Sign Language (ASL) Translation

A Project Report

Submitted in partial fulfilment of the requirements

Of

Industrial Artificial Intelligence with Cloud Computing

By

Kavi Patel (21IT433)
Rushi Patel (21IT434)

Under the Esteemed by

Jay Rathod

ACKNOWLEDGEMENT

.

"We would like to express our sincere gratitude to the Edunet Foundation for organizing the AI Saksham internship program at BVM College. This opportunity has played a vital role in improving our knowledge and skills in AI.

We are profoundly thankful to our mentor, Jay Rathod, for his invaluable guidance, encouragement, and expertise throughout this internship. His teachings and mentorship have been crucial in shaping our understanding of AI concepts and in the successful completion of our project.

Special thanks to the IT department for their unwavering support and for providing essential resources. We are also grateful to the Head of IT, Dr. Keyur Brahmbhatt, for his leadership and encouragement, which has fostered an environment conducive to creativity and innovation.

We extend our gratitude to our Principal, Dr. Indrajit Patel, for his overall support and vision, which have been crucial to our academic and professional growth.

Finally, we would like to thank BVM College for providing us with the necessary infrastructure and support to carry out our project. The conducive learning environment has significantly contributed to our growth and development during this internship.

Together, we have achieved a significant milestone, and we are thankful to each individual and organization mentioned above for their contributions to this project."
Thank you.

Kavi Patel, Rushi Patel

ABSTRACT

.

American Sign Language (ASL) is crucial for individuals with hearing impairments, but interpreting ASL signs can be challenging.

This project aims to develop a real-time ASL sign detection and translation system using Python and computer vision techniques. The system consists of data collection to train a machine learning model and live sign detection to process video frames, identify hands, and classify hand gestures into English alphabets. The implementation uses the OpenCV library for image processing and hand detection, combined with custom-built deep-learning models for sign classification.

The project aims to improve accessibility and inclusivity for individuals with hearing impairments and showcases the potential of computer vision and machine learning technologies in addressing real-world communication challenges.

TABLE OF CONTENTS

LIST OF FIGURES

.

CHAPTER 1

INTRODUCTION

CHAPTER 1

INTRODUCTION

1.1. Problem Statement:

The problem statement centres around the concept of a camera-based sign language recognition system for the deaf, which would transform sign language gestures to text and subsequently text to speech. Our goal is to create a user-friendly and straightforward solution.

Dumb individuals communicate via hand signs, thus normal folks have a hard time understanding what they're saying. As a result, systems that recognise various signs and deliver information to ordinary people are required.

1.2. Problem Definition:

Communication barriers exist for individuals with hearing impairments, particularly when interacting with those who do not understand American Sign Language (ASL). Current technological solutions for ASL translation are often limited by factors such as high cost, lack of real-time processing, or limited accuracy in varying environmental conditions. Therefore, there is a pressing need for a cost-effective, accurate, and real-time ASL detection and translation system that can facilitate seamless communication between ASL users and non-users. The goal of this project is to develop a system that can detect ASL signs through a live camera feed and translate them into corresponding English alphabets in real time, using computer vision and machine learning.

1.3. Expected Outcomes:

The expected outcome of this project is a fully functional, real-time ASL sign detection and translation system that can accurately recognize and translate American Sign Language signs into corresponding English alphabets. The key deliverables and benefits include:

1.3.1. Accurate Sign Recognition: The system will be able to detect and accurately classify a range of ASL signs from live video feed using a trained deep learning model

1.3.2. Real-time Translation: The system will provide instantaneous translation of detected ASL signs into English alphabets, offering immediate visual feedback to users

1.3.3. User-Friendly Interface: A straightforward and intuitive interface will display the recognized English letters in real-time, making it accessible for both ASL users and non-users.

1.3.4. **Robust Performance**: The system will perform effectively under various conditions, including different lighting environments and hand orientations, ensuring reliability and consistency.

1.3.5. **Enhanced Communication:** The system will bridge the communication gap between individuals proficient in ASL and those unfamiliar with the language, promoting inclusivity and better interaction in educational, professional, and social settings

1.3.6. **Cost-Effective Solution:** By utilizing widely available hardware and open-source software, the system will offer an affordable solution compared to traditional ASL interpretation methods.

1.3.7. **Scalability and Extensibility:** The project's framework will be designed to allow for future expansion, including the addition of more ASL signs, words, and phrases, as well as potential integration with other languages and dialects.

CHAPTER 2

PROPOSED METHODOLOGY

CHAPTER 2
PROPOSED METHODOLOGY

2.1. System Design:

2.1.1. Input Module:

Captures live video feed from a camera.
Streams video frames to the subsequent processing modules.

2.1.2. Hand Detection and Tracking Module:

Utilizes OpenCV or Mediapipe for detecting and tracking hands in the video frames.
Extracts and isolates hand regions for further processing.

2.1.3. Data Preprocessing Module:

Normalizes, resizes, and augments hand region images.
Prepares data for input to the classification model.

2.1.4. Feature Extraction Module:

Extracts relevant features from the preprocessed hand images.
May include shape, finger positions, and spatial orientation.

2.1.5. Classification Module:

Utilizes a trained deep learning model (e.g., CNN) to classify hand gestures into ASL signs.
Maps classified signs to corresponding English alphabets.

2.1.6. Translation Module:

Converts classified ASL signs into English alphabets.
Handles real-time translation and updates the user interface.

2.1.7. Output Module:

Displays the recognized English alphabets on the screen.
Provides real-time feedback to the user.

2.2 Modules Used:

2.2.1. Input Module:

Description: Captures live video feed from the camera.

Functionality:
Initialize the camera for capturing video.
Continuously capture frames from the live video feed.
Pass the captured frames to the Hand Detection and Tracking Module for further processing.

2.2.2. Hand Detection and Tracking Module:

Description: Detects and tracks hands in the captured video frames.

Functionality:
Utilize computer vision techniques to detect hands in each frame.
Track the position and movement of hands.
Extract the bounding box coordinates of detected hands for cropping.

2.2.3. Data Preprocessing Module:

Description: Preprocesses the hand region images for feature extraction and classification.

Functionality:
Normalize and resize the detected hand images to a standard size.
Perform data augmentation techniques to increase the variability and size of the dataset.
Ensure the images are suitable for input to the classification model.

2.2.4. Feature Extraction Module:

Description: Extracts relevant features from the preprocessed hand images.

Functionality:
Extract key features such as hand shape, finger positions, and spatial orientation from the images.
Prepare the feature set for input to the classification model.

2.2.5. Classification Module:

Description: Classifies the extracted features into corresponding ASL signs.

Functionality:
Load and utilize a trained deep learning model to classify hand gestures.
Predict the ASL sign based on the input features.

2.2.6. Translation Module:

Description: Translates the classified ASL signs into English alphabets.

Functionality:
Map the output of the classification model to the appropriate English alphabet.
Ensure accurate translation and handle any errors or ambiguities in the classification.

2.2.7. Output Module:

Description: Displays the recognized English alphabets in real-time.

Functionality:
Create a user-friendly interface to show the live video feed and recognized letters.
Provide immediate feedback to the user by overlaying the recognized alphabets on the video feed.

2.3 Data Flow Diagram

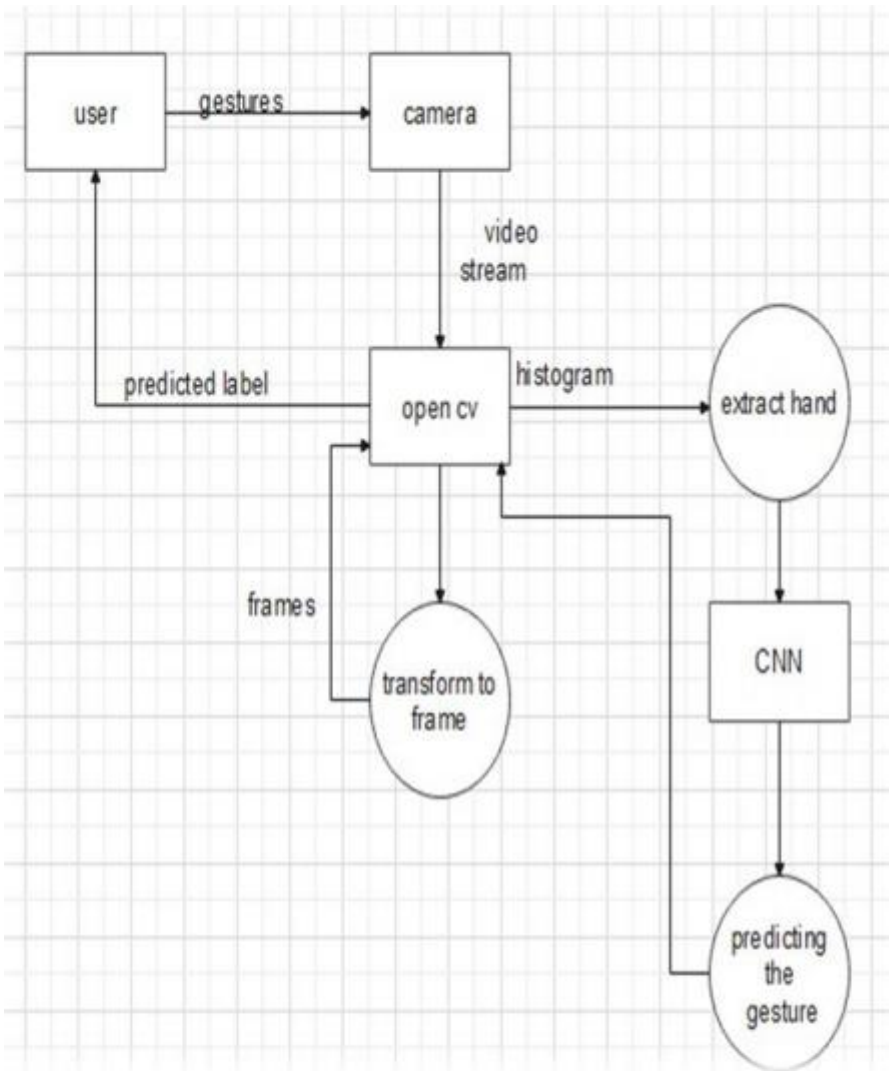

Figure-1: DFD Diagram

This Data Flow Diagram (DFD) represents the process of gesture recognition using computer vision and a Convolutional Neural Network (CNN).

Here's a detailed explanation of the components and the flow:

2.3.1. **User**:

- **Gestures**: The user performs various gestures that are intended to be recognized by the system.

2.3.2. **Camera**:

- **Video Stream**: The camera captures the user's gestures in the form of a continuous video stream.

2.3.3. **OpenCV**:

- **Video Stream**: OpenCV, an open-source computer vision and machine learning software library, processes the video stream from the camera.
- **Histogram**: OpenCV generates a histogram, which helps in segmenting and identifying the hand region from the video stream.
- **Frames**: The video stream is converted into individual frames for further processing.

2.3.4. **Extract Hand**:

- This process uses the histogram generated by OpenCV to isolate and extract the hand region from each frame. This step is crucial for focusing the gesture recognition on the relevant part of the image.

2.3.5. **CNN (Convolutional Neural Network)**:

- The extracted hand images are fed into a CNN, a type of deep learning model particularly well-suited for image recognition tasks.
- **Predicting the Gesture**: The CNN processes the hand images and predicts the gesture being performed by the user.

2.3.6. **Predicting the Gesture**:

- The output from the CNN is the predicted gesture label, which is sent back to the user as feedback.

2.3.7. **Transform to Frame**:

- This step involves converting the processed frames back into a format suitable for display or further analysis, if necessary. This might include resizing, normalization, or other preprocessing steps required by the CNN.

Flow Summary:

- The user performs gestures that are captured by the camera.
- The video stream is processed by OpenCV to generate histograms and frames.
- The hand is extracted from each frame using the histogram information.
- The extracted hand images are fed into a CNN to predict the gesture.
- The predicted gesture label is provided as feedback to the user.

2.4 Requirement Specification

2.4.1 Hardware Requirements:

1. **Camera:** High-resolution camera capable of capturing clear video footage of the driver's face under various lighting conditions.

2. **Processor:** Powerful CPU/GPU to handle real-time video processing and machine learning computations.

3. **Memory:** Sufficient RAM to process video frames and run complex algorithms without latency.

4. **Storage:** Adequate storage for saving video data and model parameters.

2.4.2 Software Requirements:

1. **Operating System:** Compatible with popular operating systems like Windows, Linux, or macOS.

2. **Programming Languages:** Python is suitable for machine learning and computer vision tasks.

3. **Libraries and Frameworks:**
    - OpenCV
    - MediaPipe
    - NumPy
    - TensorFlow/Keras
    - Matplotlib

4. **Development Environment:** IDEs like PyCharm, Visual Studio Code, or Jupyter Notebook for development and testing.

5. **Data Management:** Tools for managing and processing datasets, such as pandas and NumPy.

6. **Model Files:** Pre-trained deep learning model (e.g., keras_model.h5) and Labels file (e.g., labels.txt) containing the ASL signs and corresponding English alphabets

CHAPTER 3
IMPLEMENTATION AND RESULTS

CHAPTER 3
IMPLEMENTATION AND RESULTS

4.1 Implementation of Hand Detection

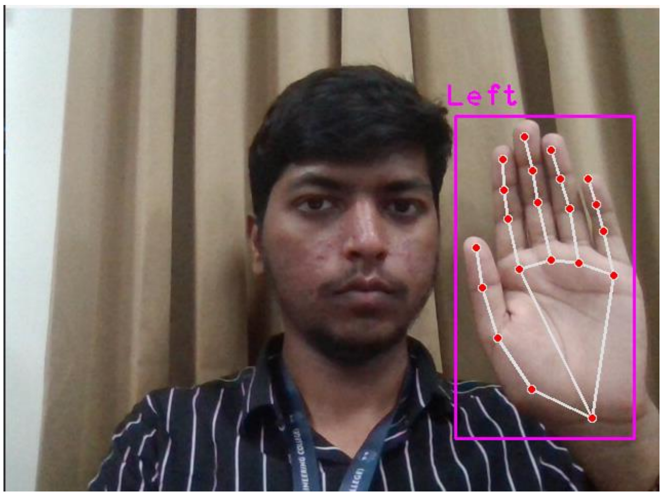

Figure-2: Hand Detection(Left)

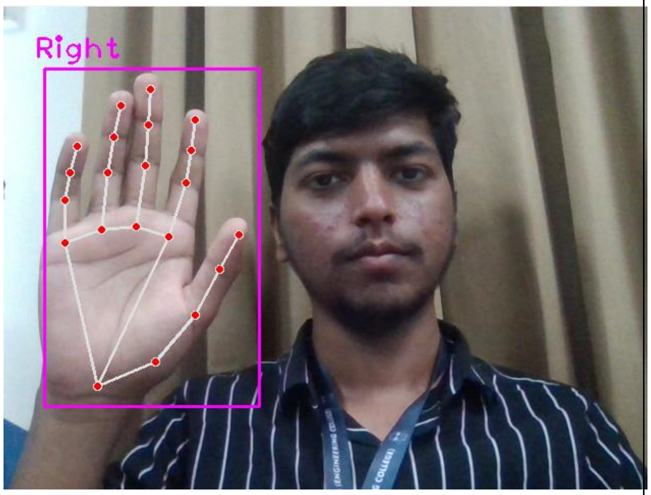

Figure-3: Hand Detection(Right)

4.2 Implementation of ASL Detection

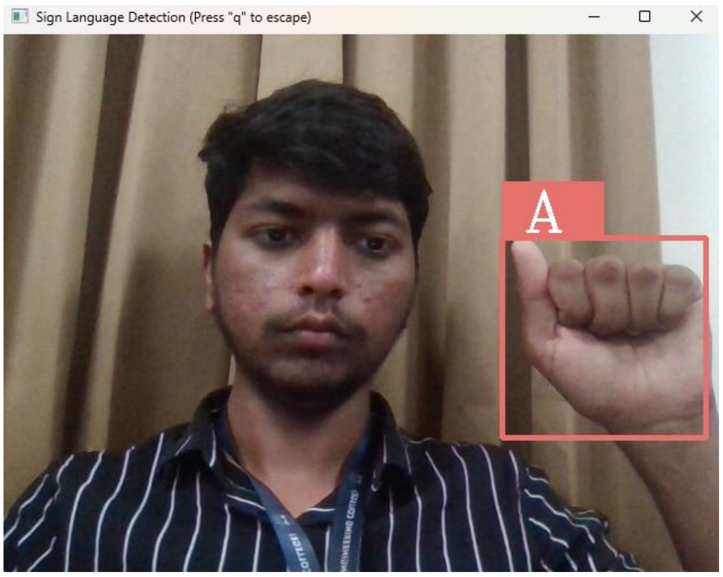

Figure-4: ASL Detection(A)

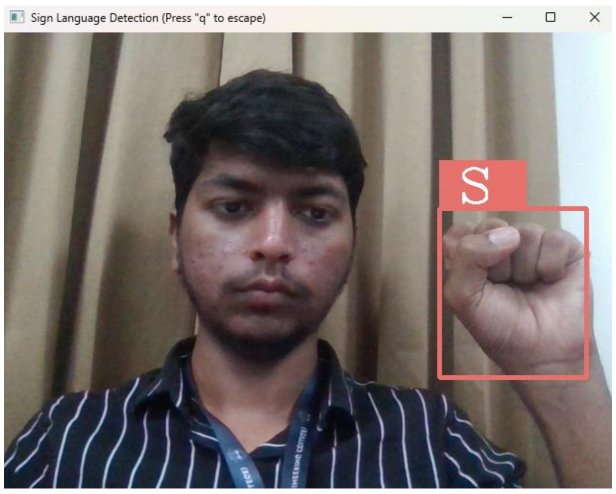

Figure-5: ASL Detection(S)

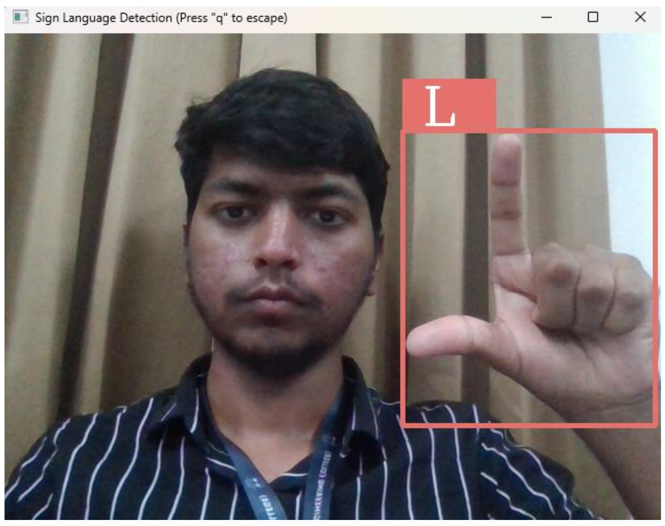

Figure-6: ASL Detection(L)

CHAPTER 4

CONCLUSION

CHAPTER 4

CONCLUSION

4.1 Advantages:

4.1.1. Enhanced Communication Accessibility:

Bridges the communication gap between ASL users and non-users, facilitating smoother interactions in various settings such as education, workplaces, and public spaces.

4.1.2. Real-Time Translation:

Provides immediate feedback by translating ASL signs into English alphabets in real-time, enabling instant communication without delays.

4.1.3. Scalability:

Designed to be easily extendable, allowing for the addition of more ASL signs, words, and phrases in the future. The system can also potentially support other sign languages.

4.1.4. Educational Tool:

Can serve as an educational resource for learning ASL, helping non-ASL users to become familiar with and learn the language.

4.1.5. Reduced Dependence on Human Interpreters:

Provides an alternative to human interpreters, which can be scarce, expensive, or unavailable in certain situations, thereby ensuring continuous and independent communication support.

4.2 Scope:

4.2.1. Expanded ASL Vocabulary:

Extend the system to recognize and translate a wider range of ASL signs, including common phrases, expressions, and sentences, to make communication more comprehensive and meaningful.

4.2.2. Multi-Language Support:

Extend the system to support multiple sign languages beyond ASL, such as British Sign Language (BSL), French Sign Language (LSF), or International Sign, to cater to a more diverse user base globally.

4.2.3. Real-Time Feedback and Correction:

Integrate feedback mechanisms to provide real-time corrections or suggestions to users based on their ASL signing, helping them improve their sign language proficiency over time.

4.2.4. **Mobile Application**:

Develop a mobile application version of the system that can run on smartphones or tablets, enabling users to carry the ASL translation tool with them wherever they go for on-the-go communication support.

4.2.5. **Social Integration**:

Integrate the system with social media platforms or communication apps to enable ASL users to communicate seamlessly with friends, family, and colleagues across different digital channels.

GitHub Link:

https://github.com/Kavipatel18/Sign-Language-Detection

Video Link:

Sign_language_translation_demo.mp4

REFERENCES

Link:

- https://ijrpr.com/uploads/V2ISSUE9/IJRPR1329.pdf
- https://www.sciencedirect.com/science/article/pii/S1877050921000442
- https://www.researchgate.net/publication/262187093_Sign_language_recognition_State_of_the_art
- https://www.nature.com/articles/s41598-023-43852-x
- https://chatgpt.com

APPENDIX

Appendix A: Sample Data

This appendix includes sample data used to train and test the ASL sign detection and translation system.

The data may consist of:
Images:
A collection of images containing various ASL signs captured from different angles, hand orientations, and lighting conditions.

Videos:
Short video clips showcasing real-world scenarios of individuals performing ASL signs with examples of different signing speeds and styles.

Appendix B: System Architecture Diagram

This appendix provides a visual representation of the system architecture, illustrating the interaction between different components.

It can be a block diagram showcasing:
Input Module: Captures live video feed from the camera.

Hand Detection and Tracking Module: Detects and tracks hands in the video frames.

Data Preprocessing Module: Preprocesses hand region images for feature extraction.

Feature Extraction Module: Extracts relevant features from the preprocessed images.

Classification Module: Classifies the extracted features into corresponding ASL signs.

Translation Module: Translates the classified ASL signs into English alphabets.

Output Module: Displays the recognized English alphabets in real-time.

Appendix C: Performance Metrics

This appendix details the performance metrics used to evaluate the ASL sign detection and translation system's effectiveness. Examples include:

Accuracy: The percentage of correctly classified ASL signs.

Precision: The ratio of true positives to the total number of positive predictions.

Recall: The ratio of true positives to the total number of actual positive cases.

False Positive Rate: The percentage of incorrectly classified ASL signs.

False Negative Rate: The percentage of missed detections of ASL signs.

Appendix D: Code Snippets

This appendix includes relevant code snippets showcasing the implementation of key functionalities, such as:
Hand detection and tracking using OpenCV or Mediapipe.
Feature extraction from hand region images.
Deep learning model training and prediction using TensorFlow or Keras.