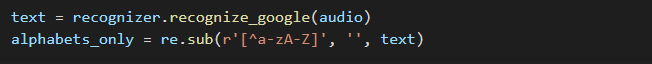
The 4\_📝 Text-to-Sign Translation.py script maps typed text to ISL gesture images.



**Fig: 7.6** Text-to-Sign

**Speech-to-Sign Translation**

The 3\_🎙️ Speech-to-Sign Translation.py script converts spoken words into ISL gestures using Google Speech Recognition.



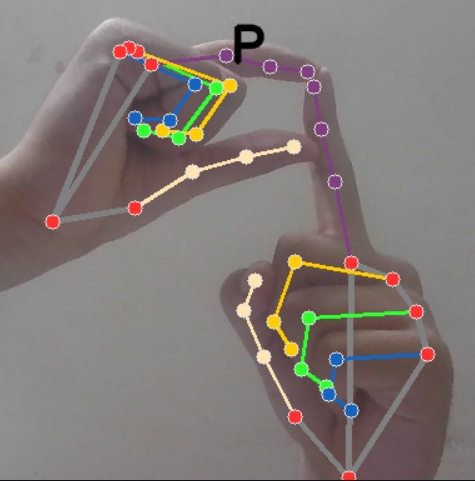
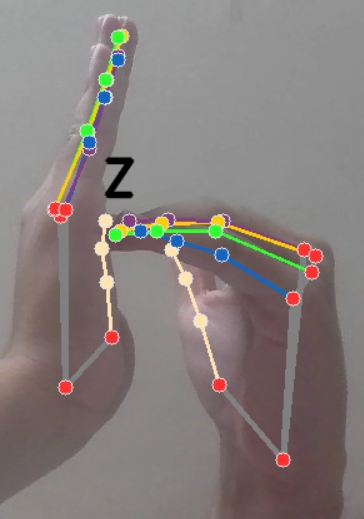
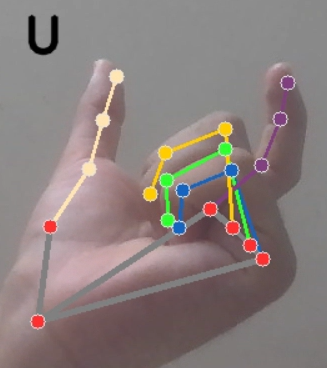
**Fig: 7.7** Speech-to-Sign

## Execution

The developed Indian Sign Language Recognition System successfully detects hand gestures, classifies alphabetic and numeric signs, and converts speech/text to sign language gestures with high accuracy.

**7.3.1 Sign Alphabets Detection-To-Speech**

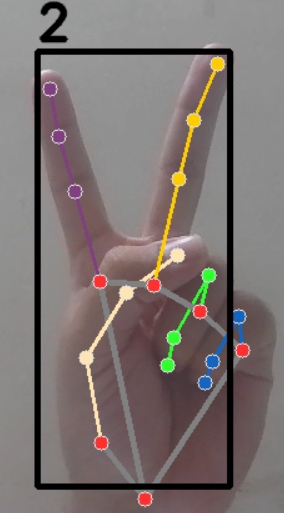
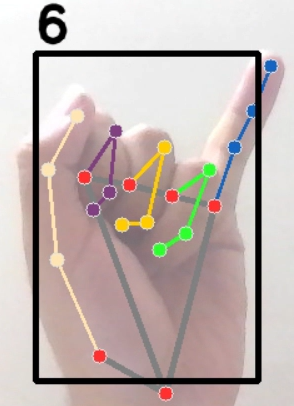
This feature enables the system to recognize individual hand gestures corresponding to letters in the sign language alphabet. Utilizing computer vision and machine learning algorithms, the system captures and interprets these gestures in real-time. Once a gesture is identified, it is converted into the corresponding letter, and as sequences of letters form words, the system employs text-to-speech technology to vocalize the constructed words, facilitating communication from sign language users to those who rely on auditory speech.

**Fig: 7.8** Detected Alphabet P **Fig: 7.9** Detected Alphabet Z **Fig: 7.10** DetectedAlphabet U

* + 1. **Sign Number Detection-To-Speech**

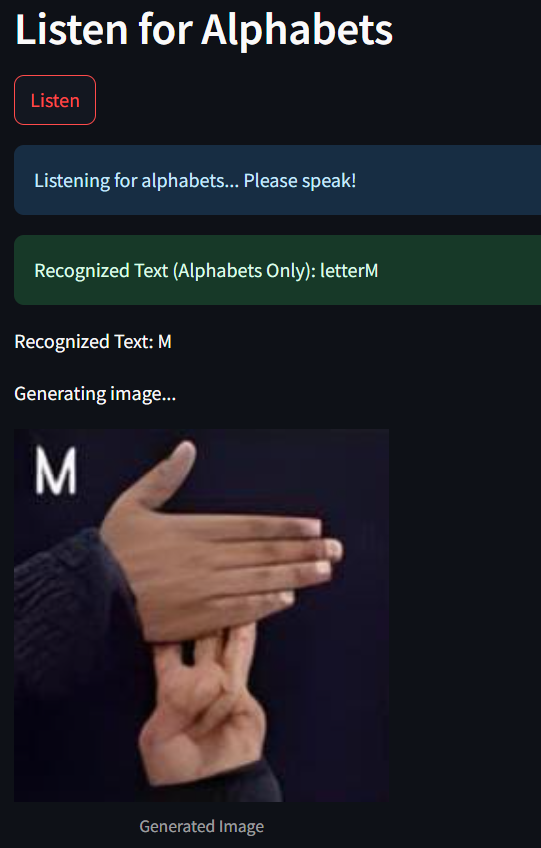
Similar to the alphabet detection, this functionality focuses on recognizing hand gestures that represent numerical values in sign language. The system processes these numerical gestures, converting them into their respective digits. Subsequently, the recognized numbers are articulated through speech synthesis, allowing sign language users to convey numerical information audibly to others.

**Fig 7.11** Detected Numbers 2 and 6

* + 1. **Speech-To-Sign**

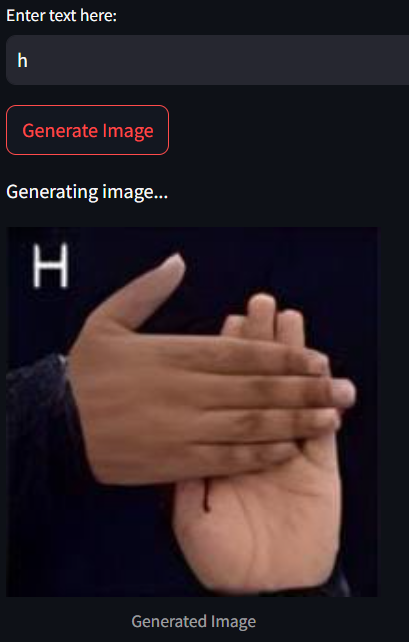
This component allows individuals who use spoken language to communicate effectively with sign language users. The system captures spoken words via a microphone and utilizes speech recognition technology to transcribe the audio into text. The transcribed text is then translated into corresponding sign language representations, which can be displayed as images or animations, enabling sign language users to understand the conveyed message.



**Fig 7.12** Speech-To-Sign

* + 1. **Text-To-Sign**

In scenarios where direct speech input is not feasible, users can input text manually into the system. The system processes the entered text and translates it into sign language, presenting the signs visually through images or animations. This feature is particularly useful for facilitating communication in environments where silence is required or for individuals who prefer typing over speaking.



**Fig 7.13** Text-To-Sign

**CHAPTER 8**

**CONCLUSION AND FUTURE ENHANCEMENT**

**8.1 Conclusion**

In conclusion, the development of a real-time system that translates sign language gestures into audible speech represents a significant advancement in bridging communication gaps between individuals who use sign language and those who do not. By leveraging computer vision and machine learning techniques, the system accurately recognizes and processes gestures, facilitating seamless and efficient communication.

Our implementation of MediaPipe Hands, OpenCV, and MLP classifiers ensures accurate real-time recognition of ISL gestures, supporting both single-hand and two-hand inputs. The integration of speech-to-sign and text-to-sign translation further enhances accessibility, making sign language communication more inclusive. This project is not just a technological achievement but a step toward digital accessibility and inclusivity in education, healthcare, and daily interactions..

Overall, this project demonstrates the potential of integrating advanced technologies to create practical solutions that address real-world challenges, ultimately contributing to a more inclusive society.

## 8.2 Future Enhancement

## Building upon the current system, several avenues for future enhancement can be explored to improve performance, expand capabilities, and increase accessibility:

## Expansion of Gesture Vocabulary:

## Extend the system to recognize a broader range of gestures i.e., words and sentence recognition for continuous sign language translation, including both static and dynamic signs, to cover more comprehensive aspects of sign language.

* **Incorporation of Facial Expressions and Body Postures:**

Integrate recognition of non-manual features such as facial expressions and body movements, which are integral components of sign language communication.

* **Support for Multiple Sign Languages:**

Adapt the system to recognize and translate various sign languages, accommodating regional variations and promoting global applicability.

* **User-Defined Gesture Addition:**

Introduce a feature that enables users to add and train their own custom sign gestures. This would enable personalization and expand the system’s usability across different communities and specialized vocabularies.

* **Improved Real-Time Processing:**

Enhance the system's processing speed and accuracy to ensure seamless real-time translation, minimizing latency and improving user experience.

* **Continuous Learning Capabilities:**

Incorporate machine learning algorithms that enable the system to learn from user interactions and continuously improve its recognition capabilities over time.

# REFERENCES

1. M. Suresh Anand, A. Kumaresan, and Dr. N. Mohan Kumar, “**An Integrated Two Way ISL (Indian Sign Language) Translation System - A New Approach**,” *International Journal of Engineering Research & Technology (IJERT)*, vol. 2, no. 10, pp. 1-6, 2013.
2. Nair, A. V., & Bindu, V., “ **A Review on Indian Sign Language Recognition**,” *International Journal of Computer Applications*, 73(22), 33-38, 2013.
3. M. Pigou, S. Dieleman, P. Kindermans, and B. Schrauwen, “**Sign Language Recognition Using Convolutional Neural Networks**,” in *Lecture Notes in Computer Science*, vol. 9119, Springer, Cham, 2015, pp. 572–578.
4. Agarwal, S. C., et al., “**Grid-Based Feature Extraction for Hand Gesture Recognition in ISL**,” *International Journal of Machine Learning*, 12(2), 210-220, 2018.
5. K. Shenoy, T. Dastane, V. Rao, and D. Vyavaharkar, “**Real-time Indian Sign Language (ISL) Recognition**,” in *Proc. 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, IEEE, Bengaluru, India, 2018.
6. A. K. Moin, A. Zhou, A. Rahimi et al., “**Sign Language Recognition Using Wearable Electronics: Implementing k-Nearest Neighbors with Dynamic Time Warping and Convolutional Neural Network Algorithms**,” *Sensors*, vol. 20, no. 14, p. 3879, 2020.
7. A. K. Singh, P. S. S. Tiwari, and S. S. Chouhan, “**Indian Sign Language Recognition Using Deep Learning**,” in *Proc. 2021 IEEE 9th Region 10 Humanitarian Technology Conference (R10-HTC)*, 2021.
8. S. Sharma and S. Singh, “**Recognition of Indian Sign Language (ISL) Using Deep Learning Model**,” *Wireless Personal Communications*, vol. 121, pp. 2719–2734, 2021.
9. I. A. Putra, O. D. Nurhayati, and D. Eridani, “**Dynamic Sign Language Recognition Using MediaPipe Library and Modified LSTM Method**,” *International Journal on Advanced Science, Engineering and Information Technology*, vol. 13, no. 2, 2023.
10. V. G. Velmathi and K. Goyal, “**Indian Sign Language Recognition Using Mediapipe Holistic**,” *arXiv preprint,* arXiv:2304.10256, 2023.