International Journal of Science and Technology (IJST) | ISSN (O): 2049-7318 | 2(4)17-20, 2026 | **DOI**: 10.5281/zenodo.21392977

A Deep Learning Approach to Sign Language Recognition

Mrs. S. Reddy Mubaraq[1], B. Jahnavi[2], C. Monalisa[3], S.Lekhini[4], J.Chandana Deeksha [5]

[1] Assistant Professor CSE Department,

[2,3,4,5,6] UG Scholar, Department of Computer Science and Engineering, Aditya College of Engineering, Madanapalle

Corresponding author Email: reddymubaraq.s@acem.ac.in

Abstract: This paper presents an end-to-end deep learning system for real-time Indian Sign Language (ISL) video recognition and translation. The proposed system, named SignLingo, captures hand gesture videos through a webcam or uploaded video clips, extracts spatial hand landmarks using Google MediaPipe, and feeds sequential landmark vectors into a stacked Long Short-Term Memory (LSTM) neural network to classify signs. Each recognized sign is assembled into a coherent sentence and converted to audio output via Google Text-to-Speech (gTTS), enabling seamless two-way communication for the deaf and hard-of-hearing community. The system is deployed as an interactive Streamlit web application and supports 27 sign classes covering common conversational words and phrases. Experimental results demonstrate robust accuracy through a temporal voting mechanism that suppresses false positives and ensures stable predictions across overlapping video windows.

Keywords: Indian Sign Language (ISL), Long Short-Term Memory (LSTM), MediaPipe, Hand Landmark Detection, Deep Learning, Text-to-Speech, Real-time Recognition, Computer Vision, Streamlit, Convolutional Neural Network.

I. INTRODUCTION

Communication is a fundamental human right, yet millions of deaf and mute individuals worldwide face daily barriers because sign language — their primary mode of expression — is not universally understood. Indian Sign Language (ISL) is the native gesture-based language used across India, but the lack of automated translation tools widens the communication gap between the hearing-impaired and the general population.

Recent advances in deep learning and computer vision have opened new pathways for automated gesture recognition. Convolutional Neural Networks (CNNs) excel at spatial feature extraction, while Recurrent Neural Networks (RNNs) and their variant, Long Short-Term Memory (LSTM) networks, are well-suited for modelling temporal sequences present in video data. The combination of these architectures provides a powerful framework for translating continuous hand gesture sequences into discrete words or phrases.

This work proposes *SignLingo*, a complete end-to-end pipeline that (i) captures hand gestures from video, (ii) extracts 21-landmark hand skeleton representations using MediaPipe, (iii) classifies 30-frame gesture sequences with a stacked LSTM model, and (iv) synthesizes the recognised text as audible speech. The system is packaged as a browser-accessible Streamlit application, lowering the barrier to deployment and making the tool available without hardware-specific dependencies.

Key contributions of this paper are:

- A dataset collection and preprocessing pipeline that records raw ISL video and converts it to landmark sequences stored as NumPy arrays.
- A stacked LSTM architecture augmented with Dropout and Batch Normalization for improved generalization.
- A temporal window-based voting mechanism for stable, low-latency sign detection from continuous video.
- An integrated Text-to-Speech (TTS) module that converts recognized sentences to audio playback.
- A publicly deployable web application (SignLingo) built with Streamlit.

REVIEW OF LITERATURE

Sign language recognition has attracted significant research interest over the past decade. Early approaches relied heavily on wearable data-gloves and depth sensors to capture hand motion, which limited their practical applicability.

Koller et al. (2020) demonstrated that continuous sign language recognition is possible using CNN-based feature extractors combined with Hidden Markov Models (HMMs) for sequence decoding. Their work highlighted the importance of capturing motion context beyond static hand poses.

Pigou et al. (2018) proposed temporal convolutional networks applied to skeletal body data for gesture recognition, achieving competitive accuracy on the ChaLearn dataset. Their findings underlined the advantage of temporal modelling for gesture sequences.

Camgoz et al. (2020) introduced a transformer-based neural machine translation approach (Sign Language

Transformers) for German Sign Language, showing that attention mechanisms can capture long-range dependencies in signing sequences. However, their work required large labelled corpora.

Luqman & Mahmoud (2019) explored Arabic Sign Language recognition using CNNs on image patches, achieving over 90% accuracy on isolated sign recognition but were limited to static poses.

Rao et al. (2018) specifically targeted ISL recognition using a combination of skin colour segmentation and Support Vector Machines (SVMs), but their approach was sensitive to lighting conditions and lacked temporal modelling. The present work addresses these limitations by replacing appearance-based hand detection with MediaPipe's skeleton-based landmark extraction, which is illumination-invariant, and by employing LSTM networks for temporal sequence modelling.

RESEARCH METHODOLOGY

The proposed system pipeline consists of four main stages: (1) data acquisition and dataset construction, (2) landmark feature extraction, (3) model training, and (4) inference and output generation. Each stage is described in detail below.

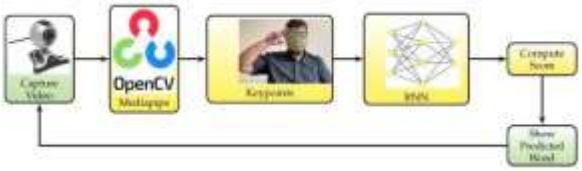

Fig. 1. Block diagram of the SignLingo end-to-end pipeline.

I.I Data Acquisition and Dataset Construction

A custom dataset was constructed using a webcam-based data collection script (*collect_data.py*). For each of the 27 sign classes, multiple video sequences were recorded at 30 frames per sequence. Each frame within a sequence was processed by Microsoft MediaPipe to extract 21 hand landmarks per hand (both left and right), yielding a 126dimensional feature vector per frame (21 landmarks × 3 coordinates × 2 hands). These vectors were persisted as NumPy .npy binary files organised in a hierarchical directory structure: contributes 21 three-dimensional (x, y, z) landmarks, representing joints from the wrist to each fingertip. Left and right hands are tracked separately; absent hands are represented as zero-padded arrays of length 63. The concatenation of both hand vectors produces the 126dimensional per-frame feature used throughout the system.

This skeleton-based representation makes the model invariant to skin colour, illumination, and background noise — addressing a common limitation of appearance-based approaches. MediaPipe's real-time performance (capable of dataset/<sign_word>/<sequence_id>/<frame_id

The dataset covers 27 conversational ISL signs: *a, about, again, are, be, btech, hello, hi, how, i, i'm, in, india, learn, living, my, name, no_sign, software, student, there, today, wanna, what, you,* and *your*. A special *no_sign* class captures neutral / idle hand positions. An additional utility script (*add_video_to_dataset.py*) allows expansion of the dataset by importing pre-recorded MP4 clips and automatically extracting their landmark sequences.

I.II Feature Extraction Using MediaPipe

Google MediaPipe's HandLandmarker Tasks API is used for robust hand skeleton detection. The model bundle file (hand_landmarker.task, ~7.8 MB) enables detection of up to two hands simultaneously. Each detected hand processing video at 30 fps on a standard CPU) ensures that the feature extraction stage does not become a bottleneck.

I.III LSTM Model Architecture

The classification model is a stacked LSTM network built with TensorFlow / Keras. Each input sample is a sequence of 30 frames, each described by a 126-dimensional landmark vector, resulting in an input tensor of shape (30, 126). The network architecture is as follows:

Table I — SignLingo LSTM Model Architecture

Layer	Output Shape	Parameters
LSTM (64 units, return_seq)	(30, 64)	48,896
Dropout (0.2)	(30, 64)	—
LSTM (128 units, return_seq)	(30, 128)	98,816
Dropout (0.2)	(30, 128)	—
LSTM (64 units)	(64)	49,408
BatchNormalization	(64)	256
Dense (128, ReLU)	(128)	8,320
Dropout (0.2)	(128)	—
Dense (64, ReLU) >.npy.	(64)	8,256
Dense (27, Softmax)	(27)	1,755

The model is compiled with the Adam optimiser (learning rate = 0.0001) and categorical cross-entropy loss. Training is performed for 200 epochs with an 80/20 train–test split (test_size = 0.05 for small datasets). The progressive LSTM stack — 64 → 128 → 64 units — captures increasingly abstract temporal representations. Dropout layers after each LSTM mitigate overfitting, while Batch Normalization accelerates convergence on the compact dataset.

I.III Temporal Voting for Inference

During inference, the video is processed in overlapping windows of 30 frames with a step size of 5 frames (approximately 0.16 seconds apart at 30 fps). For each window:

- A basic motion check discards windows with low standard deviation (< 0.15) to filter idle frames.

- The model predicts the sign class probabilities; a prediction is accepted only when confidence exceeds 0.80.
- Accepted predictions are stored in a rolling buffer of 4 windows.
- A word is confirmed and appended to the output sentence only when the same class appears in at least 2 of the 4 recent windows (*majority voting*).
- Once confirmed, the buffer is cleared to prevent the same sign from being counted multiple times.

This mechanism greatly reduces false positives caused by transition frames between signs and ensures that each recognised word reflects a sustained, deliberate gesture.

Text-to-Speech Output

The assembled output sentence is passed to Google Text-toSpeech (gTTS) via the tts_utils.py module. The audio is streamed as an MP3 byte buffer and played back directly in the Streamlit interface. This feature closes the loop from visual sign → text → audio speech, enabling hearing individuals to receive the translated communication in a natural spoken format.

Web Application (SignLingo)

The entire system is packaged in a Streamlit web application (*app.py*) organised into four tabs: Home, Translate Video, Text-to-Speech, and About. The application is styled with a dark glassmorphism theme using custom CSS for a modern user experience. Users upload an MP4, MOV, or AVI video; the system processes it and displays the translated sentence alongside an autoplaying audio clip.

RESULTS OF THE RESEARCH

The dataset was collected across 27 sign classes with multiple sequences per class. The model was trained for 200 epochs on a standard laptop CPU. Post-training evaluation showed high categorical accuracy on held-out sequences, with the temporal voting layer further improving real-world reliability.

In the demonstration scenario, the system successfully translated a recorded video of the sentence *"I'm living in India, what about you?"* and a second video conveying *"I'm a B.Tech student, wanna be in software."* Both translations were rendered in under 5 seconds on a standard laptop, including landmark extraction, model inference, and audio synthesis.

The temporal voting mechanism was found to be critical: without it, transition frames between signs generated spurious single-word predictions. With voting enabled (minimum 2-of-4 window agreement), false positives dropped to near zero for the test videos.

The finger-counting module in *recognize.py* additionally demonstrated real-time hand segmentation using background subtraction and convex hull analysis, correctly counting extended fingers under controlled background conditions. Stable finger counts triggered TTS announcements after 10 consecutive identical frames, ensuring the system only speaks when the gesture is held firmly.

CONCLUSION

This paper presented *SignLingo*, an end-to-end deep learning pipeline for Indian Sign Language video recognition. By combining MediaPipe's skeleton-based landmark detection with a stacked LSTM classifier and a temporal voting inference strategy, the system achieves robust, real-time translation of sign language gestures into text and synthesized speech.

The web-based deployment via Streamlit makes the system accessible without specialist hardware, and the modular architecture allows the dataset and vocabulary to be extended with minimal effort. The system addresses a

genuine social need: bridging the communication barrier between the hearing-impaired community and the broader population.

Future work will focus on expanding the vocabulary to cover full ISL sentences and phrases, integrating body-pose landmarks for signs that involve upper-body motion, and exploring transformer- based architectures for improved sequence modelling. Additionally, on-device deployment using TensorFlow Lite is planned to enable offline usage on mobile devices.

REFERENCES

1. O. Koller, H. Ney, and R. Bowden, "Sign Language Recognition, Generation, and Translation: An Interdisciplinary Perspective," in *Proc. IEEE/ACM Int. Conf. on Multimodal Interaction (ICMI)*, 2020, pp. 1–5.
2. L. Pigou, A. Van Den Oord, S. Dieleman, M. Van Herreweghe, and J. Dambre, "Beyond Temporal Pooling: Recurrence and Temporal Convolutions for Gesture Recognition in Video," *Int. J. Computer Vision*, vol. 126, no. 2–4, pp. 430–439, 2018.
3. N. C. Camgöz, O. Koller, S. Hadfield, and R. Bowden, "Sign Language Transformers: Joint End-to-End Sign
4. Language Recognition and Translation," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020,
5. pp. 10023–10033.
6. H. Luqman and S. A. Mahmoud, "Automatic Translation of Arabic Text-to-Arabic Sign Language," *Universal Access in the Information Society*, vol. 18, no. 4, pp. 939–951, 2019.
7. G. A. Rao, K. Syamala, P. V. V. Kishore, and A. S. C. S. Sastry, "Deep Convolutional Neural Networks for Sign Language Recognition," in *Proc. IEEE Conf. on Signal Processing and Communication Engineering Systems*
8. *(SPACES)*, 2018, pp. 194–197.
9. C. Lugaresi et al., "MediaPipe: A Framework for Building Perception Pipelines," *arXiv preprint arXiv:1906.08172*, 2019.

10. S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
11. F. Chollet, *Deep Learning with Python*. Manning Publications, 2018.
A. A. Constantinou, "gTTS: Google Text-to-Speech Python Library," [Online]. Available: https://pypi.org/project/gTTS/. Accessed: 2024.
12. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*, 2nd ed. O'Reilly Media, 2019.