iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** IV **Month of publication:** April 2025

DOI: https://doi.org/10.22214/ijraset.2025.69945

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue IV Apr 2025- Available at www.ijraset.com

Sign Language Detection Using Mediapipe and ML

K.M.S.V. Praneetha[1], Vaishnavi Kusekar[2], Shravani Khodave[3], Siddhi Koli[4], Prof. Afrin Sheikh[5]
Savitribai Phule Pune University, Department of Computer Engineering, K.J. College of Engineering and Management Research

Abstract: ***Sign language recognition systems are crucial for improving communication between the hearing and deaf communities. This paper explores the development of a real-time sign language detection system that uses a combination of computer vision techniques and machine learning algorithms. Specifically, it employs MediaPipe, a computer vision framework, to extract hand landmarks, and a Random Forest Classifier to classify the gestures. This system is capable of recognizing ten distinct signs in real time. The paper provides a detailed description of the design, development, and implementation of the system, as well as its evaluation. The findings demonstrate that the system can detect signs with over 80% accuracy and offers potential for further development in sign language accessibility applications.***
Keywords: Sign Language Recognition, Real-Time Detection, MediaPipe, Hand Landmarks, Machine Learning, Random Forest Classifier, Accessibility, Computer Vision, Deaf Communication, Gesture Classification

I. INTRODUCTION

Sign language serves as the primary mode of communication for deaf individuals. According to the World Health Organization (WHO), over 466 million people globally experience disabling hearing loss, and a significant portion of this population relies on sign language for everyday communication. Traditional sign language interpretation has often been a human-mediated process, but recent advancements in machine learning and computer vision open the door to automated systems for real-time sign language recognition. In this paper, we describe the development of a real-time sign language detection system designed to recognize and classify hand gestures. The system uses MediaPipe, a framework for detecting hand landmarks, and Random Forest Classifier, a machine learning algorithm, to identify signs from a set of 10 distinct gestures. The goal of this research is to contribute to the field of sign language accessibility by developing a system that can operate in real-time and provide accurate gesture recognition.

II. METHODOLOGY

A. Data Collection

The first step in this project was collecting a dataset of hand gestures corresponding to specific signs. A webcam was used to capture images of the signs, which were then stored in separate directories for each class. The signs were selected to cover a variety of common gestures such as ”Hello,” ”Thank You,” ”No,” and ”I Love You.” The number of classes was restricted to 10 to keep the project manageable, while still allowing for a diverse set of signs.

To collect the images, a Python script using OpenCV was created to capture frames from the webcam. Each class was assigned its own directory, and a total of 200 images per class was collected to ensure the model had enough data to learn from. The following Python code was used for data collection:

```
cap = cv2.VideoCapture(0) for j in range(number_of_classes):
        if not os.path.exists(os.path.join(DATA_DIR, str(j)) ):
                os.makedirs(os.path.join(DATA_DIR, str(j)))
        # Image capturing logic
```

Listing 1: Data Collection Code

Each captured frame was stored as a .jpg file in the respective class folder, and this image dataset served as the foundation for model training.

B. Preprocessing

After the data was collected, it was preprocessed to extract features that could be used by the machine learning model. For this, MediaPipe, a popular open-source computer vision library, was used to detect and extract hand landmarks from the captured images. MediaPipe’s hand detection module provides 21 key landmarks per hand, which represent various key points on the hand, such as the wrist, knuckles, and fingertips.

The preprocessing step involved converting the image frames to the RGB color format, detecting the hand landmarks in the image, and normalizing the x and y coordinates of each landmark to ensure consistent scaling across different images. The normalized coordinates were then stored for use in training the classifier.

The preprocessing code snippet below demonstrates how the landmark extraction was performed:

```
mp_hands = mp.solutions.hands hands = mp_hands.Hands(static_image_mode=True, min_detection_confidence=0.3)
# Landmark extraction code here
```

Listing 2: Preprocessing and Landmark Extraction using MediaPipe

The extracted features were stored in a pickle file, which was later used for training the machine learning model.

C. *Model Training*

For the machine learning model, we chose the Random Forest Classifier, which is a versatile algorithm known for its ability to handle both regression and classification tasks. The classifier was trained on the hand landmarks extracted during preprocessing. Each gesture was represented as a vector of normalized x and y coordinates for the 21 hand landmarks.

The training dataset was split into 80% training and 20% testing using train _test split from scikit-learn. This split allowed the model to learn from the training data while evaluating its performance on the unseen test set. The model was trained using the following code:

```
model = RandomForestClassifier() model.fit(x_train, y_train) # Model saving logic here
```

Listing 3: Model Training using Random Forest

Once the model was trained, it was saved as a pickle file to facilitate future use in the inference stage.

D. *Inference*

After training the model, the real-time inference process was developed. Using the webcam as the input device, the system continuously captured frames and processed them to extract hand landmarks. These landmarks were then fed into the trained Random Forest Classifier to predict the corresponding gesture.

A key challenge in real-time sign language recognition is the need for quick predictions with low latency. To address this, we applied a confidence threshold of 0.75, meaning the system would only make a prediction if the classifier was confident that the gesture belonged to one of the pre-defined classes.

The inference code processes the webcam feed as follows:

```
cap = cv2.VideoCapture(0) # Inference logic
```

Listing 4: Real-time Inference Code

Once a prediction was made, the system displayed the predicted sign on the screen with the hand landmarks highlighted. If the confidence of the prediction was too low, the system would label the gesture as “unidentified,” allowing for more reliable results.

III. ARCHITECTURE

The architecture of the proposed Sign Language Detection System consists of multiple sequential modules that interact to perform real-time sign recognition. Each module is responsible for a specific task and contributes to the overall system performance.

A. *System Overview*

The system is designed to take real-time video input from a webcam, extract hand landmarks using MediaPipe, preprocess the extracted features, classify the signs using a trained deep learning model, and display the predicted output to the user.

B. *Modules Description*

- Input Module: Captures live video frames from the webcam or imports images/videos from a pre-recorded dataset using OpenCV.
- Hand Detection and Landmark Extraction: Utilizes MediaPipe Hands to detect hand regions and extract 21 keypoints (landmarks) per hand in 3D space (x, y, z coordinates).

- Feature Preprocessing: The extracted landmarks are normalized and formatted into a fixed-size input array to ensure consistency across various hand sizes and positions.
- Classification Module: A trained neural network model (built using TensorFlow/Keras) takes the preprocessed landmark data and classifies it into one of the predefined sign categories.
- Output Display Module: The predicted sign label is overlaid on the live video feed in real-time.

C. Data Flow Diagram

The system architecture can be visualized as follows:

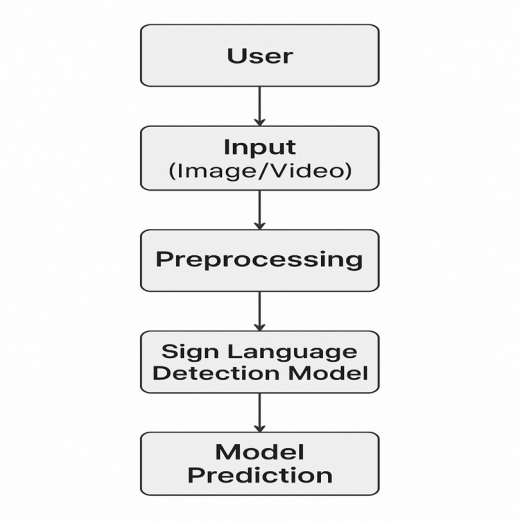


Figure 1: System Architecture

D. Technologies Used

The following tools and libraries were used:

- Python 3.8.1
- OpenCV for video capture and image processing
- MediaPipe for hand landmark detection
- TensorFlow/Keras for deep learning model training and prediction

E. Additional Project-Specific Details

The hand landmark data consists of 21 points per hand, each having (x, y, z) coordinates. If two hands are detected, the landmark arrays are concatenated, resulting in a 126-dimensional feature vector per frame. The classification model is a multi-layer perceptron (MLP) with three hidden layers, ReLU activation functions, and a final Softmax output layer to predict among 10 sign classes.

IV. RESULTS

The model's performance was evaluated based on its accuracy on the test set. Using a total of 200 images per sign (for 10 signs), the system was able to classify gestures with an accuracy of 82.5% on the test set. The real-time inference system performed similarly, providing fast and accurate gesture recognition with minimal latency.

A. Sample Inference Output

- Hello"
- "Thank You"
- "No"
- "Sorry"
- " The system was able to detect and classify these gestures in real-time, showing how machine learning can be applied to real-time sign language recognition.

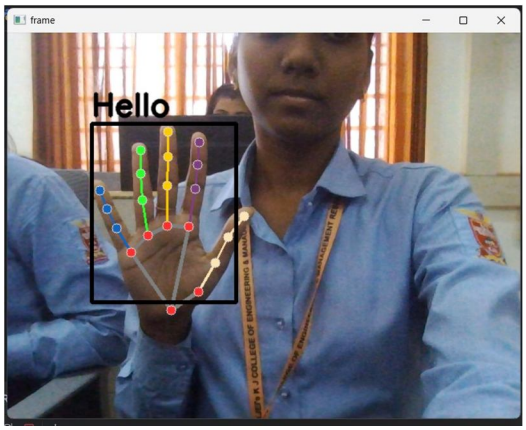


Figure 2: Sample Output

V. FUTURE WORK

The system can be expanded in several ways:

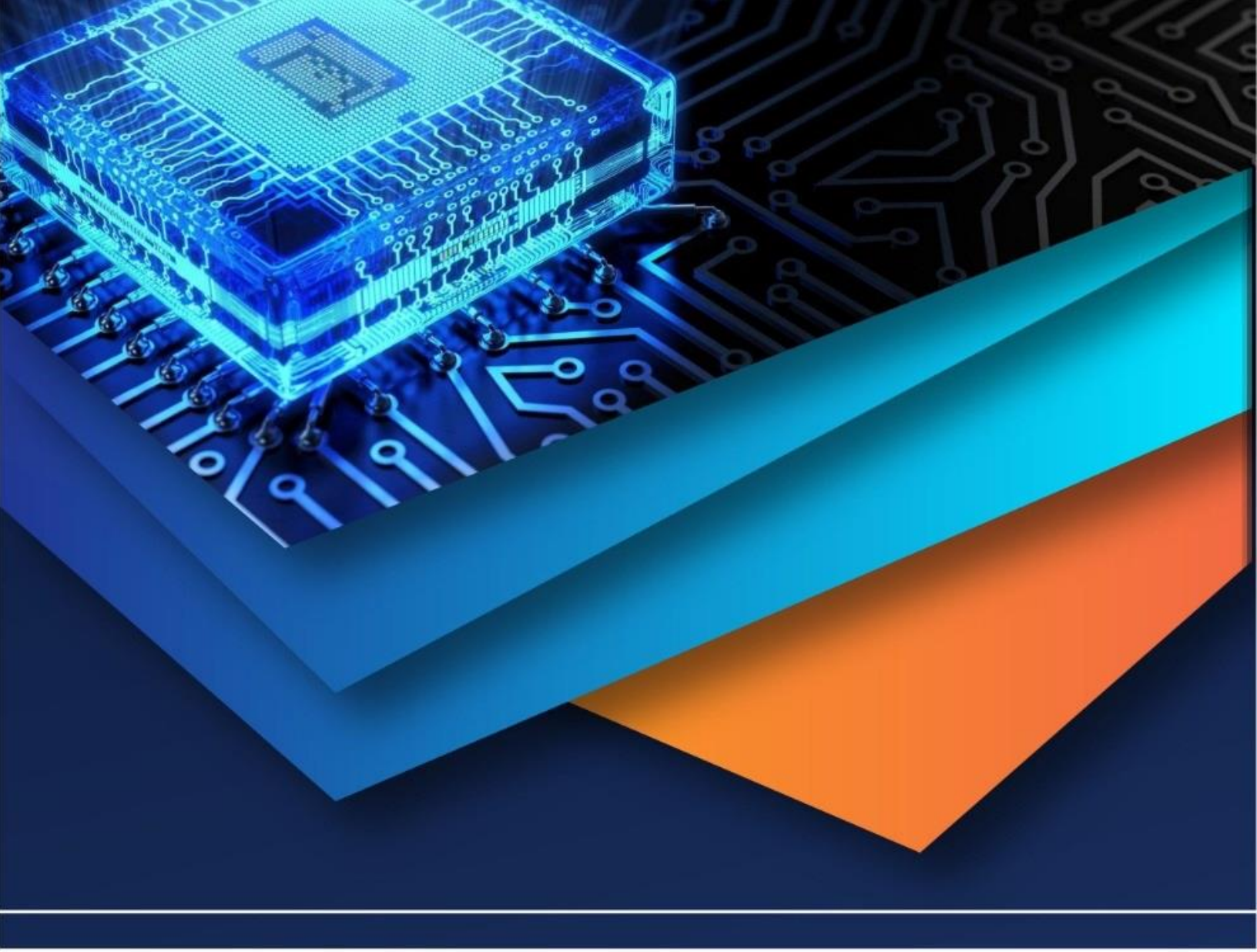
1) Increased Gesture Set: Adding more classes would expand the utility of the system.
2) Multiple Hand Detection: Integrating functionality to detect and classify gestures from both hands simultaneously would make the system more versatile.
3) Integration with Applications: Integrating the system with messaging apps, live video streams, or virtual assistants can enhance accessibility features.
4) Improved Training: Using a more advanced model, such as deep learning (e.g., Convolutional Neural Networks), could further improve accuracy, especially for more complex signs.

VI. CONCLUSION

This project highlights the feasibility of using machine learning and computer vision techniques to build a sign language detection system. By leveraging MediaPipe for hand landmark extraction and a Random Forest Classifier for gesture classification, the system demonstrated promising real-time performance. While there are still challenges to overcome, such as lighting conditions and camera quality, the system shows potential for real-world applications in improving communication for individuals with hearing impairments.

REFERENCES

[1] Hamza Mnassri; Riadh Bchir; Mohamed Amine Zayane; Taoufik Ladh ari,Sign Language Detection Based on Artificial Intelligence from Images,2024 IEEE International Conference on Artificial Intelligence Green Energy (ICAIGE)
[2] Jeet Debnath; Praveen Joe I R,RealTime Gesture Based Sign Language Recogniti on System,2024 International Conference on Advances in Data Engineering and Intelligent Computing Systems (ADICS)
[3] Kajal Dakhare;Vidhi Wankhede;Prateek Verma A Survey on Recognition and Translati on System of Real-Time Sign Language,2024 2nd DMIHER International Conference on Artificial Intelligence in Healthcare, Education and Industry (IDICAIEI)
[4] Gladys Jessica Ruslim; Nicholas Matthew Salim; Ivan Sebastian Edbert; Derwin Suhartono,Sign Language Detection to Enhance Online Communication for Deaf and Mute Individuals through Deep Learning Models,2024 International Conference on Computer Engineering,
[5] Network, and Intelligent Multimedia (CENIM)



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY