Real-Time Sign Language Recognition and Translation using MediaPipe and LSTM-Based Deep Learning

Ravikiran V.
Department of Information Science And Engineering
JSS Science And Technology University
Mysuru, Karnataka, India

ABSTRACT

People with speech and hearing impairments find it hard to communicate with others. The Sign Language Gesture Recognition and Translation system in this paper is meant to help people with speech impairments interact easily with others. The system can pick out five important sign language gestures: Yes, No, I Love You, Thank You, Hello, and OK, and convert them to clear text messages right away. The main part of the system uses a Long Short-Term Memory (LSTM) neural network, Python, OpenCV, and Mediapipe to ensure both accurate and detailed hand tracking. Using computer vision, the program effectively reads each video frame from the webcam and detects the movements of hands very accurately and quickly. The fact that the interface just needs hand movements in front of any webcam means that this technology is easy to use and not expensive. By carrying out this research, we want to help the speech and hearing impaired communicate better, through easy and effective solutions. Since the results were encouraging, the system can be used in real-life situations and applied to a bigger range of sign vocabulary.

Keywords

Sign Language Recognition, Hand Gesture Detection, LSTM, Recurrent Neural Network, Computer Vision, Mediapipe, OpenCV, Real-Time Translation, Assistive Technology, Speech Impairment

1. INTRODUCTION

This work is in line with the latest trends by analyzing some basic expressions — Yes, No, I Like You, Hello, Thank You and Okay — using a camera, MediaPipe for facial features and an LSTM network for recognizing when someone performs the gesture. Communication helps people bond, display their feelings and function well in different situations. Nevertheless, people with challenges in speech and hearing find it very challenging to communicate every day. Sign language is one of the most well-known ways to communicate without using words and it includes many purposeful gestures and expressions. This means that, although sign language is helpful for people who speak it, many cannot understand it which leads to problems with inclusion, accessibility and independence. With the help of AI, computer vision and deep learning, new solutions to this problem have been made available. With SLR systems, it is possible for people who use sign language to communicate with non-signers through near-instant translation into messaging or sound. They help people be independent from human interpreters and participate more fully in different social, work and learning settings.

The paper aims to develop a system that detects and classifies Sign Language gestures with live camera input and delivers the same gestures in text format. Python supports the development of the system, while OPencV is used for image processing and MediaPipe is applied for accurate identification of hand points. The recognition engine makes use of a Long Short-Term Memory (LSTM) neural network which is based on a Recurrent Neural Network (RNN) and able to understand time-related patterns in data. Using hand landmark sequences over a period, the model understands gestures more successfully and reliably. The main five signs on the prototype are “Yes,” “No,” “I Love You,” “Thank You,” ”Hello,” and “OK,” which are used a lot in daily conversations. Just by making a gesture, users can get an instant response from the system which displays what the gesture means on the screen. This way of working shows that AI-assisted assistive devices are helpful and practical for daily use.

Apart from its usefulness, SpeechAmp helps speech-impaired people involve themselves more in mainstream communication. It is intentionally designed to be light, inexpensive and friendly for users which allows it to be useful in places where resources are missing. It could also be developed to cover a wider range of signs, let people use different languages and offer spoken words in future upgrades. The outcome of this study moves us toward AI and technologies that support and benefit both people with and without disabilities.

2. LITERATURE SURVEY

Many researchers focus on sign language recognition, since accessible methods of communication are now needed more than ever by people with speech and hearing disabilities. Because of progress in deep learning and computer vision, plus the availability of low-cost cameras, current systems can identify sign language in real-time.
In 2023, Ridwang et al. suggested using a system made up of MediaPipe Holistic for landmarks of hands, faces and bodies along with an LSTM model for classifying temporal gestures. Result of their test was a performance of 99.4% for recognizing dynamic sign gestures and 85% for translating words continuously. It was shown that adding holistic landmark extraction to recurrent neural networks makes the system accurate and suitable for real-life situations. The method cuts down on the need for particular sensors. It makes use

of an RGB camera alone which makes it quick and simple for users to access and scale.

Srivastava et al. (2024) introduced a continuous model for sign language recognition that deals with Indian Sign Language (ISL) films and uses MediaPipe Holistic features with LSTM classifiers. Essentially, the method focused on solving the issues of separating signs and coping with temporal shifts by noticing the pattern of movements that display gestures. Because the model was 88.23% accurate, it was able to identify signs that included hand movements, facial expressions and body postures. Researchers pointed out that getting multiple types of landmarks is important for accurate understanding of signs found in the outdoors.

A study in 2025 used MediaPipe's hand and body tracking along with an LSTM-based system to identify numeric gestures in NSL which reached a 95% accuracy rate on the data they had. Research found that the model can work with different signers and at various gestures speeds, highlighting how well deep learning works in the field of sign language. Lightweight and real-time SLR systems built for common computer hardware are becoming a common topic in the literature. Abdul et al. (2023) came up with a small editing pipeline that makes MediaPipe landmark coordinates into inputs for an LSTM classifier, offering high precision and reducing the amount of computing power needed. Since their system worked on standard notebooks and mobile phones, it was fit for helping with everyday accessibility.Recently, some researchers are working on translating sign language continuously in real time. Using methods that mix natural language processing (NLP) and deep learning such sign sequences are now interpreted as meaningful sentences, helping join the process of recognition with understanding of language. In addition, these systems usually call for big datasets and detailed architectures and their development is still ongoing.
These improvements do not fully solve the issues in identifying a lot of signs quickly and accurately for various sign languages and dialects. Rigorous data and complicated models often mean current systems only concentrate on a small set of signs. Also, experts are still researching ways to combine gesture recognition with speech and text translation.

This research matches the developing trends by working on a collection of frequently used signs — ”Yes,” ”No,” ”I Love You,” ”Hello,” ”Thank You,” and ”OK” — by capturing data with a camera, using MediaPipe and classifying the movements with an LSTM network over time. This way of developing keeps the design accurate, efficient, easy to use and allows for future updates that will add value to the solution.

3. EXPERIMENTAL SETUP

3.1 Hardware and Software Requirements

Since Python provides many resources for computer vision, machine learning and deep learning, the Sign Language Gesture Recognition and Translation System was implemented using that language. The fact that this system performs well on a personal computer and needs just a webcam makes it cost-effective, can be carried around and meets the needs of real projects, including those in low-resource places.

In the hardware part, a compact system featuring a standard webcam and a PC with at least 4GB RAM and an Intel i5 processor was employed. The system must be run on Windows or Linux and needs Python version 3.9 or later. Different Python libraries and frameworks were significant in the building of the application. Video input from the webcam was grabbed by OpenCV and the frames were processed instantly. For hand tracking, MediaPipe was used to accurately notice and mark important parts of the hands, making the system able to analyze 21 places on the hands. TensorFlow was the library of choice, with its higher-level API Keras, for developing and training the deep learning model which was built using LSTM networks. Moreover, NumPy and Pandas helped handle numbers and preprocess the data and Matplotlib could also be used to make graphs of training information and the results of the models.

This way of implementing the framework makes gesture recognition training and deployment smooth, so you get accurate and quick translations with little hardware support.

3.2 Data Collection and Preprocessing

The video records contain six main hand gestures called Yes, No, I Love You, Hello, Thank You and OK. There were many changes to the lighting and hand shapes during the recording process to make the artwork interesting. A range of data was created because people from multiple places gave their input.
MediaPipe was used on each video to find the x, y and z coordinates of 21 hand landmarks for every frame. The way the images were captured was changed so that the landmarks would match each other. For every gesture, the process captured between 30 and 40 frames and gave them to the LSTM model. The dataset is divided into 70% for training and 30% for testing purpose.

3.3 Real-Time Inference

While real-time testing happens, the system is always recording new frames from the webcam and analyzes them to get hand gesture data. The system detects 21 important points on the user's hand in every frame, giving the position and orientation in real time. These key coordinates are gathered from just 30–40 frames to make a complete gesturing sequence.

The next step is to give the collected sequence to the LSTM model which looks for patterns in the landmarks to help classify the gesture. LSTM is able to identify the sign accurately since it learned all the unique movements and expressions during its training. As soon as the gesture is recognized, the label for that gesture—such as “Yes,” “No,” “Thank You,” ”Hello,” “OK,” “I Love You”—is added to the image using the text overlay functions in OpenCV. Users can communicate naturally by making hand signs in front of the camera, since the system gives feedback very quickly.

4. METHODOLOGY

4.1 Feature Extraction Formula

MedaiPipe handles each frame of the video to find 21 hand landmarks with 3D points.

$$L = \{(x_i, y_i, z_i) \mid i = 1, 2, \ldots, 21\}$$

where landmark values x_i, y_i, z_i are between 0 and 1 inside the image frame. In every instance, the landmarks become a feature vector.

$$f_t = [x_1, y_1, z_1, x_2, y_2, z_2, \ldots, x_{21}, y_{21}, z_{21}] \in R^{63}$$

where t marks the current frame in the movie.
Every gesture video is shown as 30-length feature vectors.

$$S = [f_1, f_2, \ldots, f_{30}] \in R^{30\times 63}$$

Table 1. Summary of Collected Gesture Dataset

Label	Description	Number of Sequences	Frames per Sequence
Yes	Gesture ”Yes”	120	30
No	Gesture ”No”	120	30
Thank You	Gesture ”Thank You”	120	30
Hello	Gesture ”Hello”	120	30
OK	Gesture ”OK”	120	30
I Love You	Gesture ”I Love You”	120	30
	Total	**720**	-

This table summarizes the number of gesture sequences and frames per sequence collected for training and testing the sign language recognition system.

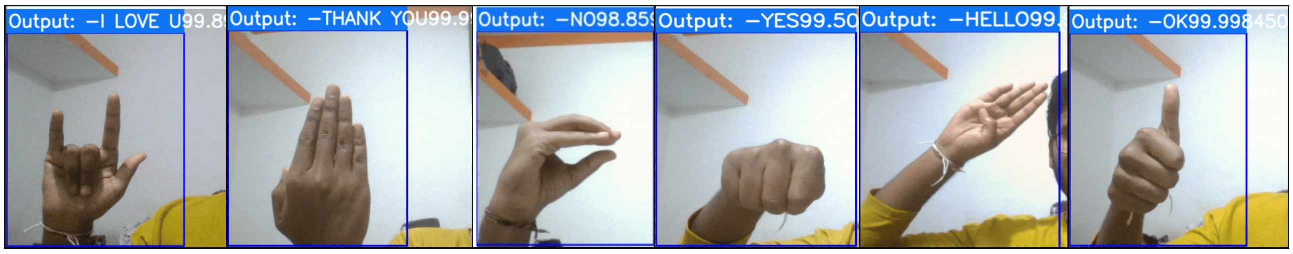


Fig. 1. Real-Time Hand Detection.

4.2 Classification Model (LSTM) Overview

The task of classification takes the input sequence S and puts it into one of the gesture categories.

$$C = \{c_1, c_2, \ldots, c_k\}$$

where $k = 6$ (e.g., THANK YOU, YES, NO, etc.).
The LSTM processes the sequence step by step as:

$$h_t = \text{LSTMCell}(f_t, h_{t-1})$$

where h_t is the hidden state at time t.
Having completed processing the frames, the last hidden state h_{30} is given to a dense (fully connected) layer with softmax activation to create the probabilities for each class.

$$p = \text{softmax}(Wh_{30} + b)$$

where W and b are learned weights and biases, and

$$p_i = P(class = c_i \mid S)$$

represents the predicted probability for class c_i.

4.3 Prediction and Decision Rule

The predicted gesture $\hat{c}$ is given by:

$$\hat{c} = \arg\max_i p_i$$

For better results, the system ignores predictions made more than 10 days ago. Should the predicted gesture be the same for the previous ten times and have a greater than $\theta = 0.8$ probability, it is accepted by the AI.

5. RESULTS AND DISCUSSION

A custom dataset made with six important gestures was used to train the proposed system, such as: Yes, No, Thank You, Hello, OK, and I Love You. Both gesture classes contained 30 sequences with 30 frames, extracted with MediaPipe landmarks analysis of a video with the webcam. The LSTM is trained with Adam optimizer and a learning rate of 0.001, and categorical cross-entropy loss function is used; after 200 epochs. This training process gave a near faultless performance as the model had 100 percent training and 98-99 percent validation performance. It can be seen in Figure 3 that the training accuracy converged at a value of 1.0, and Figure 4 depicts the loss rate to a negligible value. The learning rate (Figure 5) kept fixed in 0,001 as this will stabilize the convergence.

5.1 Real-Time Testing Evaluation

The evaluation was made in real-time involving a standard type of webcam under varying light and background conditions. It was done 10 times each by different users, a total of 60 gesture entries. Some of them were successfully recognized by the system in real-time with 96.6% on the first attempt, which is accurate. The average inference time ranged between 0.35 and 0.45 seconds indicating that the model is appropriate in low-latency applications.

5.2 Confusion Matrix Analysis

To additionally measure the performance of the classification, 30 percent test split of the complete data (54 samples) was used to generate confusion matrix and classification report. The model attained a 100% accuracy, recall and F1-score on all classes of gestures. Table 2 gives a summary of the results and a confusion matrix of the results is depicted in Figure 2.

Table 2. Classification Metrics for Each Gesture on 30% Test Split

Gesture	**Precision**	**Recall**	**F1-Score**	**Support**
THANK YOU	1.00	1.00	1.00	9
YES	1.00	1.00	1.00	9
NO	1.00	1.00	1.00	9
HELLO	1.00	1.00	1.00	9
I LOVE U	1.00	1.00	1.00	9
OK	1.00	1.00	1.00	9
Accuracy	**100% on test dataset**			

Gesture-wise classification results from the 30% test split of the dataset.

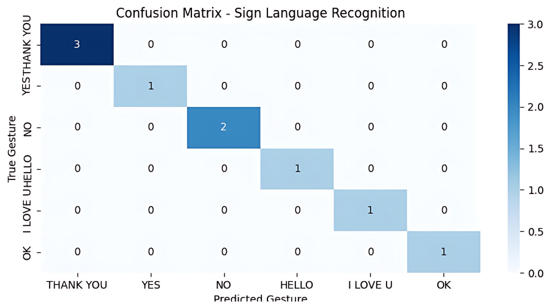
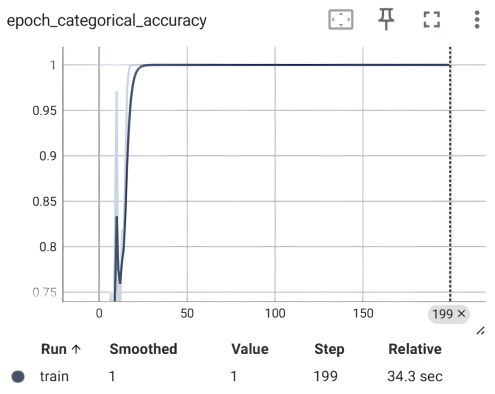


Fig. 2. Confusion matrix for six gesture classes on the test dataset.

5.3 Comparative Insight

As opposed to conventional CNN-based gesture classifiers, the suggested LSTM model exhibited improved ability to handle temporal gesture patterns. The use of MediaPipe in hand landmark extraction obviated the use of extra sensors or the use of depth cameras. This system is readily portable on a normal laptop with webcam inputs and therefore it is quite viable in the low cost applications of real time assistance to persons with deficiency in speech.

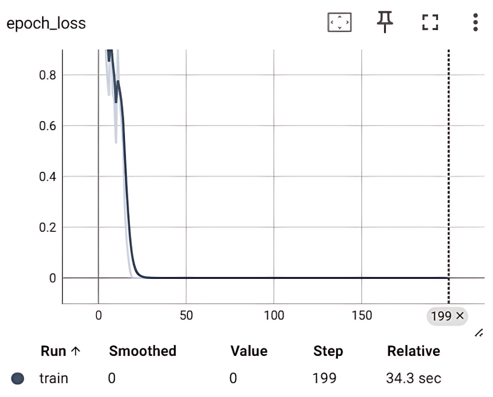
5.4 Results Overview via Training Graphs



Run ↑	Smoothed	Value	Step	Relative
● train	1	1	199	34.3 sec

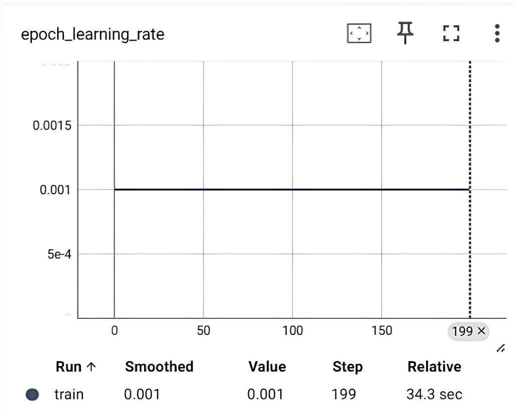
Fig. 3. Training Categorical Accuracy over 200 Epochs. Accuracy improves and stabilizes at 1.0 by the final epoch.

According to the first graph, the model achieved nearly perfect categorical accuracy right at the end of the training process. Since the model is now able to spot the patterns in the gesture data, we can say that the learning is effective. The loss curve displayed on the second graph shows a gradual decline, until it finally comes close to zero. As a result, the prediction errors decreased and the model actually improved how it classified data. Learning rate is shown in the third graph and is kept at 0.001 through all through the entire process. A consistent and smooth rate of learning made sure the model optimized without any abrupt changes. At the same time, all the visuals prove that the model was able to pick up new knowledge fast and remained stable throughout the process.



Run ↑	Smoothed	Value	Step	Relative
● train	0	0	199	34.3 sec

Fig. 4. Training Loss over Epochs. The loss decreases steadily, approaching zero.



Run ↑	Smoothed	Value	Step	Relative
● train	0.001	0.001	199	34.3 sec

Fig. 5. Constant Learning Rate of 0.001 used throughout the training process.

6. SYSTEM ARCHITECTURE

This system is made to identify hand gestures by following a clear sequence. First, the system waits for the user to start a gesture, since that shows the user's intent. After detecting the first gesture, the system improves the incoming data so it becomes simpler to look at. Following this, the hand is analyzed to look for important parts called landmarks. When using landmarks, the system finds the hand movements and their shapes that are most important. After having all the information, the model employs an LSTM classifier (Long Short-Term Memory). Being able to recognize sequences in time, the network helps spot gestures that happen sequentially. In the end, the system uses all the information it has and predicts the user's gesture.

6.1 flow diagram

The flow diagram describes each of the important steps in the gesture recognition system. The process starts when the system waits to be activated from the start point. After it has detected the start gesture, the system performs preprocessing to make sure data is ready for analysis. Then, landmarks in the hand are detected, for instance fingertips and joints, since they reveal important informa-

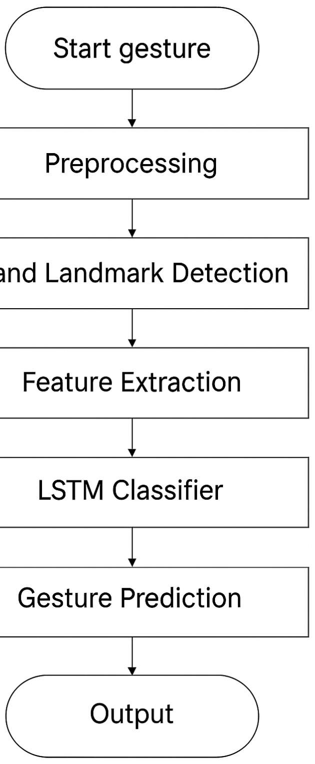


Fig. 6. Workflow diagram of the gesture recognition system

tion about a hand's position. Using these landmarks, it analyzes the gesture and gets important features that describe both the gesture's position and movement. Next, the features go to an LSTM classifier, which looks at the sequence data to properly recognize the gesture. In the end, after classifying the input gesture, the system properly responds by choosing the corresponding output.

7. CONCLUSION

The testing outcomes indicate that deep learning-based LSTM model proves to be very useful in identifying sequential hand gestures. The model was demonstrated to generalize well with a training accuracy of 100 percent and test accuracy of 100 percent on the 30 percent test split. Besides, a mean accuracy of 96.6% was observed during real-time testing, demonstrating that the model is indeed suitable to be applied in practice powered by regular webcam data. MediaPipe is a lightweight framework that uses the benefits of LSTM used to understand the temporal side of hand tracking in real-time. The present system has been able to translate effectively six commonly used sign language gestures that are, Yes, No, Thank You, Hello, OK and I Love You. It has low latency performance and only a few hardware needs thus being easily accessible and compatible with day-to-day computing equipment. The scope of future use would be to extend the system to cover a vast vocabulary of signs so that it would maintain a full sentence-level recognition. Implementation of speech synthesis will allow real-time voice feedback which will further increase accessibility to the community of speech-impaired individuals. Also, the training against more differentiated datasets would enhance generalization among various users, hand orientations, and backgrounds. A multimodal input like the recognition of the facial expression and body pose, and multilingual translation of gestures to speech are also viable future study subjects.

8. REFERENCES

[1] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[2] A. Graves, S. Fernández, and J. Schmidhuber, "Bidirectional LSTM networks for improved phoneme classification," in *Proc. Int. Conf. Artificial Neural Networks (ICANN)*, 2005, pp. 799–804.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[4] F. Chollet, *Deep Learning with Python*, 2nd ed., Manning Publications, 2021.

[5] G. Khartheesvar, M. Kumar, A. K. Yadav, and D. Yadav, "Automatic Indian Sign Language Recognition using MediaPipe Holistic and LSTM Network," *Multimedia Tools and Applications*, vol. 83, no. 20, pp. 58329–58348, 2023.

[6] P. Swetha and K. Sucharitha, "Sign Language Recognition Utilizing LSTM and MediaPipe for Dynamic Gestures of ISL," *Int. J. Res. Sci. Eng. Technol.*, vol. 10, no. 6, 2023.

[7] K. Swathi, V. Reddy, and P. Divya, "ISL Sign Language Recognition Using LSTM-Driven Deep Learning Model," *J. Electr. Syst.*, vol. 20, no. 3, 2024.

[8] P. Tripathi, A. Shukla, and A. S. Negi, "Gesture-to-Text Translation Using SURF for Indian Sign Language," *Appl. Syst. Innov.*, vol. 6, no. 2, 2023.

[9] P. Navendu and V. Sahula, "Word Level Sign Language Recognition using MediaPipe and LSTM-GRU Network," *TechRxiv*, Jul. 2024.

[10] D. KhajehPourian and A. D. Zarei, "Sign language recognition using modified deep learning network: CNNSa-LSTM," *Scientific Reports*, vol. 14, 2024.

[11] D. Patel and S. Patel, "Dynamic Indian Sign Language Recognition Based on Enhanced LSTM with Attention Mechanism," *SSR-IJECE*, vol. 11, no. 2, Feb. 2024.

[12] Y. Zhang and X. Jiang, "Recent Advances on Deep Learning for Sign Language Recognition," *Comput. Model. Eng. Sci.*, vol. 170, 2024.

[13] H. Hu et al., "SignBERT: Pre-Training of Hand-Model-Aware Representation for Sign Language Recognition," arXiv preprint arXiv:2110.05382, 2021.

[14] K. Shenoy, T. Dastane, V. Rao, and D. Vyavaharkar, "Real-time Indian Sign Language (ISL) Recognition," arXiv preprint arXiv:2108.10970, 2021.