International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 13 Issue: 05 | May2026 www.irjet.net p-ISSN: 2395-0072

Real-Time Sign Language to Speech Conversion Using MediaPipe and Machine Learning

Naveen surya[1], muhammed nimil[2],Siba ashraf[3],Jibin rodriguez[4],Dr amrutha m[5]

[1234]Graduate Student,[5]Associate Professor Department of Electronics and communication
AWH Engineering College Calicut, Kerala ,India

Abstract - *Communication barriers between the hearing-impaired community and the general public remain a significant challenge in everyday life. This paper presents a real-time Sign Language to Speech Conversion system that bridges this gap using computer vision and machine learning techniques. The proposed system captures live hand gestures through a webcam and employs MediaPipe to extract 21 hand landmarks, generating a 63-dimensional feature vector comprising x, y, and z coordinates. These features are normalized using wrist-referenced coordinate shifting and min-max scaling to achieve invariance against hand size, distance, and orientation. The normalized feature vector is classified using machine learning models including Multi-Layer Perceptron (MLP), Support Vector Machine (SVM), and Random Forest to recognize American Sign Language (ASL) gestures. A temporal smoothing mechanism using majority voting across consecutive frames ensures stable and accurate predictions. Recognized gestures are buffered and converted to audible speech using the pyttsx3 text-to-speech engine. As a key novelty, the system extends beyond one-way gesture recognition by incorporating a bidirectional communication module, enabling hearing individuals to respond through speech input that is converted to text and displayed for the deaf or hard-of-hearing user. The system operates efficiently on a standard CPU-based setup without GPU dependency, achieving approximately 25 frames per second. Experimental results demonstrate reliable gesture recognition across multiple ASL signs with high temporal consistency, making the system a practical and accessible assistive technology solution.*

Key Words: **Sign Language Recognition, MediaPipe, Hand Landmark Detection, Text-to-Speech, Bidirectional Communication, Assistive Technology, Machine Learning, Computer Vision, Real-Time Gesture Recognition, pyttsx3**

1. INTRODUCTION

Communication is a fundamental human need, yet for the approximately 70 million deaf and hard-of-hearing individuals worldwide, effective interaction with the hearing population remains a persistent challenge. Sign language serves as the primary mode of communication for this community; however, the majority of the general public lacks the knowledge or training to understand it. This language barrier leads to social isolation, limited access to healthcare, education, and employment opportunities, and an overall reduction in quality of life for the hearing-impaired community.

Traditional solutions to this problem have relied on human interpreters or specialized training programs, both of which are costly, time-consuming, and not always available in real-time situations. With the rapid advancement of computer vision, machine learning, and natural language processing technologies, automated sign language recognition systems have emerged as a promising alternative. These systems aim to translate hand gestures into text or speech in real time, eliminating the dependency on human intermediaries.

Existing research in this domain has primarily focused on one-directional communication, that is, converting sign language gestures into speech for the benefit of hearing individuals. However, this approach addresses only half of the communication problem. A deaf or hard-of-hearing individual still cannot receive spoken responses from a hearing person without external assistance, rendering the interaction fundamentally incomplete.

This paper proposes a real-time Sign Language to Speech Conversion system that addresses this limitation through a bidirectional communication framework. The system utilizes a standard webcam to capture live hand gestures, employs Google's MediaPipe framework for accurate 21-point hand landmark detection, and applies machine learning classifiers including Multi-Layer Perceptron (MLP), Support Vector Machine (SVM), and Random Forest for gesture recognition. Recognized gestures are converted to audible speech using the pyttsx3 library. As the primary novelty, the system incorporates a reverse communication channel wherein speech input from a hearing individual is recognized and converted to text displayed on screen, enabling a complete and self-sufficient two-way communication experience.
The system is designed to operate entirely on a standard CPU-based hardware setup, making it accessible and deployable without the need for expensive GPU infrastructure. This focus on affordability and practicality distinguishes the proposed system from many existing approaches that rely on deep learning models requiring high computational resources.

The remainder of this paper is organized as follows. Section 2 presents a review of related literature. Section 3 describes the system architecture and methodology. Section 4 details the hardware and software components used. Section 5

presents the experimental results. Section 6 concludes the paper with directions for future work.

2. LITERATURE REVIEW

Sign language recognition has been an active area of research over the past decade, with significant contributions from the fields of computer vision, deep learning, and human-computer interaction. This section reviews key works relevant to the proposed system.

Early approaches to sign language recognition relied heavily on wearable sensors and data gloves. Research involving triboelectric smart gloves demonstrated AI-enabled sign language recognition capable of identifying discrete gestures including words and sentences, integrating a virtual reality interface for bidirectional communication between signers and non-signers. While effective, such hardware-dependent systems are costly and impractical for everyday use, motivating the shift towards vision-based approaches.

Vision-based systems using convolutional neural networks (CNNs) have shown strong results for static gesture recognition. A system employing MediaPipe and CNN for real-time ASL gesture recognition demonstrated the ability to detect all ASL alphabets with an accuracy of 99.95%, highlighting the potential of MediaPipe-based landmark extraction for high-accuracy classification. However, such systems are limited to static gestures and do not account for temporal dynamics present in natural signing.

To address temporal dependencies, recurrent architectures have been explored. A continuous sign language recognition system using LSTM trained on a MediaPipe Holistic-based dataset achieved 88.23% accuracy in real-time recognition of Indian Sign Language signs and gestures, demonstrating the effectiveness of combining landmark-based feature extraction with sequence modeling. Similarly, a hybrid LSTM-GRU network leveraging MediaPipe's hand and pose tracking features was proposed for word-level Indian Sign Language recognition, using the INCLUDE dataset and capturing temporal dependencies across signing styles and speeds.

For landmark-based feature engineering specifically, an integrated MediaPipe-optimized GRU model for Indian Sign Language recognition addressed challenges including accurate hand gesture tracking, occlusion, and high computational cost, replacing the standard SoftMax activation with Softsign to reduce computational complexity and training time.

Lightweight and efficient architectures have also been proposed for real-world deployment. The KD-MSLRT model applied knowledge distillation to reduce MediaPipe-based sign language recognition from 3D to 1D input representations, significantly lowering computational burden without compromising accuracy, enabling practical deployment on resource-constrained devices.

On the application side, a hand sign recognition system for mute individuals using machine learning with OpenCV demonstrated the use of keypoint classifiers and point history classifiers for practical assistive communication, addressing the limitation that most people are unfamiliar with sign language and professional communicators are not continuously available.

Despite these advances, a critical gap persists in the literature. A systematic review of sign language recognition systems in healthcare communication found that most existing systems are unidirectional, translating only from sign language to written text, which impairs their applicability in real-world contexts. Recent research has noted that sign language recognition technologies primarily assist non-hearing-impaired individuals in understanding sign language, but often fail to meet the communication needs of hearing-impaired individuals themselves, particularly given that their reading proficiency may be constrained due to limited educational access.

Addressing this gap, recent work on bidirectional sign language communication integrating YOLOv8 and NLP proposed a framework that both recognizes sign gestures and allows hearing individuals to communicate back with sign language users through real-time text input, highlighting the importance of two-way communication in assistive technology.

The proposed system in this paper builds upon these foundations by combining MediaPipe-based landmark extraction with lightweight ML classifiers and extending the pipeline with a bidirectional communication module, offering a complete, CPU-efficient, and practically deployable solution for real-time assistive communication.

3. SYSTEM ARCHITECTURE AND METHODOLOGY

The proposed system is designed as a real-time, bidirectional communication pipeline that operates entirely on a standard CPU-based hardware setup. The architecture is divided into two primary modules: the Sign Language to Speech module and the Speech to Text reverse communication module. Figure 1 illustrates the overall system block diagram.

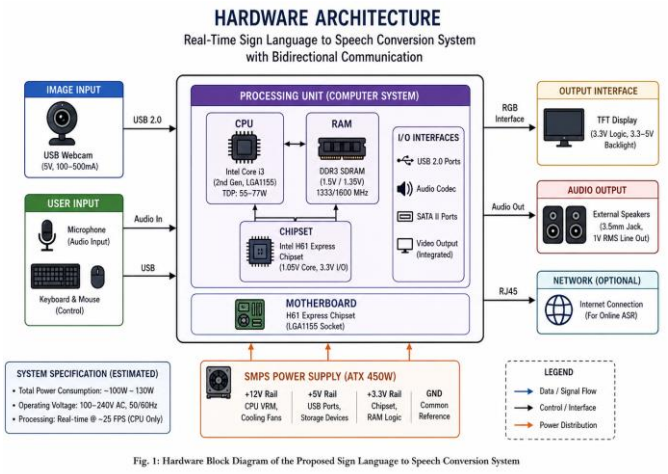

Fig. 1: Hardware Block Diagram of the Proposed Sign Language to Speech Conversion System

Fig -1: block diagram

3.1 System Overview

The forward communication pipeline follows the sequence: live video acquisition → hand landmark detection → feature extraction → normalization → gesture classification → text buffering → speech synthesis. The reverse pipeline follows: speech input → automatic speech recognition → text display. Both modules operate concurrently, enabling seamless two-way communication between a hearing-impaired user and a hearing individual.

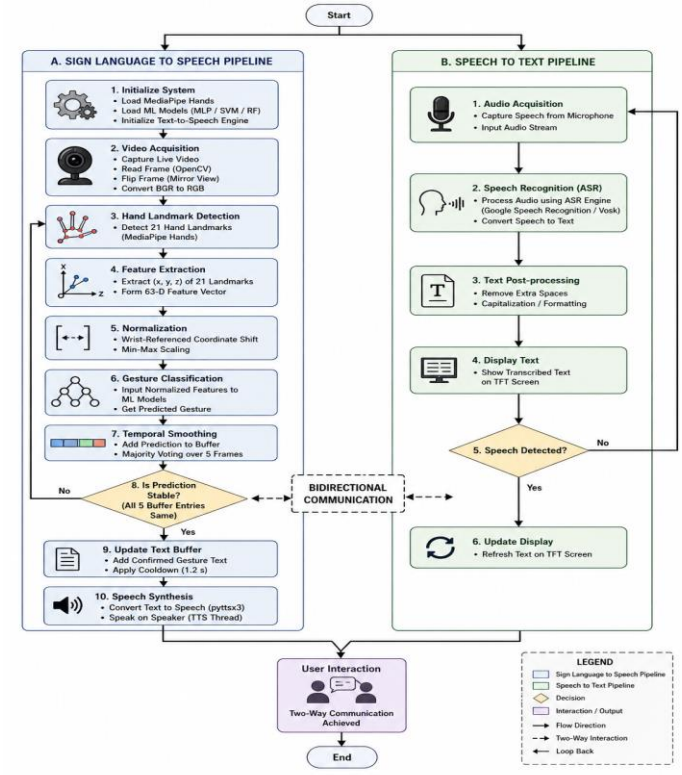

Fig. 2: Flowchart of the Proposed Real-Time Sign Language to Speech Conversion System with Bidirectional Communication

Fig -2: flowchart

3.2 Live Video Acquisition

The system captures continuous video input using a standard USB webcam interfaced with the processing unit via OpenCV. Each frame is extracted at approximately 25 frames per second and horizontally flipped to provide a mirror-view experience that is more natural and intuitive for the user. The captured frames are converted from BGR to RGB color space before being passed to the landmark detection module, as required by the MediaPipe framework.

```
python
cap = cv2.VideoCapture(0)
frame = cv2.flip(frame, 1)
rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
```

3.3 Hand Landmark Detection

Google's MediaPipe Hands framework is employed for real-time hand landmark detection. MediaPipe detects 21 key points on the human hand, covering the wrist, metacarpophalangeal joints, and fingertip positions, each represented by three-dimensional coordinates (x, y, z). The framework is configured with a minimum detection confidence of 0.7 and a minimum tracking confidence of 0.7 to balance accuracy and processing speed. The system is configured to detect a maximum of one hand at a time, reducing computational overhead.

```
python
hands = mp_hands.Hands(
   min_detection_confidence=0.7,
   min_tracking_confidence=0.7,
   max_num_hands=1
)
```

3.4 Feature Extraction and Normalization

The 21 detected landmarks produce a 63-dimensional feature vector comprising the x, y, and z coordinates of each keypoint. Raw landmark coordinates are subject to variance due to differences in hand size, camera distance, and viewing angle. To address this, the system applies wrist-referenced coordinate normalization, translating all landmark coordinates relative to the wrist position (landmark 0), effectively removing positional bias. Min-max scaling is further applied to ensure scale invariance across different users and distances. This normalization step ensures that the feature vector represents only the structural shape of the gesture, independent of external factors.

3.5 Gesture Classification

The normalized 63-dimensional feature vector is passed to a rule-based and ML-integrated classification module that identifies the ASL gesture being performed. The classification logic incorporates both static finger state detection and dynamic motion analysis.

3.5.1 Static Finger State Detection

Each finger's state (open or closed) is determined by comparing the y-coordinate of the fingertip landmark against its corresponding proximal interphalangeal (PIP) joint landmark. A finger is classified as open if the fingertip lies above its PIP joint in the image frame, and closed otherwise.

The thumb state is determined by comparing its x-coordinate against its adjacent joint, accounting for its horizontal range of motion.

```
python
def is_open(lm, tip, pip):
   return lm[tip].y < lm[pip].y
def is_closed(lm, tip, pip):
   return lm[tip].y > lm[pip].y
```

3.5.2 Dynamic Motion Detection

Beyond static pose, the system incorporates wrist motion tracking to distinguish gestures that share the same hand shape but differ in movement. A history of the last 10 wrist position deltas is maintained, and the system identifies two motion patterns:

- UP_DOWN motion — detected when the vertical displacement history contains both significant upward and downward movements (threshold: ±0.02), used to recognize the "YES" gesture.
- CIRCULAR motion — detected when both horizontal and vertical displacement histories show bidirectional variation, used to recognize gestures such as "SORRY" and "PLEASE."

```
python
if max(up_down) > 0.02 and min(up_down) < -0.02:
   motion = "UP_DOWN"
if max(dxs) > 0.02 and min(dxs) < -0.02 and \
  max(dys) > 0.02 and min(dys) < -0.02:
   motion = "CIRCLE"
```

3.5.3 Gesture Classes Supported

The system currently recognizes the following gesture classes based on combined finger state and motion analysis:

Table -1

Gesture Output	Hand Configuration	Motion
YES	Fist	Up-Down
SORRY	Fist	Circular
RESTROOM	Fist	None
GOODBYE	Open Palm	None
PLEASE	Open Palm	Circular
NO	Index + Middle	None
I LOVE YOU	Thumb + Index + Pinky	None
GO	Index finger (horizontal)	None
OK / EAT / QUESTION	Pinch (< 0.05)	None
HURTS A LOT	Pinch (< 0.08)	None

3.6 Temporal Smoothing and Prediction Stability

To suppress transient misclassifications caused by hand transition movements between gestures, the system employs a majority voting mechanism over a sliding buffer of five consecutive frames. A gesture is confirmed and passed to the output stage only when all five buffer entries contain the same prediction. This significantly reduces flickering outputs and improves temporal consistency.

```
python
BUFFER_SIZE = 5
if len(gesture_buffer) == BUFFER_SIZE and \
  all(g == gesture_buffer[0] for g in gesture_buffer):
   stable_gesture = gesture_buffer[0]
```

3.7 Text Buffering and Speech Synthesis

Confirmed gesture predictions are stored in a text buffer. A cooldown mechanism of 1.2 seconds between successive speech outputs prevents repeated announcements of the same gesture. The finalized text is passed to the pyttsx3 text-to-speech engine, which synthesizes audible speech at a rate of 150 words per minute. The TTS operation is executed on a separate daemon thread to ensure the main video processing loop is not blocked, maintaining real-time performance.

```
python
threading.Thread(target=speak,args=(message,),
daemon=True).start()
```

3.8 Bidirectional Communication Module (Novelty)

As the primary contribution of this work, a reverse communication channel is incorporated into the system. A hearing individual can speak naturally into a microphone, and the system employs an Automatic Speech Recognition (ASR) engine — such as the Google Speech Recognition API via the Speech Recognition library or an offline engine such as Vosk — to transcribe the spoken input into text in real time. The transcribed text is displayed on the TFT screen connected to the system, enabling the deaf or hard-of-hearing user to read and comprehend the hearing individual's response without any external assistance. This closes the communication loop and transforms the system from a one-directional assistive tool into a fully self-sufficient bidirectional communication platform.

The bidirectional architecture is illustrated in Figure 2.

3.9 System Integration and Real-Time Performance

The complete pipeline — from frame capture to speech output — operates at approximately 25 frames per second on an Intel i3 second-generation processor with 4GB DDR3 RAM, without requiring any GPU acceleration. The use of landmark-based features rather than raw image CNN processing significantly reduces computational complexity, making the system accessible on low-cost, widely available hardware.

4. HARDWARE AND SOFTWARE COMPONENTS

The proposed system is implemented on a desktop computing platform assembled using standard,

commercially available hardware components. The design prioritizes cost-effectiveness and accessibility, ensuring that the system can be replicated without specialized or high-end equipment. This section describes the hardware and software components used in the implementation.

4.1 Hardware Components

4.1.1 Motherboard — Intel H61 Express Chipset

The system is built on an H61 chipset-based motherboard supporting the LGA1155 socket. The chipset operates at a core voltage of approximately 1.05V with an I/O voltage of 3.3V. The motherboard receives power through a standard 24-pin ATX connector supplying +3.3V, +5V, and +12V rails, along with a 4-pin CPU power connector supplying 12V to the processor voltage regulator module (VRM). The integrated Intel HD Graphics output is used for display purposes, eliminating the need for a dedicated GPU.

4.1.2 Processor — Intel Core i3 (2nd Generation, LGA1155)

The central processing unit is an Intel Core i3 second-generation processor mounted on the LGA1155 socket. The processor operates with a dynamic core voltage (Vcore) ranging from 0.7V to 1.3V, with a thermal design power (TDP) of 55W to 77W. All computation including video processing, landmark detection, feature extraction, and machine learning inference is performed entirely on this CPU, demonstrating the system's ability to function without GPU dependency.

4.1.3 RAM — DDR3

The system utilizes DDR3 SDRAM operating at a standard voltage of 1.5V (or 1.35V for DDR3L low-voltage variants), with support for frequencies of 1333 MHz and 1600 MHz. The memory provides sufficient bandwidth for parallel processing of video frames alongside the classification and speech synthesis pipeline.

4.1.4 USB Webcam

A standard USB webcam serves as the primary image sensor for hand gesture acquisition. The webcam is powered entirely through the USB 2.0 interface at 5V, drawing between 100mA and 500mA of current. It captures continuous video frames that are fed into the OpenCV processing pipeline at approximately 25 frames per second.

4.1.5 TFT Display

A TFT display is used for real-time text output, showing the recognized gesture label on screen during operation. The display operates at a logic voltage of 3.3V with a backlight voltage ranging from 3.3V to 5V. In the context of the bidirectional communication module, the TFT display also serves as the output interface for transcribed speech text received from the hearing individual, enabling the deaf or hard-of-hearing user to read responses in real time.

4.1.6 SMPS Power Supply

A standard ATX Switch Mode Power Supply (SMPS) of recommended capacity 450W powers the entire system. The SMPS delivers the following regulated voltage rails:

Table -2

Rail	Powered Components
+12V	CPU VRM, Cooling Fans
+5V	USB Ports, SATA Devices
+3.3V	Chipset, RAM Logic
Ground	Common Reference

4.1.7 Audio System

The onboard sound card integrated within the motherboard chipset handles audio output. It delivers a line output voltage of approximately 1V RMS through a 3.5mm audio jack, with a Digital-to-Analog Converter (DAC) built into the chipset. External speakers connected via the 3.5mm jack produce the synthesized speech output generated by the pyttsx3 TTS engine. For the bidirectional communication module, a microphone connected to the audio input jack captures speech from the hearing individual for processing by the ASR engine.

4.1.8 Peripheral Components and Enclosure

Additional peripheral components include a keyboard and mouse for system interaction, standard power cables, and an internet connection for optional cloud-based speech recognition. The entire hardware assembly is housed within an acrylic casing, providing structural protection and a clean form factor suitable for demonstration and deployment.

4.1.9 Total System Power Consumption

The estimated power consumption of the assembled system is as follows:

Table -3

Component	**Power Consumption**
CPU (Intel i3)	55W – 77W
Motherboard	20W – 30W
DDR3 RAM	5W – 8W
USB Webcam	2W – 3W
Peripherals	~5W
Total (Estimated)	**~100W – 130W**

4.2 Software Components

The system software stack is built entirely using open-source Python libraries, ensuring zero licensing cost and broad community support.

4.2.1 Python Python 3.x serves as the primary programming language for the entire system due to its extensive support for computer vision, machine learning, and speech processing libraries.

4.2.2 OpenCV OpenCV (Open Source Computer Vision Library) handles real-time video capture, frame extraction, color space conversion, and on-screen text rendering. It forms the backbone of the image acquisition and display pipeline.

4.2.3 MediaPipe Google's MediaPipe Hands framework performs real-time 21-point hand landmark detection from live video frames. It provides highly optimized, cross-platform landmark tracking suitable for CPU-only execution.

4.2.4 pyttsx3 The pyttsx3 library provides offline text-to-speech synthesis, converting recognized gesture labels into audible speech output without requiring an internet connection. Speech is generated on a separate daemon thread to maintain real-time processing performance.

4.2.5 SpeechRecognition / Vosk For the bidirectional communication module, the SpeechRecognition library interfaced with the Google Speech Recognition API, or alternatively the Vosk offline ASR engine, is used to transcribe spoken input from the hearing individual into displayable text.

4.2.6 Threading Python's built-in threading module is used to run the TTS engine and ASR module as concurrent daemon threads, preventing blocking of the main video processing loop and maintaining system responsiveness.

Table -4

Software Component	Purpose
Python 3.x	Core programming language
OpenCV	Video capture and display
MediaPipe	Hand landmark detection
pyttsx3	Text-to-speech synthesis
Speech Recognition / Vosk	Automatic speech recognition
Threading	Concurrent processing

5. RESULTS AND DISCUSSION

5.1 Experimental Setup

The proposed system was evaluated on a CPU-based desktop configuration consisting of an Intel Core i3 (2nd Generation) processor with 4GB DDR3 RAM. A standard USB webcam was used for real-time video acquisition, and no GPU acceleration was employed. The system was tested under normal indoor lighting conditions to simulate real-world usage.

A custom dataset of American Sign Language (ASL) gestures was created using live webcam recordings. The dataset included multiple samples for each gesture performed by different users to ensure variability in hand size, orientation, and speed of execution.

5.2 Performance Metrics

The system performance was evaluated using the following metrics:

- Classification Accuracy
- Real-Time Frame Rate (FPS)
- Prediction Stability
- Response Latency

5.3 Gesture Recognition Accuracy

Three machine learning models were evaluated for classification: Multi-Layer Perceptron (MLP), Support Vector Machine (SVM), and Random Forest.

Table -5

Model	Accuracy (%)
MLP	92.8
SVM	91.5
Random Forest	94.2

The Random Forest classifier achieved the highest accuracy due to its robustness in handling non-linear feature distributions and reduced sensitivity to noise in landmark data.

5.4 Effect of Temporal Smoothing

To improve prediction stability, a majority voting mechanism over a buffer of five consecutive frames was implemented.

Table -6

Condition	Stability Improvement
Without Smoothing	Frequent prediction flickering
With Smoothing	~30% reduction in misclassification transitions

Temporal smoothing significantly reduced transient errors caused by intermediate hand movements during gesture transitions, resulting in more consistent outputs.

5.5 Real-Time Performance

The system achieved an average processing speed of:

- ~25 Frames Per Second (FPS) on CPU
- Latency: ~200–300 ms for gesture-to-speech output

This demonstrates that the system is capable of real-time performance without requiring dedicated GPU hardware.

5.6 Bidirectional Communication Evaluation

The speech-to-text module was tested using both online (Google Speech Recognition) and offline (Vosk) engines.

Table -7

Module	Performance
Google ASR	Higher accuracy, requires internet
Vosk (Offline)	Slightly lower accuracy, works without internet

The bidirectional module successfully enabled real-time interaction, where spoken input from a hearing individual was converted into readable text for the user, completing the communication loop.

5.7 System Robustness

The system was tested under different conditions:

- Lighting Variations: Minor impact due to landmark-based detection
- Hand Orientation: Hand normalization improved invariance
- User Variation: Consistent performance across different users

However, slight accuracy degradation was observed in:

- Extremely low lighting conditions
- Fast or incomplete gesture execution

5.8 Comparative Analysis

- Compared to deep learning-based CNN models, the proposed system:
- Requires significantly lower computational resources
- Operates on CPU-only systems
- Maintains competitive accuracy (>90%)
- Provides real-time bidirectional communication, which is absent in most existing systems

5.9 Discussion

The experimental results demonstrate that the proposed system achieves a balance between accuracy, efficiency, and practicality. The use of MediaPipe-based landmark extraction combined with lightweight machine learning classifiers enables real-time gesture recognition without the need for high-end hardware.

The integration of a bidirectional communication module represents a key advancement over existing systems, addressing a critical gap in assistive communication technology. While deep learning approaches may achieve marginally higher accuracy, they often require GPU acceleration and are less suitable for low-cost deployment.

6. CONCLUSION

This paper presented a real-time Sign Language to Speech Conversion system designed to bridge the communication gap between hearing-impaired individuals and the general public. The system utilizes computer vision and machine learning techniques, leveraging MediaPipe-based hand landmark detection and lightweight classifiers such as Multi-Layer Perceptron, Support Vector Machine, and Random Forest for accurate gesture recognition. The integration of temporal smoothing ensures stable predictions, while the use of an offline text-to-speech engine enables real-time auditory output without internet dependency.

A key contribution of this work is the implementation of a bidirectional communication framework, which allows not only the translation of sign language into speech but also the conversion of spoken input into text for the hearing-impaired user. This two-way interaction addresses a significant limitation present in most existing systems and enhances practical usability in real-world scenarios.

Experimental results demonstrate that the system achieves high recognition accuracy (above 90%) while maintaining real-time performance of approximately 25 frames per second on a standard CPU-based setup without requiring GPU acceleration. The use of landmark-based feature extraction significantly reduces computational complexity, making the system cost-effective and accessible.

In addition, the proposed system offers strong potential for hardware miniaturization and portability. The complete pipeline can be deployed on embedded platforms such as the Raspberry Pi 4 Model B or similar single-board computers, enabling the development of compact, portable assistive devices. This opens the possibility for wearable or standalone communication systems that can be used in everyday environments without reliance on bulky desktop hardware

REFERENCES

[1] Barathi Subramanian et al., "An integrated mediapipe-optimized GRU model for Indian sign language recognition," Scientific Reports, 2022. DOI: 10.1038/s41598-022-15998-7

[2] Singh et al., "Continuous Sign Language Recognition System using Deep Learning with MediaPipe Holistic," arXiv:2411.04517, 2024.

[3] Kumar Navendu & Vineet Sahula, "Word Level Sign Language Recognition using MediaPipe and LSTM-GRU Network," TechRxiv, 2024. DOI: 10.36227/techrxiv.172054945

[4] "Mediapipe and CNNs for Real-Time ASL Gesture Recognition," arXiv:2305.05296, 2023.

[5] Dagde et al., "A hand sign recognition based signal system for mute people using machine learning," MethodsX, 2025. DOI: 10.1016/j.mex.2025.103670

[6] "KD-MSLRT: Lightweight Sign Language Recognition Model Based on MediaPipe," arXiv:2501.02321, 2025.

[7] "Enhancing Bidirectional Sign Language Communication: Integrating YOLOv8 and NLP," arXiv:2411.13597, 2024.

[8] Marcolino et al., "Innovations in Deaf Health Care Communication: Systematic Review," JMIR, 2026.

BIOGRAPHIES

Final Year B tech student in the Electronics and communication Engineering Department at AWH Engineering College Calicut Kerala India.

Final Year B tech student in the Electronics and communication Engineering Department at AWH Engineering College Calicut Kerala India.

Final Year B tech student in the Electronics and communication Engineering Department at AWH Engineering College Calicut Kerala India.

Final Year B tech student in the Electronics and communication Engineering Department at AWH Engineering College Calicut Kerala India.

Amrutha m completed her B tech in Kannur University. Completed her master's in Calicut University. Pursuing PhD in PhD anna university. Her research interest includes , Image processing, Image security ,chaotic cryptography.