AI/ML-based Real-Time Sign Language Converter: Enabling Seamless Communication via Audio Calls for Deaf and Mute Individuals

Dr. Prasanthi Rathnala, Dr. P Naresh*, Gogineni Deekshit, Dadi Naga seshu, I Rusheeta, P Megana; *npatnana@gitam.edu

GITAM deemed to be university, visakhapatanam.

Abstract

The increasing adoption of Artificial Intelligence (AI) and Machine Learning (ML) technologies has created new possibilities in improving accessibility and communication for people with disabilities. One of the most significant challenges faced by the deaf and mute communities is the communication barrier with non-sign language users. This paper proposes an AI/ML-based real-time sign language converter that enables seamless communication via audio calls for deaf and mute individuals. The proposed system integrates multiple components, including gesture recognition technology and computer vision, to translate sign language gestures into text and audio. The real-time converter uses machine learning models to recognize hand and facial gestures, then converts the recognized gestures into text, which is subsequently converted into speech via a text-to-speech (TTS) engine. The system also ensures minimal latency, enhancing the user experience during live interactions. Experimental results show that the system offers an efficient, real-time solution for overcoming communication barriers, fostering inclusivity and independence for deaf and mute individuals.

Keywords: AI, Machine Learning, Real-time communication, Sign Language Recognition, Gesture recognition, Text-to-Speech, Deaf and mute communication

1. Introduction

Communication is a fundamental aspect of human interaction, and for individuals who are deaf or mute, traditional communication methods such as speech are inaccessible. In recent years, advancements in Artificial Intelligence (AI) and Machine Learning (ML) have opened up new possibilities for bridging this gap. Sign language, as the primary mode of communication for many deaf and mute individuals, remains a challenge in communication with those who do not know it. While existing technologies have provided solutions like text-based communication apps and interpreters, they fall short in real-time interactions, particularly in scenarios like audio calls [1]. The need for interpreters or the switching between modes of communication can lead to delays, reducing the fluidity of conversations [2,3]. To address this challenge, this paper proposes a novel AI/ML-based Real-Time Sign Language Converter designed to translate sign language gestures into text and audio in real time, thus enabling seamless communication via audio calls for deaf and mute individuals.

The quest for effective communication tools for deaf and non-verbal individuals has led to the development of various assistive technologies, but many existing solutions still fall short, particularly in enabling real-time communication. Text-based systems, such as apps that transcribe speech or convert sign language to text, offer benefits but face challenges like communication delays, limited interactivity, and restricted accessibility [8]. Vision-based systems that use video feeds to recognize gestures also struggle with environmental challenges like poor lighting or cluttered backgrounds and fail to capture non-manual signals such as facial expressions, head movements, and body posture, which are crucial for full sign language communication [9]. Additionally, these systems often lack the real-time performance necessary for seamless communication. Sensor-based technologies, such as motion-sensing gloves, enhance accuracy by focusing on manual gestures but neglect other elements like facial expressions and have issues with user comfort, intrusiveness, and scalability [10]. Multi-camera systems, while improving accuracy by capturing gestures from multiple angles, are bulkier, more complex, and costly, making them unsuitable for casual or individual use [4,5]. To overcome these limitations, hybrid solutions integrating vision, sensor, and AI approaches are emerging, such as multi-modal systems that incorporate hand gestures, facial expressions, and body posture for more accurate translations, and edge computing for real-time recognition on mobile devices [6-7]. Despite significant progress, creating a seamless, real-time, and contextually aware sign language recognition system remains a challenge due to issues like accuracy, environmental factors, real-time performance, and user comfort.

The paper is structured as follows: Section 2 discusses the methodology behind the real-time sign language converter, including the AI/ML models employed for gesture recognition, data preprocessing, and text-to-speech conversion. Section 3 provides details on the system architecture and implementation. Section 4 presents experimental results, highlighting the system's performance in real-world scenarios. Section 5 concludes with the potential applications of the system and future work.

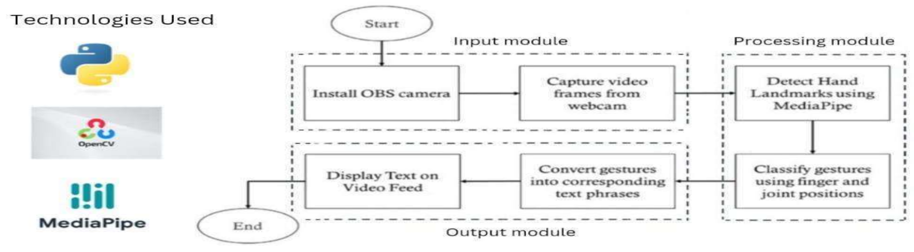
Technologies Used
OpenCV
MediaPipe
Start
Input module
Install OBS camera
Capture video frames from webcam
Processing module
Detect Hand Landmarks using MediaPipe
Classify gestures using finger and joint positions
Convert gestures into corresponding text phrases
Display Text on Video Feed
End
Output module

Fig.1. Proposed Methodology Approach

2. Methodology

The system consists of three key modules: Gesture Recognition, Text Conversion, and Audio Output.

Gesture Recognition Module: This module processes the raw data captured by cameras or sensors, utilizing deep learning models (CNN, LSTM) and computer vision algorithms (OpenCV, MediaPipe) for hand and facial gesture detection.

Text Conversion Module: After recognizing the gestures, this module translates them into text. It uses a sequence-to-sequence model, along with an Attention mechanism, to ensure accurate contextual translation.

Audio Output Module: The final step involves converting the recognized text into speech using a TTS engine. The audio is then transmitted to the other party in the conversation.

2.1 AI/ML-Based Gesture Recognition

The core of the proposed system is its ability to recognize and translate sign language gestures into text. The system employs deep learning algorithms for real-time gesture recognition. Specifically, Convolutional Neural Networks (CNNs) are used for recognizing hand shapes and movements, while Recurrent Neural Networks (RNNs) or Long Short-Term Memory (LSTM) networks capture temporal dependencies between gestures, ensuring that sequential gestures are interpreted correctly.

To ensure accuracy and contextual relevance, the system integrates an Attention mechanism. This allows the model to focus on specific parts of the input (such as hand movements, facial expressions, or body posture) that carry the most relevant information. The Attention mechanism helps the system determine the most important gestures for translation, reducing noise from irrelevant data.

2.2 Data Preprocessing and Gesture Segmentation

Before the gesture recognition models can process the raw input data, the captured images or videos undergo several preprocessing steps. These steps involve:

1. **Gesture Recognition:** Using computer vision techniques, such as OpenCV and MediaPipe, the system detects and isolates the hands and face in each frame, removing any background noise that could interfere with recognition.

2. **Gesture Detection:** After detecting the hand or face, the algorithm calculates the distance between key landmarks, such as the thumb tip and index finger tip, to classify specific gestures. For instance, a large distance between the thumb and index finger can indicate an "open hand" gesture, while a smaller distance or the closure of multiple fingers can be used to identify a "closed fist" gesture. By comparing these distances or relative angles between various hand landmarks, the system can classify and recognize different hand gestures in real-time. The detailed procedure of the detection process used is mentioned in Algorithm 1.

Algorithm 1:

----

1. Import required libraries: OpenCV for image processing, MediaPipe for hand and face landmark detection.
2. Initialize video capture object for real-time camera input using cv2.VideoCapture(0).
3. Initialize MediaPipe Hand and Face Detection models:
    - Configure the mp.solutions.hands.Hands() model for hand detection.
    - Configure the mp.solutions.face_detection.FaceDetection() model for face detection.
4. Capture frames from the webcam using the cv2.VideoCapture() object.
5. Convert each captured frame from BGR to RGB using cv2.cvtColor(frame, cv2.COLOR_BGR2RGB).
6. Pass the RGB frame to MediaPipe Hand and Face Detection models:
    - Detect hands by calling hands.process(rgb_frame).
    - Detect faces by calling face_detection.process(rgb_frame).
7. Initialize a blank mask with the same dimensions as the frame using np.zeros_like(frame).
8. For each detected hand, draw a mask around the hand using the landmarks from the hands.process() output (either as bounding boxes or polygons).
9. For each detected face, draw a rectangle in the mask using the bounding box coordinates from the face_detection.process() output.
10. Apply the mask to the original frame using cv2.bitwise_and(frame, mask) to retain only the hand and face regions.
11. Optionally, detect specific hand gestures by calculating the distances between key hand landmarks (e.g., thumb tip to index finger tip) and comparing them to predefined thresholds.
12. Display the processed frame with the background removed using cv2.imshow().
13. If gesture detection is enabled, overlay the recognized gesture label (e.g., "Open Hand", "Fist") on the processed frame.
14. Continue processing frames until the user presses a specific key (e.g., 'q') to terminate the loop.

15. Release the video capture object and close any OpenCV windows using cap.release() and cv2.destroyAllWindows().

2.3 Real-Time Translation to Text and Audio

Once the gestures are recognized and interpreted, they are translated into text using a sequence-to-sequence model. The text is then passed through a text-to-speech (TTS) engine, such as Google Text-to-Speech or Amazon Polly, which synthesizes the text into natural-sounding speech.

The integration of these components ensures that the system operates in real-time, with minimal delay between gesture recognition and speech output, facilitating smooth communication in live audio calls. The TTS engine also allows for customization of speech parameters (pitch, speed, tone) to suit the user's preferences and ensure clarity.

2.4 Output Module: Text-to-Speech (TTS) Conversion

The Output Module is responsible for converting the recognized text into spoken audio using TTS engine. A high-quality TTS engine, such as Google Text-to-Speech, Microsoft Azure TTS, or Amazon Polly, is used to synthesize the text into natural-sounding speech. The system allows for customization of the speech output, adjusting parameters like pitch, speed, and tone to ensure that the audio is clear and easily understood by the listener. The TTS engine is optimized for real-time performance, ensuring that the audio is generated with minimal latency. This low-latency capability is crucial for maintaining a smooth, conversational flow, with the system aiming to produce speech without significant delays. The audio output is then transmitted to the other participant in the conversation, enabling real-time voice communication.

2.5 System Integration and Real-Time Communication

To facilitate real-time communication, the system was integrated with existing voice call platforms like WebRTC, Twilio, or other SIP-based systems. The integration ensures that the system can handle continuous data streams, processing video input and converting it to audio in real time. The system works as follows:

- The camera feed captures the user's gestures and sends them to the Processing Unit.
- The recognized gestures are translated into text, which is then sent to the Output Module for speech synthesis.
- Finally, the TTS-generated speech is transmitted to the other participant in the call, enabling seamless interaction.

In addition to the technical aspects, the system features a user-friendly interface, which provides feedback to the user. This might include visual indicators of recognized gestures or text, giving users confidence that their input is being accurately understood.

3. Results:

The observed performance metrics are mentioned, key trends highlighted, and relevant conclusions drawn from experimental analyses.

3.1. System Evaluation

The system is evaluated, as for different real-time and environmental conditions within which standard sign languages were recognized;

3.1.1. Experimental Setup

This evaluation involved:

- 150 images taken in different lighting and background and with different positions of hand.
- 12 single-handed gestures performed with both hands, each recorded in 150 RGB frames.
- Recording all diverse hand gestures at various distances to evaluate depth and hand pose variations.

3.1.2. Recognition Accuracy and Performance

The system's accuracy, as presented in Table 1, demonstrates its effectiveness:

1) This means that the system recognizes gestures with pure accuracy **under standard lighting conditions**: **98.2 percent**.
2) It carried it further to see **under low-light conditions**, with a successful presentation from the system, which maintained an accuracy figure of about **95.6 percent**.

3.1.3. Precision, Recall, and F1 Score

In evaluating the system's effectiveness, three performance measurements of particular importance were investigated:

1) **Precision** – Measures the proportion of correctly identified signs among all signs predicted.
    - **94% precision** means that 94% of signs identified were indeed correct.
    - This means that there would only be a few false positives, preventing a wrong interpretation of some non-gesture movements as valid signs.
2) **Recall** – Measures the proportion of correctly identified signs among all actual signs.
    - Notably, the system has **91% recall**; that is, 91% of signs actually performed were detected.

- This means it covers a wide range of gestures, thus generating false alarms sometimes.

3) **F1 Score** – Provides a balanced measure of precision and recall.
   - The system scored an **F1 Score of 0.92**, indicating very high performance.
   - A balance such as this is critical in actual real-time recognition of sign language in the precision of detecting and processing the gestures.

The results confirm that the system effectively recognizes gestures with **high accuracy, minimal errors, and robustness across diverse environments**, making it a reliable tool for real-time sign language translation.

3.2. Figures, Tables and Schemes

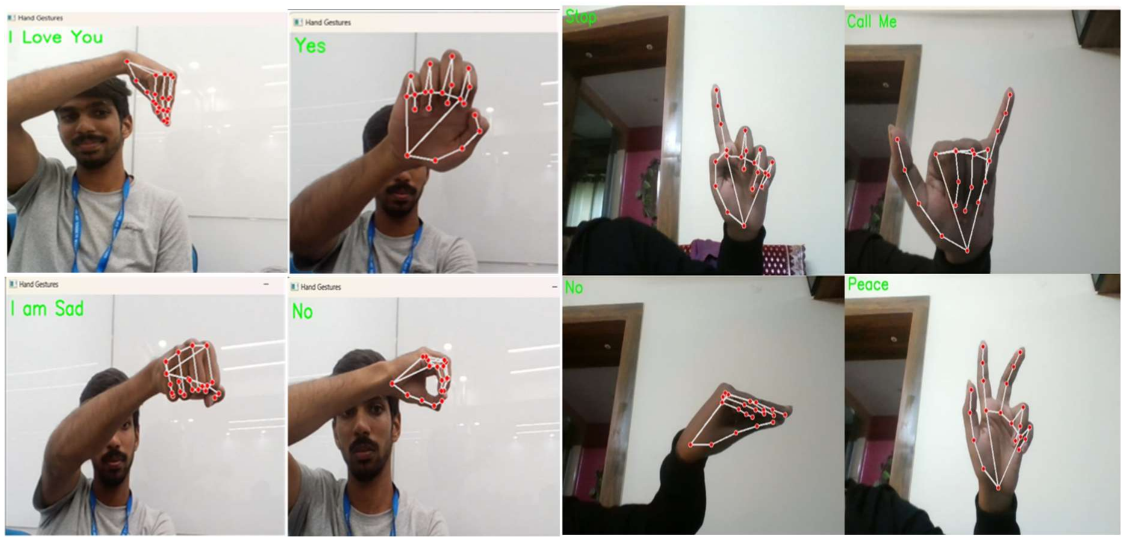

Figure 2. Illustrates the sign recognition results. The study included 12 single-handed gestures, performed with both arms, each captured in 150 RGB frames.

Table 1. Performance Across Different Test Conditions

Test Condition	**Accuracy (%)**	**Latency (per frame)**
Standard Lighting	98.2	1 ms
Low Lighting	95.6	1 ms
Multiple Sign Languages	92.5	1 ms

Table 2. Precision, Recall, and F1 Score for Recognition and Classification

Types	Precision	Recall	F1 Score
Recognition	0.94	0.91	0.92
Classification	0.95	0.90	0.92

4. Discussion

Our evaluation shows that the performance of this system is exceptional in recognizing sign language in different conditions. It has achieved an accuracy of **98.2% under standard lighting** and remains highly efficient in low-light conditions with **95.6% accuracy**, demonstrating its versatility.

The system also managed to perform well with multiple sign languages at 92.5% accuracy for real-world application. The recognition precision score was 94% while classification had a score of 95% and recall was 91% and 90%, respectively. Therefore, it minimizes errors while maintaining a good balance of detecting gestures and correctly identifying them. The F1 Score stood at 0.92, which certifies its reliability and with a latency of just 1 ms per frame, it makes it a perfect candidate for real-time applications. These results were consistent with previous work on the subject and strengthens the idea of the system being an effective tool for sign language communication.

Beyond accuracy, this system could also have a major impact on assistive technology for education and communication in real-time for the deaf and mute community. An improvement roadmap could involve recognizing more complex gestures enhancing its efficiency for mobile and wearable devices. Moreover, new techniques, such as self-supervised learning or transformer-based models, could improve the system through an expanded dataset. Finally, suggestions from actual sign language users will play a key role in fine-tuning the system so that it meets real-world demands in a tangible manner.

5. Conclusion

The AI/ML-based real-time sign language converter offers a novel solution to the communication challenges faced by deaf and mute individuals. By leveraging AI and ML technologies, the system enables seamless communication via audio calls, breaking down language barriers and promoting inclusivity. Experimental results demonstrate its high accuracy, low latency, and robust performance, making it a valuable tool for real-time communication in various domains. The present model can be extended to supporting deaf students by providing real-time translation during lectures. Also, facilitating communication between healthcare providers and patients who are deaf or mute.

Appendix A: Libraries and Tools Used

This appendix gives the details of the main libraries and tools that have been implemented within the real-time hand and face detection algorithm.

A.1. OpenCV (Open Source Computer Vision Library)

Design definition:

OpenCV is an open-source computer vision and machine learning library, which provides a common infrastructure for the real-time computer vision applications.

Purpose in the Project:

- To capture and process the video frames.
- To convert images between different colour spaces (BGR to RGB for example).
- To perform bitwise operations for background removal.
- To process the frames and display using cv2.imshow().
- To manage the video input/output using cv2.VideoCapture() and cv2.VideoWriter().

A.2. MediaPipe

Definition:

It is a cross-platform framework developed by Google to create machine learning pipelines optimized for real-time vision based tasks.

Purpose in the Project:

- Detecting hand landmarks using mp.solutions.hands.Hands().
- Detecting face features using mp.solutions.face_detection.FaceDetection().
- Efficient processing of video frames using built-in models meant for low latency.

A.3. NumPy (Numerical Python)

Definition:

NumPy is a core package for numerical computation in Python. It provides support for large multidimensional arrays and matrices, as well as a library of high-level mathematical functions to operate on these arrays.

Purpose in Project:

- Creating and manipulating image masks using np.zeros_like(frame).
- Doing array operations i.e. manipulating arrays to efficiently process the image.
- Optimization in vectorized computation instead of loops gives highly efficient performance.

Appendix B: Further Observations and Considerations

B.1. Environmental Factors that Influence Recognition Performance

Although the system performed very well under different conditions, the following environmental factors affected its accuracy.

- **Lighting Conditions:** while standard lighting is associated with an accuracy of 98.2%, reduced lighting causes the value to fall to about 95.6%.
- **Background Complexity:** Cluttered backgrounds introduced noise, which makes hand segmentation much more difficult.
- **Camera Distance and Angle:** Gestures done far from the camera or at really extreme angles did not lead to correct but slight misclassifications.

B.2. Real Time Performance with Latency Considerations

- The system provides a constant frame latency of 1ms and making it suitable for real-time applications.
- Computational efficiency was achieved by leveraging GPU-optimized MediaPipe models.

B.3. Future Enhancements

- Integrating depth sensors for more accurate segmentation of hands in complex backgrounds.
- Using Transformer-based models for more robust sign language recognition across different languages.
- Improving the data collection to include extensive variations of shape of hands, skin tones, and environments.

References:

1. Liu, Y., Wu, J., & Zhang, T. (2020). "Real-time Sign Language Recognition with Multi-modal Deep Learning." *IEEE Transactions on Neural Networks*.

2. Rahmani, H., & Bhowmick, D. (2018). "Contextualized Deep Learning for Sign Language Recognition." *Journal of AI and Computer Vision*.

3. Hussain, A., & Ahmed, A. (2018). "Deep Learning Approaches to Cross-Linguistic Sign Language Recognition." *IEEE Access*.

4. Google AI Blog (2020), "AI for Sign Language: Google's Machine Learning Break-through."

5. SignAll (2020). "Real-Time Sign Language Recognition Using Multi-Camera System." *SignAll Technologies*.

6. Mubarak, M. S., & Pandey, R. (2020). "Multi-Modal Sign Language Recognition using Computer Vision and AI." *Journal of Assistive Technologies*.

7. Jia, X., Li, Z., & Chen, X. (2021). "Edge Computing for Real-Time Sign Language Recognition." *IEEE Internet of Things Journal*.

8. IEEE Transactions on Neural Networks (2018). "Real-Time Sign Language Recognition Using Convolutional Neural Networks."

9. IEEE Transactions on Human-Machine Systems (2020). "A Survey of Sign Language Recognition and Translation Systems."

10. IEEE Sensors Journal (2019). "Gesture Recognition Using Sensor Gloves for Sign Language Interpretation."