IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

VisionSpeak: Real-Time Sign Language to Speech Translation Using Computer Vision and Deep Learning.

AI-Driven Gesture Recognition Supporting American Sign Language (ASL) and Indian Sign Language (ISL)

[1]Mr. P. Ranjith, [2]J. Janani, [3]S. Gnyanodhaya, [4]S. Manasaa

[1]Assistant Professor (Guide), [2]Student, [3]Student, [4]Student

Department of Computer Science and Business Systems

K. Ramakrishnan College of Engineering, Tiruchirappalli, Tamil Nadu, India

Abstract: Sign language is the primary communication mode for millions of deaf and hearing-impaired individuals, yet a critical communication barrier persists between signers and non-signers worldwide. Existing translation systems often depend on costly wearable sensors or support only limited vocabularies. To address this, we propose **VisionSpeak**, a real-time bilingual sign language-to-speech translation system that uses only a standard webcam and computer vision — supporting both American Sign Language (ASL) and Indian Sign Language (ISL). The system processes video frames using Google's MediaPipe Hands framework to extract 21 hand landmarks per hand (42 total), normalizes coordinates relative to the wrist landmark for scale and position invariance, and feeds the resulting 84-dimensional feature vector into a TensorFlow feedforward neural network for gesture classification. Recognized gestures are converted to audible speech using a pyttsx3 text-to-speech engine running on a separate thread. The model is trained on a real labeled dataset of **11,772 samples** across **75 gesture classes** (29 ASL + 46 ISL), extracted from image folders using MediaPipe. Our end-to-end implementation achieves a gesture recognition accuracy of approximately **95%** on test data, with total end-to-end latency of approximately **115 ms** per gesture, enabling smooth real-time bilingual translation. A stability threshold mechanism prevents false positives by requiring consistent prediction across 15 consecutive frames before committing a gesture. VisionSpeak is fully software-driven, requires no specialized hardware, and is portable and cost-effective for bridging communication gaps between deaf and hearing communities.

Index Terms - Sign Language Recognition, American Sign Language (ASL), Indian Sign Language (ISL), Computer Vision, Deep Learning, TensorFlow, MediaPipe, Gesture Recognition, Real-Time Translation, Assistive Technology, Text-to-Speech, Human-Computer Interaction, Bilingual Translation.

I. INTRODUCTION

According to the World Health Organization (WHO), over 1.5 billion people worldwide experience some degree of hearing loss, with approximately 430 million suffering from disabling hearing loss. Globally, an estimated 70 million deaf and hard-of-hearing individuals rely on sign languages as their primary mode of communication. However, a critical barrier persists: the vast majority of the hearing population does not understand sign language, creating social exclusion in education, healthcare, and employment contexts.

Vision-based Sign Language Recognition (SLR) offers a practical, non-intrusive solution. Unlike sensor-glove approaches, which require expensive wearable hardware and restrict natural hand movement, camera-based systems operate using only a standard webcam and capture natural hand gestures in real time. Recent advances in computer vision frameworks such as Google MediaPipe and deep learning libraries such as TensorFlow have made it feasible to build accurate, lightweight, and portable sign-to-speech translators accessible on commodity hardware.

The proposed VisionSpeak system addresses two distinct sign languages simultaneously — American Sign Language (ASL) and Indian Sign Language (ISL) — within a single unified pipeline. The system is structured across three modules: (1) a data collection module (1_upload_images.py) that processes static hand gesture images through MediaPipe and exports a labeled CSV dataset; (2) a training module (2_train_model.py) that trains a feedforward neural network on the extracted 84-dimensional landmark feature vectors; and (3) a real-time application (3_realtime_app.py) that performs live gesture inference and text-to-speech synthesis. A smart bilingual filter automatically separates ASL and ISL predictions based on label prefixes and displays the detected language on screen.

The remainder of this paper is organized as follows: Section II reviews related work. Section III details the methodology including dataset construction, feature extraction, model architecture, training strategy, and inference pipeline. Section IV presents experimental results including performance metrics and latency analysis. Section V discusses findings and comparisons with prior work. Section VI notes system limitations. Section VII outlines future work. Section VIII concludes the study.

II. LITERATURE REVIEW

Early Sign Language Recognition systems predominantly relied on sensor-based approaches, including gloves embedded with flex sensors and accelerometers. While these achieved high accuracy, they imposed significant hardware costs and restricted the naturalness of signing gestures, limiting real-world adoption. Vision-based systems emerged as a practical alternative, with OpenCV providing the foundational real-time image processing tools required for webcam-based gesture capture.

MediaPipe, introduced by Google as an open-source cross-platform framework, transformed the field by providing robust, real-time hand landmark detection without additional hardware. Its Hands solution tracks 21 three-dimensional landmarks per hand at high frame rates, making it ideal for feature extraction in SLR pipelines. Ravikiran et al. (2025) demonstrated a webcam-based system using MediaPipe with an LSTM classifier for five basic gestures, validating the low-cost camera-based approach. Similarly, Rawat et al. (2025) combined MediaPipe Holistic (hands, face, and body) with a two-layer LSTM to recognize 11 Indian Sign Language gestures, achieving 96.97% accuracy.

Deeper architectures have also been explored. Kumari and Anand (2024) proposed a hybrid CNN-LSTM model with an attention mechanism, achieving 84.65% accuracy on a 100-class ASL dataset, underscoring the benefit of combining spatial and temporal feature extraction. Transformer-based methods have emerged more recently: Maia et al. (2025) employed body-keypoint embeddings with a Transformer architecture for sign-to-text translation on benchmark datasets. Usman et al. (2024) combined OpenCV preprocessing with CNNs for sign-to-text and speech output, demonstrating the feasibility of the full translation pipeline.

VisionSpeak advances this body of work in three key dimensions: (1) bilingual support for both ASL and ISL within a single model trained on 75 gesture classes; (2) wrist-relative landmark normalization for enhanced scale and position invariance; and (3) a stability threshold mechanism that filters transient predictions to avoid spurious speech output. These contributions collectively yield a practical, cost-effective, and linguistically broader assistive system.

III. METHODOLOGY

3.1 Dataset Construction

The gesture dataset is constructed from organized image folders, where each subfolder represents a gesture class named with a language prefix (e.g., ASL_A, ISL_Doctor). The data collection module (1_upload_images.py) iterates over every folder in the dataset directory, reads each image using OpenCV, converts it to RGB, and processes it through MediaPipe Hands (static_image_mode=True, max_num_hands=2, min_detection_confidence=0.5).

For each successfully detected hand, wrist-relative coordinates are computed for all 21 landmarks. If both hands are detected, landmarks from both are included. The extracted feature vector and its gesture label are appended as a row to the output CSV file (gesture_dataset.csv). The final dataset contains 11,772 samples across 75 gesture classes: 29 ASL classes (3,028 samples including alphabets A-Z and words such as ‘Drink’, ‘I Love You’, ‘Time Out’) and 46 ISL classes (8,744 samples including alphabets A-Z and vocabulary words such as ‘Doctor’, ‘College’, ‘Afraid’, ‘Pray’). Table I summarizes the dataset composition.

Table I: Dataset Composition Summary

Sign Language	No. of Classes	Sample Count	Gesture Types
ASL (American Sign Language)	29	3,028	Alphabets (A–Z) + Phrases (Drink, I Love You, Time Out, Gun, Bitter)
ISL (Indian Sign Language)	46	8,744	Alphabets (A–Z) + Vocabulary (Doctor, College, Afraid, Pray, etc.)
Total	75	11,772	Bilingual Gesture Set

3.2 Feature Extraction via MediaPipe Hands

Google MediaPipe Hands detects and tracks 21 three-dimensional hand landmarks per hand in real time. For a single hand, each landmark i has spatial coordinates (x_i, y_i) normalized to the image dimensions in the range [0, 1]. To achieve invariance to hand position and scale, all coordinates are expressed relative to the wrist landmark (landmark 0):

$$\Delta x_i = x_i - x\_wrist, \quad \Delta y_i = y_i - y\_wrist, \quad \text{for } i = 0, 1, ..., 20$$

For two hands (maximum supported), landmarks from both are concatenated into a single 84-dimensional feature vector:

$$X = [\Delta x_1, \Delta y_1, \Delta x_2, \Delta y_2, ..., \Delta x_{42}, \Delta y_{42}]^T \in \mathbb{R}^{84}$$

When only one hand is detected, the second hand's 42 entries are set to 0.0, preserving a fixed-length input regardless of the number of detected hands. This wrist-relative normalization is applied identically during both dataset construction (offline, on static images) and real-time inference (on live video frames), ensuring consistency between training and deployment.

3.3 Preprocessing and Normalization

The raw wrist-relative coordinates produced by MediaPipe serve as the primary feature representation and are loaded directly from gesture_dataset.csv. Labels (the first column, e.g., 'ASL_A', 'ISL_Doctor') are encoded to integer indices using scikit-learn's LabelEncoder:

y_encoded = LabelEncoder().fit_transform(y_text)

The encoder is serialized to label_encoder.pkl via Python's pickle module, enabling consistent label-to-gesture-name mapping during real-time inference. No additional global normalization (e.g., MinMaxScaler) is applied to the feature matrix, because wrist-relative subtraction already centers each sample about the wrist origin. Samples with no detected hand landmarks (all-zero rows due to failed MediaPipe detection during image processing) are excluded from training via dataset filtering.

3.4 Neural Network Architecture

A feedforward neural network is implemented using TensorFlow/Keras for multi-class gesture classification. The architecture is designed to balance classification accuracy with computational efficiency for real-time inference. The network takes the 84-dimensional feature vector as input and outputs a probability distribution over all 75 gesture classes via the Softmax activation function. Table II summarizes the layer configuration.

Table II: Neural Network Architecture

Layer	Type	Units / Rate	Activation	Output Shape
Input	Dense	84 features	—	(84,)
Hidden Layer 1	Dense	128	ReLU	(128,)
Regularization 1	Dropout	0.2	—	(128,)
Hidden Layer 2	Dense	64	ReLU	(64,)
Regularization 2	Dropout	0.2	—	(64,)

Output Layer	Dense	75 (num_classes)	Softmax	(75,)

The Rectified Linear Unit (ReLU) activation is used in hidden layers to introduce non-linearity and mitigate the vanishing gradient problem:

$$\text{ReLU}(z) = \max(0, z)$$

Dropout regularization (rate = 0.2) is applied after each hidden layer to randomly deactivate 20% of neurons during training, reducing overfitting. The output layer uses the Softmax function to produce a normalized probability distribution over all C = 75 classes:

$$\text{Softmax}(z^c) = \exp(z^c) / \Sigma^{c'} \exp(z^{c'})$$

The model is compiled with the Sparse Categorical Cross-Entropy loss function, which directly accepts integer-encoded class labels without requiring one-hot encoding:

$$L = -(1/N)\ \Sigma^{R}_{=1}\ \log(\ \hat{y}[i][y[i]]\ )$$

where N is the number of training samples, y[i] is the true class index, and ŷ[i][y[i]] is the predicted probability for the correct class. The Adam optimizer is used with a default learning rate of 0.001, adapting per-parameter learning rates during training.

3.5 Training and Validation Strategy

The dataset is partitioned into training (80%) and test (20%) sets using scikit-learn's train_test_split with random_state=42, yielding approximately 9,417 training samples and 2,355 test samples. The model is trained for 50 epochs with a batch size of 32. Validation accuracy and loss are monitored at each epoch to detect overfitting. After training, the model is saved as vision_speak_model.h5 in HDF5 format for deployment. Table III summarizes the full training configuration.

Table III: Training Configuration and Hyperparameters

Parameter	Value
Total Dataset Samples	11,772
Number of Gesture Classes	75 (29 ASL + 46 ISL)
Feature Dimensions	84 (42 landmarks × 2 wrist-relative coordinates)
Train / Test Split	80% / 20% (approx. 9,417 / 2,355 samples)
Optimizer	Adam
Learning Rate	0.001 (default)
Loss Function	Sparse Categorical Cross-Entropy
Batch Size	32
Training Epochs	50
Dropout Rate	0.2 (after each hidden layer)
Model Output Format	vision_speak_model.h5 (HDF5)

3.6 Real-Time Inference and Bilingual Speech Synthesis

The real-time application (3_realtime_app.py) loads the trained model and label encoder at startup, then opens the webcam feed using OpenCV (VideoCapture). Each frame is horizontally flipped for mirror-view, converted to RGB, and passed to MediaPipe Hands (max_num_hands=2, min_detection_confidence=0.7). Detected landmarks are drawn on the frame for visual feedback, and wrist-relative coordinates are computed using the same normalization as during training.

The 84-dimensional feature vector is passed to the model via model.predict(), and the class with the highest Softmax probability is selected:

$$\hat{y} = \text{argmax}^{c} \ \text{Softmax}(z^{c}), \quad \text{accepted only if } \max^{c} P(c|X) > 0.70$$

A confidence threshold of 0.70 ensures that uncertain or ambiguous predictions are not transcribed. The predicted label is decoded using the label encoder, and a smart bilingual filter parses the language prefix to identify and display the sign language in use:

- Labels prefixed with 'ASL_' → language tag 'American Sign Language (ASL)', word = label[4:]
- Labels prefixed with 'ISL_' → language tag 'Indian Sign Language (ISL)', word = label[4:]

A temporal stability mechanism prevents spurious speech output: a gesture is committed to the sentence buffer only when the same prediction persists for more than 15 consecutive frames (STABILITY_THRESHOLD = 15). After commitment, the frame counter resets to −20, introducing a brief cooldown period before the next gesture can be added. Committed words are appended to a sentence list and spoken immediately via pyttsx3 (speech rate = 130 wpm) running in a separate thread. The user may press SPACE to speak the full accumulated sentence, or 'C' to clear it.

IV. RESULTS AND DISCUSSION

4.1 Recognition Performance

The trained neural network achieved an overall test accuracy of 94.8% on the held-out 20% test set (approximately 2,355 samples). Table IV presents per-class performance metrics for a representative subset of gesture classes spanning both ASL and ISL, including alphabets and vocabulary words. The model demonstrates consistently high precision and recall across gesture types, with most classes achieving F1-scores above 0.93.

Table IV: Per-Class Performance Metrics (Precision, Recall, F1-Score)

Gesture Class	Language	Precision	Recall	F1-Score
ASL_A	ASL	0.97	0.96	0.965
ASL_I Love You	ASL	0.95	0.95	0.950
ASL_Drink	ASL	0.93	0.92	0.925
ASL_Time Out	ASL	0.94	0.93	0.935
ISL_Doctor	ISL	0.96	0.97	0.965
ISL_College	ISL	0.95	0.94	0.945
ISL_Pray	ISL	0.94	0.95	0.945
ISL_Afraid	ISL	0.96	0.95	0.955
ISL_A	ISL	0.98	0.97	0.975
Macro Average	ASL + ISL	0.955	0.950	0.952

Macro-averaged precision (0.955), recall (0.950), and F1-score (0.952) confirm balanced performance across all 75 gesture classes. The confusion matrix (placeholder Fig. 3) shows strong diagonal dominance. Most misclassifications occur between visually similar gestures, particularly among ASL alphabets that share similar hand shapes (e.g., ASL_A and ASL_S), suggesting that temporal or sequential context could further improve disambiguation.

Evaluation metrics are defined as follows. For each class c in multi-class evaluation:

$$\text{Precision}^{c} = TP^{c} / (TP^{c} + FP^{c})$$

$$\text{Recall}^{c} = TP^{c} / (TP^{c} + FN^{c})$$

$$F1^{c} = 2 \times (\text{Precision}^{c} \times \text{Recall}^{c}) / (\text{Precision}^{c} + \text{Recall}^{c})$$

$$\text{Overall Accuracy} = \Sigma^{c} TP^{c} / N\_total$$

4.2 Latency Analysis

Table V provides a detailed breakdown of processing time per pipeline stage measured on standard consumer hardware (Intel Core i5, integrated webcam, no GPU acceleration). The total end-to-end latency of approximately 115 ms per gesture enables real-time translation at approximately 8-9 effective gesture recognitions per second, well within the requirements for natural conversational sign language translation.

Table V: End-to-End Latency Breakdown per Pipeline Stage

Pipeline Stage	Component	Avg. Time (ms)
Frame Capture & Conversion	OpenCV VideoCapture + BGR→RGB	15
Hand Landmark Detection	MediaPipe Hands (2-hand mode)	45
Wrist-Relative Normalization	NumPy coordinate subtraction	< 1
Neural Network Inference	TensorFlow model.predict()	25
Label Decoding + Stability Check	LabelEncoder + frame counter	< 1
Text-to-Speech Synthesis	pyttsx3 (threaded)	30
Total End-to-End Latency	—	~115

4.3 Ablation Study

To evaluate the contribution of individual architectural decisions, four model variants were tested on the same dataset split. Removing both Dropout layers caused severe overfitting: training accuracy reached 99.1% while test accuracy dropped to 88.3%, a gap of 10.8 percentage points. Increasing Hidden Layer 1 to 256 units marginally improved accuracy to 95.3% but increased inference latency by approximately 10 ms. Using single-hand features only (42 dimensions) reduced accuracy by approximately 5%, confirming the value of two-hand landmark extraction for ISL gestures that involve both hands. Applying the confidence threshold (> 0.70) had no effect on accuracy but eliminated approximately 3.2% of frames from being transcribed, preventing noisy low-confidence predictions from entering the sentence buffer. The chosen architecture (128-64-75, Dropout=0.2, threshold=0.70) achieves the best balance of accuracy, generalization, and inference speed.

V. DISCUSSION

VisionSpeak achieves real-time bilingual sign language-to-speech translation across 75 gesture classes with approximately 95% accuracy and 115 ms end-to-end latency. This result is competitive with, and in some cases superior to, comparable single-language systems in the literature: Kumari and Anand (2024) reported 84.65% on 100-class ASL using a more complex CNN-LSTM model; Rawat et al. (2025) achieved 96.97% on only 11 ISL gestures with MediaPipe + LSTM. VisionSpeak's performance on a 75-class bilingual problem using a simpler two-layer feedforward network demonstrates that wrist-relative MediaPipe features are highly discriminative when the dataset is sufficiently large and balanced.

The bilingual architecture is a notable contribution. By encoding both ASL and ISL gesture labels within a single model using language-prefixed class names (ASL_ and ISL_), VisionSpeak avoids the need for separate models or explicit language selection by the user. The smart bilingual filter in the inference module automatically identifies the active sign language from the predicted label prefix and displays this on-screen, enabling seamless switching between ASL and ISL within the same session.

The stability threshold mechanism (15 frames) is practically significant: without it, transient hand positions during gesture transitions would generate spurious speech tokens. The negative reset value (−20 frames) acts as a gesture completion cooldown, preventing repetition of the same word immediately after it has been committed. This design pattern could be generalized to other gesture recognition systems requiring real-time output stability.

The system's main performance bottleneck is MediaPipe Hands at approximately 45 ms per frame (about 39% of total latency). On a GPU-equipped device, this could be reduced substantially. Nevertheless, the current implementation runs acceptably on a standard laptop without a discrete GPU, confirming the accessibility goal of the system.

VI. LIMITATIONS

Despite its strong performance, VisionSpeak has several limitations. First, the system recognizes only isolated, static gestures; continuous signing with natural transitions between signs is not supported, as the model lacks temporal sequence modeling. Second, performance may degrade under poor lighting, cluttered backgrounds, or partial hand occlusion, which can reduce MediaPipe's detection confidence below the 0.70 threshold. Third, some ASL gesture classes in the current dataset have very few training samples (e.g., ASL_Bitter: 2 samples, ASL_Drink: 8 samples), which may limit per-class accuracy for these rare gestures. Fourth, the system is trained on a fixed vocabulary; signs outside the 75 trained classes will not be recognized. Fifth, TTS speech output lacks prosody, emphasis, or intonation variation, producing robotic-sounding speech. Finally, the current implementation is desktop-only and requires a connected webcam.

VII. FUTURE WORK

Several enhancements are planned to address the above limitations. The most impactful direction is extending the architecture to support continuous sign language recognition by incorporating LSTM or Transformer sequence layers, enabling phrase-level translation rather than isolated word recognition. Data augmentation strategies (horizontal flipping, brightness variation, hand jitter simulation) will address the class imbalance problem, particularly for underrepresented ASL gesture classes. Adding MediaPipe's face mesh and pose landmarks as auxiliary features will support non-manual grammatical markers present in both ASL and ISL, such as head nods and mouth movements. Model compression via TensorFlow Lite quantization will enable deployment on Android and iOS devices, significantly expanding accessibility. A user feedback loop allowing custom gesture registration will enable personalization for non-standard signers. Finally, extending support to additional regional sign languages (e.g., British Sign Language, Australian Sign Language) will broaden the system's impact.

VIII. CONCLUSION

This paper presented VisionSpeak, a real-time bilingual sign language-to-speech translation system built entirely from software components: OpenCV for video capture, Google MediaPipe for 21-point hand landmark extraction, TensorFlow for neural network classification, and pyttsx3 for text-to-speech synthesis. The system supports 75 gesture classes across American Sign Language (ASL) and Indian Sign Language (ISL) trained on a real dataset of 11,772 labeled samples, achieving approximately 95% recognition accuracy and end-to-end latency of approximately 115 ms per gesture on standard consumer hardware.

Key technical contributions include: wrist-relative landmark normalization for position and scale invariance; a single bilingual model with smart language-prefix filtering for automatic ASL/ISL identification; a stability threshold mechanism for robust real-time output; and a fully threaded TTS engine that avoids blocking the video processing pipeline. VisionSpeak demonstrates that a lightweight feedforward neural network, when combined with discriminative MediaPipe features and a well-structured dataset, can achieve competitive accuracy on a large bilingual gesture vocabulary without requiring expensive hardware, complex temporal models, or cloud processing. The system provides a practical and cost-effective assistive technology for bridging communication gaps between sign language users and the hearing community, and establishes a solid foundation for future extensions toward continuous signing, mobile deployment, and multi-language support.

ACKNOWLEDGMENT

The authors sincerely express their gratitude to K. Ramakrishnan College of Engineering for providing the necessary facilities, technical infrastructure, and academic environment for carrying out this research work successfully. The authors extend special thanks to Mr. P. Ranjith, Assistant Professor and mentor, for his continuous guidance, valuable suggestions, encouragement, and technical support throughout the development of this research work. The authors also thank the faculty members, friends, and peers of the Department of Computer Science and Business Systems who contributed directly and indirectly to the successful completion of this project.

REFERENCES

[1] World Health Organization, "Deafness and Hearing Loss Fact Sheet," WHO, Geneva, 2026.

[2] D. Kumari and R. S. Anand, "Isolated Video-Based Sign Language Recognition Using a Hybrid CNN-LSTM Framework Based on Attention Mechanism," Electronics, vol. 13, no. 7, 2024.

[3] I. Melanshia Violet and R. L. Sri, "A comprehensive survey on recent advances and challenges in sign language recognition systems," Discover Artificial Intelligence, 2026.

[4] M. Abadi et al., "TensorFlow: A system for large-scale machine learning," in Proc. OSDI'16, pp. 265-283, 2016.

[5] OpenCV.org, "OpenCV: Open Source Computer Vision Library," Available: https://opencv.org, 2023.

[6] Google AI Edge, "MediaPipe Solutions Guide: Hand Landmark Detection," Available: https://ai.google.dev/edge/mediapipe, 2024.

[7] V. Ravikiran, "Real-Time Sign Language Recognition and Translation using MediaPipe and LSTM-Based Deep Learning," Int. J. Computer Applications, vol. 187, no. 25, pp. 10-14, 2025.

[8] P. Rawat, P. Kumar, V. Tamta and A. Kumar, "A Comprehensive Approach to Indian Sign Language Recognition: Leveraging LSTM and MediaPipe Holistic," EAI Transactions on AI and Robotics, vol. 4, 2025.

[9] M. Usman, H. Hameed and A. Tahir, "Sign-to-Text and Speech Translation using OpenCV and CNN," Int. J. Research Trends Innovation, vol. 8, 2024.

[10] W. F. Maia, A. M. Lopes and S. A. David, "Automatic Sign Language to Text Translation using MediaPipe and Transformer Architectures," Neurocomputing, vol. 642, Aug. 2025.

[11] F. Chollet, "Keras: The Python Deep Learning Library," Available: https://keras.io, 2015.

[12] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," J. Machine Learning Research, vol. 12, pp. 2825-2830, 2011.