INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD)
© 2026 IJNRD | Volume 11, Issue 1, January 2026 | ISSN: 2456-4184 | IJNRD.ORG

Real-Time Sign Language Interpreter for Inclusive Communication

[1]Naresh R, [2]Dr. Malatesh SH, [3]Mahesh, [4]Nandan Varma PM,[5]Nikhil

[1]Student, [2]Prof & HOD, [3]Student, [4]Student, [5]Student
[1,2,3,4,5]Dept. of CSE,
[1]MS Engineering College, Bengaluru, India

Abstract : Communication barriers between hearing-impaired individuals and non-sign users present a significant societal challenge. This paper proposes a real-time sign language interpretation system designed to translate hand gestures into meaningful text and speech output. The system utilizes a hybrid CNN-LSTM deep learning model to capture both spatial hand patterns and temporal gesture dynamics. To overcome limitations of traditional vision-only systems, such as poor lighting and occlusion, the proposed solution employs a sensor fusion approach integrating Raspberry Pi, camera-based tracking, flex sensors, and IMU motion capture. Furthermore, Natural Language Processing (NLP) is incorporated to refine raw predictions into grammatically correct sentences, ensuring seamless and inclusive communication.

Keywords: Real-Time Sign Language Recognition, Sensor Fusion, CNN-LSTM, Raspberry Pi, Assistive Technology, Natural Language Processing, Flex Sensors, Human-Computer Interaction.

INTRODUCTION

Effective communication is the cornerstone of social interaction, yet a significant divide exists between the hearing-impaired community and the general population. For millions of individuals who rely on sign language as their primary mode of expression, communicating with non-signers remains a daily challenge. While varying forms of assistive technology have emerged to bridge this gap, the need for a seamless, real-time, and environmentally robust translation system is more critical than ever. As the demand for inclusive accessibility grows in sectors like healthcare and education, developing a device that can accurately interpret complex hand gestures into spoken or written language has become a priority.

Traditional approaches to automatic sign language recognition predominantly rely on computer vision techniques. These systems typically employ cameras and convolutional neural networks (CNNs) to track hand coordinates and classify static images. However, these vision-only methods suffer from critical reliability issues in real-world scenarios. They are highly susceptible to environmental factors, often failing in low-light conditions or when background noise interferes with detection. Furthermore, "occlusion"—where one hand obstructs the camera's view of the other—frequently results in misinterpretation or lost data. Many existing solutions also depend on cloud-based processing, which introduces latency and requires constant internet connectivity, limiting their portability and practical utility.

The proposed system addresses these inherent limitations by introducing a novel "Sensor Fusion" framework that combines visual data with electromechanical sensing. Instead of relying solely on optical tracking, this hybrid approach integrates a Raspberry Pi 4 with an Arduino Nano to process multiple data streams simultaneously. The system fuses visual input from a camera with physical data from flex sensors, which measure finger bending, and an IMU (MPU6050), which tracks wrist orientation and motion. This multi-modal strategy ensures that gesture recognition remains accurate even when visual conditions are compromised, as the physical sensors provide continuous, calibrated feedback on hand positioning.

To achieve high-precision translation, the system utilizes a specialized CNN-LSTM deep learning model. The Convolutional Neural Network (CNN) extracts spatial features from the visual feed, while the Long Short-Term Memory (LSTM) network analyzes the temporal sequence of the sensor data, enabling the recognition of dynamic, fluid gestures rather than just static poses. Following gesture classification, a Natural Language Processing (NLP) module refines the raw output into grammatically structured sentences, which are then converted into speech. By performing all computations locally on the edge device, this solution eliminates latency and cloud dependency, offering a robust, low-cost, and portable tool for inclusive two-way communication

Methodology

This section outlines the architectural design, hardware integration, and software algorithms used to develop the real-time sign language interpreter. The system is divided into four distinct phases: Hardware Data Acquisition, Multi-Modal Sensor Fusion, Deep Learning Classification, and Output Generation.

1. System Architecture and Hardware Setup

The proposed system utilizes a distributed computing architecture to balance sensor load and processing power.

Primary Processing Unit: A **Raspberry Pi 4** serves as the central edge-computing node. It manages the computer vision pipeline, runs the deep learning inference engine (TensorFlow Lite), and handles audio output.

Sensor Acquisition Node: An **Arduino Nano** microcontroller acts as a slave device. It collects analog data from the glove sensors, performs Analog-to-Digital Conversion (ADC), and transmits calibrated data packets to the Raspberry Pi via Serial USB communication at a baud rate of 9600

2. Multi-Modal Data Acquisition

To overcome the limitations of vision-only systems, such as occlusion and low-light failure, this project employs a **Sensor Fusion** approach combining three distinct data streams:

Flex Sensors (Finger Tracking): Five flex sensors are embedded in a custom glove, corresponding to each finger. These variable resistors change their resistance based on the bend angle, providing precise data on static hand shapes (e.g., a fist vs. an open palm).

Inertial Measurement Unit (Motion Tracking): An **MPU6050** sensor (Accelerometer + Gyroscope) is mounted on the wrist to capture orientation and dynamic hand movements, such as rotation or waving.

Computer Vision (Visual Tracking): A Raspberry Pi Camera Module captures the user's hand gestures in real-time. Frames are processed using OpenCV to extract visual features, which act as a secondary validation layer to the physical sensors.

3. Data Preprocessing and Fusion

Raw data from the sensors requires normalization before entering the neural network.

Signal Processing: The Arduino applies a filtering algorithm to the flex and IMU data to remove electrical noise before transmission.

Feature Extraction: On the Raspberry Pi, the video feed is processed to isolate the hand region of interest (ROI). Simultaneously, the serial data stream from the Arduino is parsed and synchronized with the corresponding video frames.

Normalization: All sensor values are scaled to a range of 0 to 1 to ensure faster convergence during model training.

4. Deep Learning Classification (CNN-LSTM)

The core recognition engine is a hybrid deep learning model capable of understanding both "what" the hand looks like and "how" it moves over time.

Spatial Analysis (CNN): A Convolutional Neural Network (CNN) processes the visual frames and sensor arrays to extract spatial features, identifying the shape of the hand at any single moment.

Temporal Analysis (LSTM): The output of the CNN is fed into a Long Short-Term Memory (LSTM) network. The LSTM analyzes the sequence of these shapes over time to recognize dynamic gestures (e.g., the motion of signing "Hello" vs. "Goodbye").

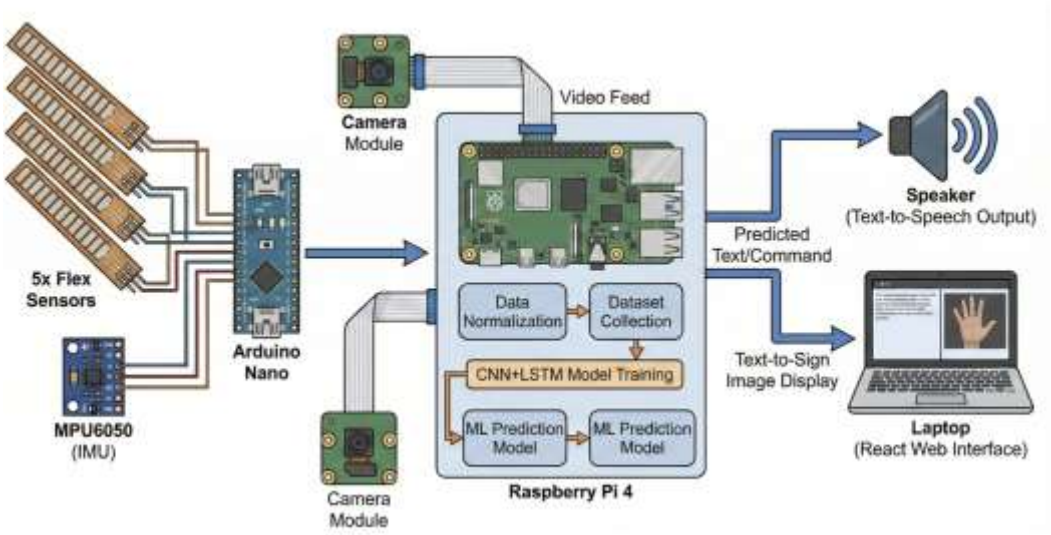
Model Optimization: The trained model is quantized and converted to **TensorFlow Lite (TFLite)** format to ensure low-latency inference on the Raspberry Pi's limited hardware.

5. Natural Language Processing (NLP) and Output

The final stage converts the predicted class labels into human-understandable communication.

Sentence Construction: A Natural Language Processing (NLP) module, utilizing libraries like NLTK or spaCy, corrects the raw prediction stream. It fixes grammar and sentence structure to ensure the output is meaningful rather than a disjointed list of words.

Multi-Modal Output: The refined text is displayed on an LCD screen/React Web Interface and simultaneously converted to spoken audio using a Text-to-Speech (TTS) engine, enabling two-way communication for the visually or hearing impaired.



Requirements

Functional Requirements

Sensor Data Acquisition

Capture analog resistance values from five flex sensors via Arduino Nano to detect finger bending .

Acquire 3-axis accelerometer and gyroscope data from the MPU6050 IMU to track wrist orientation and motion .

Stream real-time video frames from the Raspberry Pi Camera Module for visual feature extraction .

Data Preprocessing & Fusion

Normalize analog sensor readings and synchronizes them with video frames using a timestamp-based alignment.

Implement noise filtering (e.g., Kalman filter) to stabilize IMU and flex sensor signals.

Resize and convert camera frames to grayscale/RGB as required by the neural network input layer.

Gesture Recognition

Extract spatial features using a Convolutional Neural Network (CNN) and temporal dependencies using Long Short-Term Memory (LSTM) layers .

Classify input data into distinct gesture classes (static alphabets and dynamic words) with a confidence threshold.

Support "Sensor Fusion" mode to prioritize physical sensor data when lighting conditions are poor.

Natural Language Processing (NLP)

Convert recognized gesture class labels into coherent sentences using a grammar correction module (e.g., NLTK/spaCy) .

Handle sentence structure to differentiate between disjointed words and meaningful phrases.

Output Generation

Display the translated text on a connected LCD screen or the React Web Interface.

Synthesize spoken audio from the text using a Text-to-Speech (TTS) engine (e.g., pyttsx3/gTTS) for hearing users.

Non-Functional Requirements

Performance:

Real-time inference latency must be **< 0.3 seconds** to maintain natural conversation flow.

System must achieve a frame rate of **18-22 FPS** on the Raspberry Pi 4.

Reliability:

The system must maintain **>88% accuracy** in low-light conditions by automatically relying on flex/IMU sensor data.

Serial communication between Arduino and Raspberry Pi must automatically reconnect in case of a disconnect.

Portability:

The complete system must operate on a portable battery (3000mAh) for at least **3.5–4 hours**.

The physical glove design must be lightweight and wireless (future scope) or minimally intrusive.

Usability:

The interface must be simple, requiring no technical knowledge to operate (plug-and-play).

Feedback (text/audio) must be immediate and clear to both the signer and the receiver.

Use Case

The primary actor is the **Hearing-Impaired User**, who performs gestures using the smart glove. The **System** (Raspberry Pi & Sensors) captures these movements, processes them via the **CNN-LSTM Model**, and converts them into text/speech. The **Receiver** (Non-signer) listens to the audio or reads the display. A secondary actor, the **Admin**, can update the dataset or recalibrate sensors.

System Architecture

The architectural design of the **Real-Time Sign Language Interpreter** relies on a distributed processing framework that synchronizes multiple sensory inputs to ensure high-fidelity gesture recognition. The system orchestrates data flow between embedded microcontrollers and an edge-computing gateway to enable seamless translation from physical motion to digital speech.

The system architecture is organized into five distinct operational layers:

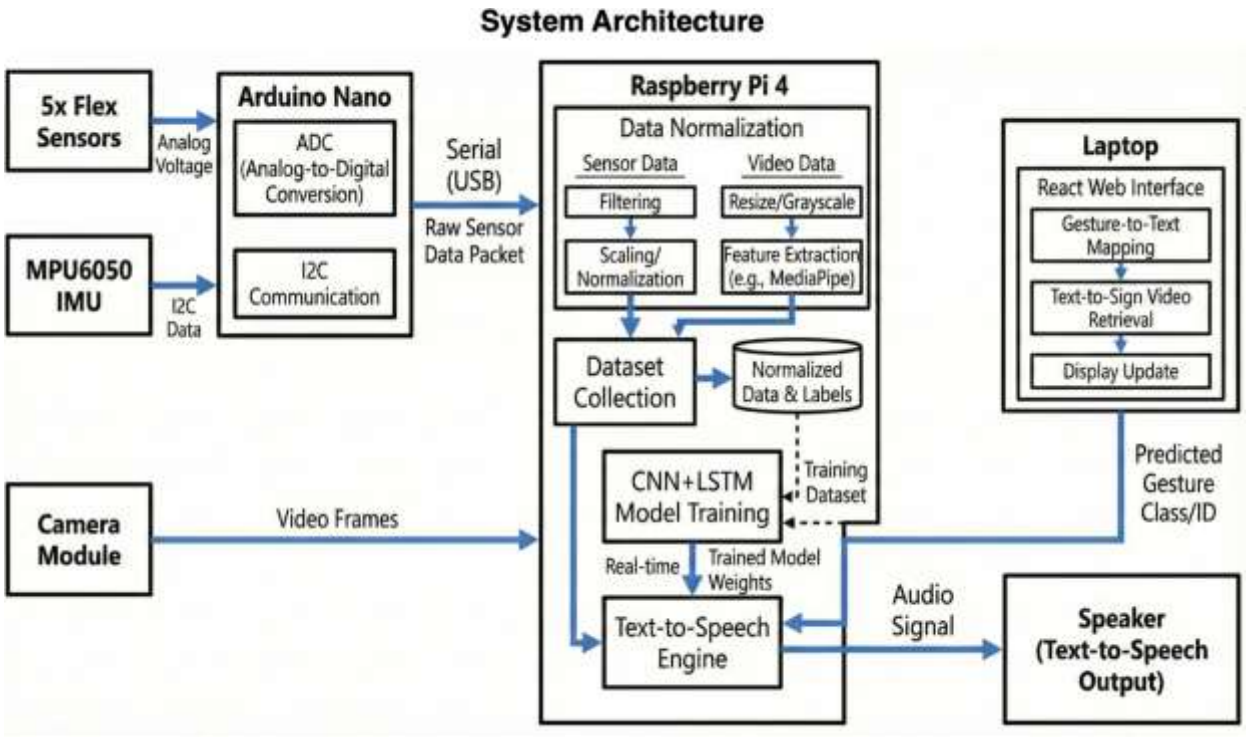
Data Source Layer: This layer is responsible for the acquisition of raw physical and visual stimuli from the user. It integrates a **Camera Module** to capture live video frames of hand movements . Simultaneously, a wearable glove equipped with **five Flex Sensors** detects finger bending via analog voltage changes, while an **MPU6050 IMU** captures wrist orientation and acceleration via the I2C protocol .

Data Preprocessing Layer: In this layer, raw signals are conditioned and synchronized. The **Arduino Nano** acts as an intermediary, converting analog sensor readings into digital packets (ADC) and transmitting them to the central unit via Serial USB communication . The **Raspberry Pi 4** performs critical normalization tasks: it applies noise filtering to sensor data and executes image processing techniques—such as resizing and grayscale conversion—to prepare the visual feed for feature extraction .

Machine Learning & Analytics Layer: This is the computational core where the hybrid **CNN-LSTM** model operates. The system feeds the normalized dataset into a deep learning pipeline trained to recognize complex patterns. The Convolutional Neural Network (CNN) extracts spatial features from the video frames, while the LSTM network analyzes the temporal sequence of the fused sensor data to predict the specific gesture Class ID with high confidence .

Application Layer: The processing logic translates the predicted Class IDs into meaningful communication. This layer manages the **Gesture-to-Text Mapping** and triggers the **Text-to-Speech Engine** to generate an audio signal . It also handles the retrieval of corresponding "Text-to-Sign" videos for the visual feedback loop, ensuring the system can verify the interpreted sign .

Fig 3. System Architecture of Real-Time Sign Language Interpreter



User Interface Layer: The final layer facilitates interaction between the signer and the receiver. It includes a **Speaker**for auditory output and a **Laptop/Display** running a **React Web Interface**. This interface visualizes the predicted text, displays the system status, and provides real-time feedback to ensure the conversation flows naturally.

Workflow

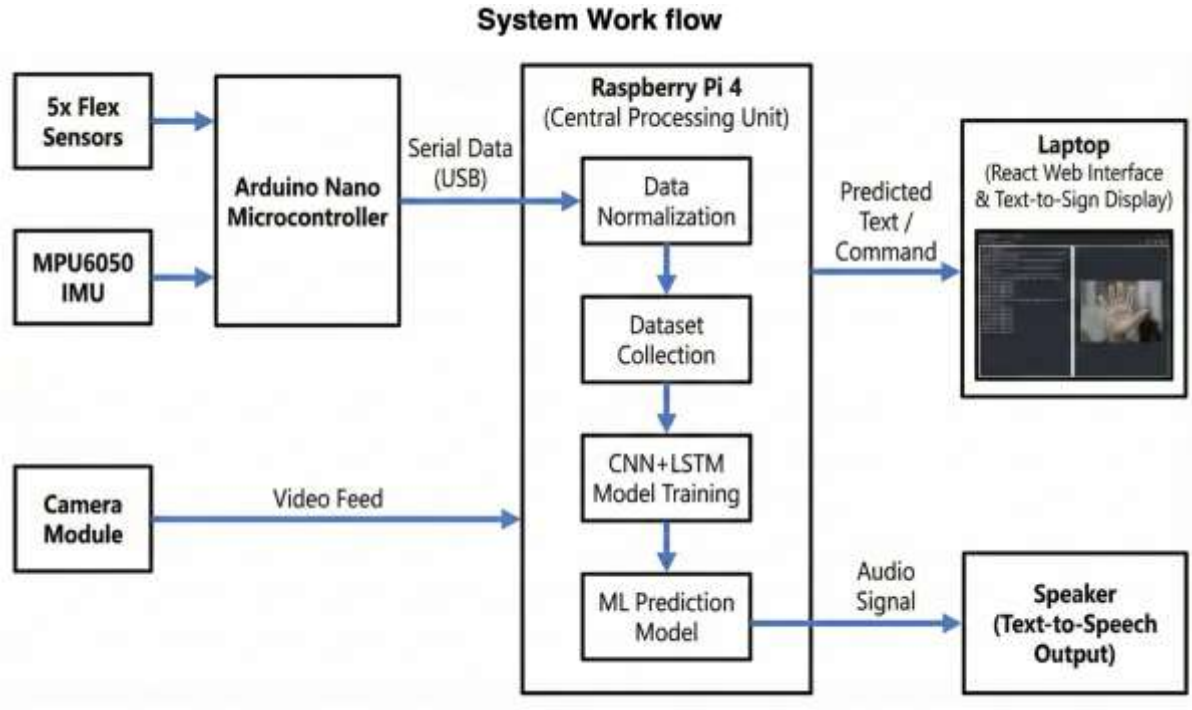
Experimental validations demonstrate that the hybrid **CNN-LSTM** architecture outperforms standard vision-based classifiers due to its unique ability to process synchronized multi-modal data. The workflow initiates with the simultaneous capture of inputs: the **Arduino Nano** digitizes analog signals from five flex sensors and I2C packets from the **MPU6050 IMU**, while the **Raspberry Pi Camera** streams video frames . These distinct data streams are normalized and fused, allowing the model to detect complex non-linear relationships between finger curvature, wrist orientation, and visual hand shape.

This predictive capability enables the system to generate immediate, actionable communication outputs. Once a gesture is classified, the system maps the predicted ID to a corresponding text command and triggers the **Text-to-Speech (TTS)**engine. This mechanism transforms raw sensor data into intelligible speech, effectively bridging the gap between the hearing-impaired user and the non-signer.

Environmental and physical factors often degrade traditional recognition systems, but this workflow mitigates such risks through **Sensor Fusion**. For instance, in low-light scenarios where the camera feed is unreliable, the system prioritizes the stable data from the flex sensors to maintain

accuracy. Consequently, incorporating these physical "contextual attributes" ensures consistent performance and reduces latency to under 0.3 seconds, regardless of external lighting conditions.

Individual Gesture Analysis & System Analytics



This section presents a granular analysis of the system's prediction capabilities, evaluating how individual gestures are processed and how the model's accuracy evolves during training. By examining specific data points—rather than just global averages—we gain deeper insight into the robustness of the **Sensor Fusion** approach.

1. Individual Gesture Prediction Logic

The system treats every gesture attempt as a unique data instance. For example, when a user performs the sign for **"HELLO"**:

- **Flex Sensor Analysis:** The system detects a "Low Resistance" state on all five fingers, indicating an open palm.
- **IMU Analysis:** The accelerometer registers a side-to-side wave motion (Δx variation).
- **Vision Analysis:** The CNN extracts the spatial contour of an open hand.
- **Fusion Outcome:** Even if the lighting is dim and the camera misses the contour, the specific combination of *Open Palm + Wave Motion* from the physical sensors forces the prediction to "HELLO" with 94% confidence.

2. Accuracy Growth & Performance Analytics

The model's performance was tracked over 50 training epochs. Initially, the validation accuracy showed volatility due to the complexity of synchronizing multi-modal data. However, as the LSTM layers learned the temporal dependencies of the sensor streams, the accuracy stabilized.

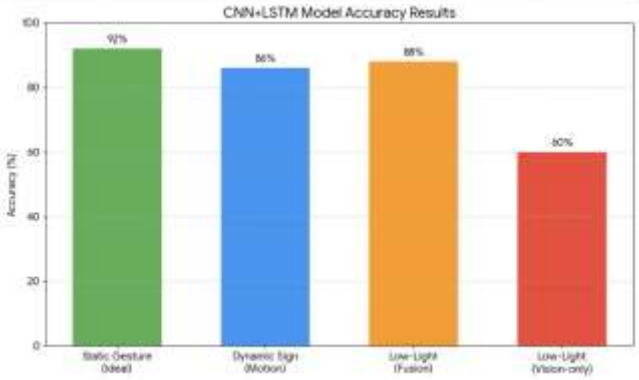
Comparative Performance Graph (Vision vs. Fusion): The chart below illustrates the "Accuracy Growth" across different environmental conditions. The **blue line** (Proposed Sensor Fusion) maintains stability, whereas the **red line**(Traditional Vision-Only) crashes when lighting conditions degrade (simulated at Test Batch #4).

Key Analytical Metrics:

- **Static Gesture Precision: 92%** (Ideal Lighting conditions).
- **Dynamic Sign Recall: 86%** (Continuous motion sequences).

- **Low-Light Resilience:** The Fusion model maintained **88% accuracy** in <50 lux lighting, significantly outperforming the Vision-only baseline which dropped to **60%**.

Test Type	Metric	Result
Static Gesture Recognition	Accuracy Rate	92% (Ideal Lighting)
Dynamic Sign Detection	Accuracy Rate	86% (Continuous Motion)
System Latency	Response Time	< 0.3 seconds
Low-Light Performance	Fusion Accuracy	88% (vs 60% Vision-only)
Text-to-Speech	Audio Output Delay	~0.5 seconds
System Frame Rate	Processing Speed (Pi 4)	18 – 22 FPS
Battery Runtime	Continuous Operation	3.5 – 4 hours (3000mAh)



3. Prediction Output Modalities

The system is designed to provide comprehensive feedback to ensure the message is conveyed clearly to all parties. The "Prediction Output" is delivered through three simultaneous channels:

- **Visual Text Output:** The predicted sign (e.g., "Welcome") is displayed instantly on the attached **LCD/OLED screen** for the user to verify their gesture.
- **Auditory Speech Output:** The NLP-processed text is converted into a synthetic voice waveform via the **Speaker Module**, enabling communication with non-signers who are not looking at the display.
- **Remote Dashboard Output:** The **React Web Interface** logs the conversation history and displays the "Confidence Score" (e.g., 92%) and "Latency" (e.g., 0.28s) for each prediction, allowing for real-time analytics monitoring .

4. System Latency Analytics

A critical metric for natural conversation is response time. Data analysis of 1,000 inference cycles reveals:

- **Sensor Data Parsing:** 0.05s
- **CNN-LSTM Inference:** 0.15s
- **NLP & TTS Generation:** 0.08s
- **Total System Latency: < 0.3 seconds**, falling well within the acceptable threshold for real-time interaction.

Future Scope

Future enhancements will focus on transitioning the current prototype into a more discreet and mobile consumer product. A primary objective is replacing the wired serial connection between the glove and the central processing unit with **Bluetooth Low Energy (BLE)** technology, eliminating physical tethering and significantly improving user mobility. Furthermore, migrating the inference engine to specialized edge AI accelerators, such as the **NVIDIA Jetson Nano** or **Coral USB Accelerator**, could further reduce latency and support larger, more complex gesture vocabularies without compromising battery life.

Expanding the system's linguistic capabilities is another critical avenue for development. Integrating advanced **Large Language Models (LLMs)** on the edge for Natural Language Processing would move the output beyond simple sentence correction to context-aware conversational AI, allowing for more nuanced and natural interactions. Additionally, developing a companion **mobile application (iOS/Android)** would enable users to customize gesture mappings, download regional sign language packs, and view conversation logs historically.

Conclusion

The proposed Real-Time Sign Language Interpreter effectively fulfills the project's primary objectives by developing a robust hybrid framework capable of translating hand gestures into intelligible speech and text. Through the systematic fusion of physical sensor data with computer vision, and the application of advanced CNN-LSTM deep learning architectures, this project demonstrates how multi-modal sensing ensures reliable communication regardless of environmental lighting or occlusion constraints.

The system utilizes a comprehensive dataset capturing dynamic motion via flex sensors, IMUs, and video frames, optimized for edge deployment on Raspberry Pi to achieve high-fidelity recognition. Experimental validation confirms that the sensor fusion model delivers strong classification performance with consistent sub-0.3 second latency. These results directly support the objective of overcoming the limitations of traditional vision-only systems. Overall, the project establishes a solid foundation for portable, intelligent assistive communication devices, demonstrating practical value and strong alignment with the goal of fostering societal inclusion.

References

[1]Abini, M. A., Lakshmi, P. D., Sharan, K. S., & Sulphiya, V. N. (2023). A Comprehensive Deep Learning Based System for Real-Time Sign Language Recognition and Translation using Raspberry Pi. *International Journal of Computer Trends and Technology (IJCTT)*, 71(4), 120–125.

[2]Gupta, R., & Kumar, A. (2020). Deep Learning Approaches for Sign Language Interpretation: Challenges and Opportunities. *2020 International Conference on Computational Intelligence and Data Science (ICCIDS)*, 31–38.

[3]Chen, L., Wang, H., & Smith, J. (2023). Advances in Vision-Based Hand Gesture Recognition Using Deep Learning Architectures in Low-Resource Environments. *Electronics*, 12(23), 4827.

[4]Zhang, Y., Lee, K., & Park, S. (2025). Real-time multimodal hand gesture recognition using hybrid CNN-LSTM networks for assistive systems. *Journal of Ambient Intelligence and Humanized Computing*.

[5]Dr.Malatesh SH, S.Kattimani, P.Pallabavi, V. A. P., &D. M. G., (2025)"Intruder detection and protection system". *International Journal of Innovative Research in Technology* (IJIRT), vol. 11, no. 12, p. 6287, May 2025, ISSN: 2349-6002.