AI PROJECT

“REAL TIME

SIGN LANGUAGE RECOGNITION”

Subgroup-2C12

(2024-2025)

Project Report File

**SUBMITTED TO:**

DR. JINEE GOYAL

**SUBMITTED BY:**

1. RAIZA DUGGAL (102303068)
2. SNEHA GUPTA (102303033)
3. HIMANGI DEVGUN (102303035)

ABSTRACT

This project addresses the critical communication gap between hearing-impaired individuals and the general public by developing a real-time sign language recognition system. The system leverages computer vision and machine learning techniques to interpret hand gestures and convert them into text, thus facilitating seamless communication. A standard webcam is used to capture hand gestures, which are processed using MediaPipe to extract 21 key hand landmarks. These landmarks serve as input features for a Random Forest Classifier, trained to recognize and classify different signs. Tools like OpenCV are used for real-time video feed processing and visualization. The project shows that with affordable and open-source technologies, it is possible to build scalable and effective solutions for the deaf and hard-of-hearing community.

INTRODUCTION

Communication is a fundamental human right, and for individuals with hearing or speech impairments, sign language serves as their primary mode of interaction. However, the lack of widespread understanding of sign language among the general public poses significant challenges. Recent advancements in computer vision and machine learning provide an opportunity to automate the interpretation of sign language. This project implements a real-time sign language recognition system using hand landmark detection and machine learning to recognize static gestures.

PROBLEM STATEMENT

Despite being a primary communication method for millions of hearing-impaired individuals, sign language remains largely inaccessible to those unfamiliar with it, leading to communication barriers in daily life. Traditional solutions like human interpreters or specialized gloves are often expensive, impractical, or unavailable. There is a need for a cost-effective, real-time system that can interpret sign language using commonly available devices like webcams. This project addresses this gap by developing a system that recognizes hand gestures through computer vision and machine learning, translating them into text to facilitate smoother, more inclusive communication between hearing-impaired users and the general population.

OBJECTIVES

* To develop a gesture recognition system that uses hand landmarks to classify sign language gestures.
* To build a custom dataset of hand gestures representing alphabets and numbers.
* To train a machine learning model (Random Forest) to classify gestures with high accuracy.
* To implement a real-time prediction interface using webcam input.
* To ensure the system is scalable and modifiable for additional gestures or users.

METHODOLOGY

**1.** **Tools and Libraries**: Python .OpenCV for webcam and frame handling. MediaPipe for hand landmark detection. scikit-learn for training the Random Forest Classifier. joblib for model serialization.

**2. Data Collection:** Users display hand signs in front of the webcam. MediaPipe extracts 21 landmarks (x, y coordinates) per hand. The landmark data is appended to a CSV file along with the gesture label.

**3. Feature Extraction:** Each gesture is represented by 42 features (x and y coordinates of 21 landmarks).

**4. Model Training:** Data is split into training and testing sets using train\_test\_split. A Random Forest Classifier with 100 trees is trained.Accuracy is evaluated on the test set.

**5. Real-time Prediction:**The trained model is loaded.Webcam captures live video.Each frame is processed to detect hand landmarks.The model predicts the gesture based on the landmark vector.

LIMITATIONS

* Supports only one hand and one user at a time.
* Can't differentiate subtle finger movements (e.g., similar signs).
* Limited vocabulary unless dataset is expanded.
* No continuous sign sentence interpretation (word-level only).

RESULTS

* The Random Forest Classifier trained on the custom dataset achieved an **average test accuracy of approximately 99%**.
* Real-time prediction using a webcam successfully recognized static hand gestures for alphabets with high consistency.
* The system required users to hold each gesture for **about 1 to 1.5 seconds** for reliable detection and prediction.
* The live feedback was responsive with minimal delay, demonstrating smooth real-time recognition.
* The system worked best under good lighting conditions and with a clear background.
* Misclassification was rare but occurred occasionally with gestures that had similar hand shapes or finger positions.
* The dataset used in training included **26 alphabet gestures**, collected manually through webcam input.

CONCLUSION

The Real-Time Sign Language Recognition System successfully demonstrates how computer vision and machine learning can enhance communication accessibility for the deaf and hard-of-hearing community. By using MediaPipe for hand landmark detection and a Random Forest Classifier for gesture recognition, the system efficiently translates hand signs into text in real time. This solution is low-cost, uses open-source tools, and runs on basic hardware, making it feasible for broader adoption in educational and assistive technology settings. Despite its current limitations, such as a restricted vocabulary and sensitivity to lighting and hand positioning, the project lays a solid foundation for more advanced systems incorporating a larger dataset and more robust deep learning models. Future improvements could include support for dynamic gestures, multilingual sign support, and integration with speech synthesis. Overall, the project highlights the transformative role of AI and computer vision in making communication more inclusive and breaking down long-standing accessibility barriers.

REFERENCES

* Google MediaPipe: <https://mediapipe.dev/>
* OpenCV Library: <https://opencv.org/>
* scikit-learn Documentation: <https://scikit-learn.org/>
* Hand Gesture Recognition Papers and Studies
* Joblib Documentation: <https://joblib.readthedocs.io/en/latest/>
* Real-Time Hand Gesture Recognition using Machine Learning – ResearchGate
* Random Forest Classifier - Breiman, L. (2001). Machine Learning Journal
* Sign Language Alphabet Dataset (custom-built or this project)

FUTURE SCOPE

**1. Use a More Advanced Model (CNN, LSTM, etc.)**

Explanation:

Replace the Random Forest with a deep learning model like a Convolutional Neural Network (CNN) or a Recurrent Neural Network (RNN), such as LSTM.CNNs can be used if you use images directly instead of just hand landmarks.LSTMs or Temporal Convolutional Networks can handle dynamic signs (like gestures over time).

**2. Support Dynamic Gestures**

Explanation:

Right now, your system only supports static signs (like individual letters). Dynamic signs (like "hello", "thank you") involve movement over time. You could capture sequences of frames and use models like LSTM or 3D CNNs to recognize the sequence. Add temporal smoothing to reduce noise in predictions.

**3. Data Normalization and Scaling**

Explanation:

Currently, raw (x, y) coordinates are used, which may vary based on hand distance from the camera. Normalize coordinates based on hand size or relative distance between landmarks.This will reduce model confusion caused by varying hand sizes or positions.

**4. Augment and Expand Dataset**

Explanation:

More diverse training data leads to better generalization. Capture data in different lighting, angles, and hand orientations. Include data from multiple users with different hand shapes and skin tones.

**5. Use a GUI or Mobile App**

Explanation:

Improve user experience by building: A desktop GUI using Tkinter or PyQt. A mobile app using TensorFlow Lite or MediaPipe Android/iOS SDK.

**6. Add Audio/Text-to-Speech Output**

Explanation:

For real-world use, after predicting a gesture, convert it to spoken words using text-to-speech libraries like pyttsx3 or gTTS.

**7. Support Two-Handed Signs**

Explanation:

Many signs use both hands. MediaPipe supports detecting multiple hands, so your system can be extended to recognize dual-hand gestures.

**8. Implement Confidence Threshold**

Explanation:

Add a threshold for prediction confidence. Only accept predictions when the model is sufficiently sure (e.g., ≥ 80%). Helps reduce false positives. Can be visualized in the output window.

**9. Cloud or API Integration**

Explanation:

Turn your model into a service using Flask or FastAPI, or deploy it on the cloud (e.g., AWS Lambda, GCP) so it can be accessed via API.

**10. Train with More ML Algorithms for Benchmarking**

Explanation:

Try and compare other ML algorithms like: K-Nearest Neighbors (KNN). Support Vector Machines (SVM) XGBoost LightGBM . Choose the one with the best balance of accuracy and inference speed.