# A MediaPipe and CNN-Based Framework for Real-Time Sign Language Recognition and Translation

*Priyanka* More[1], *Shivam* Koli[1], *Malhar* Khandekar[1], *Pathan* Arsh[1] and *Danyal* Ahmad[1]
[1]Department of Computer Science, Vishwakarma Institute of Information Technology, Pune, Maharashtra, India.
priyanka.more@viit.ac.in, shivam.22311902@viit.ac.in, malhar.22311890@viit.ac.in, arsh.22311910@viit.ac.in, danyalahmad.22311938@viit.ac.in

**Abstract.** This research is being aimed at the real-time sign language translation system developed in the absence of dedicated equipment like sensor gloves, under the sole observation of a standard camera and computer vision techniques. The project was divided into two major phases: Static Sign Recognition and Dynamic Sign Recognition. During the static phase, a fully original dataset comprising 120,000 images was created for 24 alphabets (A–Z excluding J and Z). Each image was then treated through two custom-designed filters: silhouette extraction and finger detection, carried out using MediaPipe for precise annotation and cropping on its own. The training of the static model happened with custom-designed CNN applied on the data to achieve accuracies of 94% (finger detection), 89% (silhouette), and 91% (combined). In the dynamic phase, signs with movement were captured using MediaPipe landmarks and registered as NumPy arrays (CSV and .NPY) files. Hand and facial keypoints were stored at every frame for temporal learning. This phase included one-hand, two-hand, and hand-with-face categories. The whole system achieved 15-20 FPS real-time inference on CPU alone. While the static signs and single-handed dynamic signs were performing very well, the accuracy of complex two-handed and face-assisted gestures suffered due to scarcity and noise in the data. The results provide a very good baseline to improve upon for the future using deep temporal networks like LSTM or 3D CNNs.
**Keywords:** Sign Language Recognition; CNN; MediaPipe; NumPy; Dynamic Gesture Recognition; Real-Time Translation

## 1 Introduction

Sign languages are systems of visual communication that are important to millions of people around the world. Unfortunately, it is very much the case that exceptions leave many of them in the dark due to errors brought about by the lack of those who would translate either the oral or the written language. However, deep learning and computer vision have brought a technique whereby hand gesture recognition systems can bring about automation of sign translation.

This proposed research aims to develop a real-time, camera-based interpreter of sign languages, processing both static and dynamic signs. Our work deviates from earlier work in that it depended on wearing sensory devices and pursued a vision-based learning model, using MediaPipe for landmark extraction, CNN for classification, and based model for encoding temporal dynamic gestures through NumPy.

The purpose was to build a low-cost, highly accurate model capable of performing real-time recognition with minimum computational overhead.

## 2 Literature Survey

Earlier research in sign language recognition primarily relied on glove-based sensors or marker-based tracking **[4]**, which limited user convenience and scalability. Later, image-based models using CNNs **[1]** and transfer learning architectures like MobileNet, ResNet, and InceptionV3 **[3]** improved accuracy but required massive datasets and GPU resources. Paper **[2]** uses annotation software on actually camera footage. Except images other ways to proceed is csv **[9]** which is used along with LSTM and deep-learning, Recent frameworks such as MediaPipe **[5]**, ssd+v2 **[11]** OpenPose, and DeepHand introduced efficient landmark detection from RGB frames, significantly reducing prepro- cessing complexity. Focusing on grammatical structure from sign language to english **[7]**.However, most imple- mentations focused either on static alphabets or predefined gesture sets. Dynamic signs involving temporal motion patterns still remain challenging, often requiring recurrent architectures or sequence models.

Our approach combines these domains — first establishing a robust static recognition base, then extending it into a structured temporal dataset using coordinate tracking and NumPy-based frame storage, ensuring both accuracy and real-time responsiveness.

## 3 Gap Analysis

Existing methods face several critical limitations:

- **Dependence on External Devices:** Many solutions use data gloves or depth sensors,

restricting general usability.

- **Dataset Limitations:** Most datasets are limited in di- versity, background conditions, or alphabet coverage.
- **Dynamic Gesture Inaccuracy:** Temporal gesture recognition often suffers from lack of clean sequential data
- **High Computational Cost:** Deep CNN models with millions of parameters hinder real-time processing on standard CPUs
- **Manual Annotation Issues:** Labeling of large-scale datasets remains a tedious and error-prone process.

These lacunae provided the motivation for the devel- opment of a dual-stage camera-only sign translator based on automatic annotation, light CNNs, and landmark-based temporal encoding.

## 4 Gap Analysis – Proposed Analysis

To address these issues, the research proposed the follow- ing framework

- **Automated Dataset Generation:** Using MediaPipe, we automated hand detection, cropping, and align- ment for all static alphabet signs.
- **Dual Filtering:** Created two complementary datasets
  — Silhouette Filter (binary edge-based) and Finger Detection Filter (color-coded landmarks per finger).
- **Dynamic Sign Encoding:** Stored temporal sign se- quences as CSV and NumPy (.NPY) files containing 3D coordinates of hand, face, and shoulder keypoints.
- **Separation of Motion Categories:** Divided dynamic gestures into one-hand, two-hand, and hand-with-face for clarity and targeted model development.
- **Combined Accuracy Evaluation:** Fused results from both CNNs to achieve a hybrid accuracy of 91%.

## 5 Algorithm

The system's algorithm is divided into two stages:

### 5.1 Static Sign Recognition Algorithm

1. Capture live frame from webcam.
2. Detect hand using MediaPipe Hand Detection.
3. Automatically crop image and resize to 224×224.
4. Apply pre-selected filter:
   - Silhouette Filter → Edge detection **[Fig3]**.
   - Finger Detection Filter → Color-coded land-marks **[Fig2]**.
5. Feed processed image to CNN.
6. Use ReLU activation and MaxPooling.
7. Flatten and classify through dense layer.
8. Display recognized sign (A–Z).

#### 5.1.1 *Mathematical Formulations*

Convolution:

$$O = (I * K) + b$$

Activation (ReLU):

$$f(x) = \max(0, x)$$

Classification (Softmax):

$$P(y_i) = \frac{\sum p^{zi}}{je^{zi}}$$

Existing methods face several critical limitations:

### 5.2 Dynamic Sign Recognition Algorithm

1. Initialize MediaPipe holistic model (detect hands + face + pose).
2. Record coordinates for each frame in the form: [hand_0_x, hand_0_y, hand_0_z, ..., hand_20_z, ear_L_x, ..., shoulder_R_z]
3. Store each frame's data in CSV or NumPy array.
4. For multi-hand gestures, track both hand landmarks independently.
5. Detect gesture start and end automatically using coor- dinate variation thresholds.
6. Aggregate time-series data and train lightweight tem- poral CNN / LSTM network.
7. Predict sign label from motion sequence.

## 6 System Requirements

### 6.1 Hardware

- Intel i5 / Ryzen 5 or higher
- GB RAM minimum
- Optional GPU (GTX 1050 or above)

### 6.2 Software

- Python 3.10+
- TensorFlow / Keras

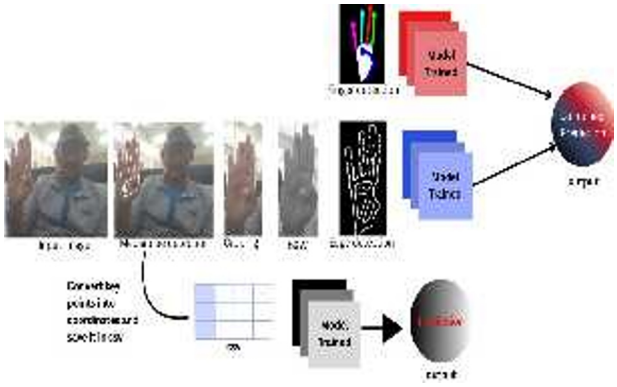

**Fig. 1:** Overview of Identified Research Gaps and Proposed System Flow.

Fig.1 explains step-by-step how the input image is processed from the initial capture through preprocessing (silhouette and finger filters), landmark extraction, model inference, and final predicted label.

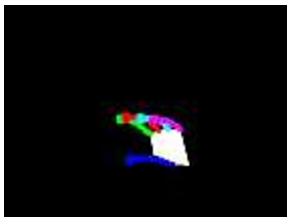

**Fig. 2:** Finger Detection Filtering.
Fig.2 explains the final pre-processing step where individual fingers are detected and color-coded using MediaPipe before model training.

- OpenCV
- MediaPipe
- NumPy, Pandas, Matplotlib

## 7 Methodology and Design

### 7.1 Static Sign Dataset Preparation

The static dataset was built entirely from scratch using a custom Python script for data collection interfacing with MediaPipe Hands; just about 300 raw images were taken for each letter of the alphabet (A–Z, except for J and Z). In application, MediaPipe automatically detected the hand and cropped it onto a consistent frame of 640×480.
Two filters were applied:

- **Silhouette Filter:** Converts image to binary edges for contour emphasis.
- **Finger Detection Filter:** Each finger assigned a fixed

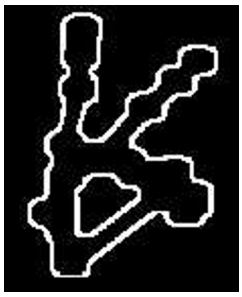

**Fig. 3:** Edge Detection / Silhouette Filtering.
Fig.3 explains the binary edge detection stage (silhouette filter) applied during pre-processing to extract clean hand contours for model training.
RGB color (thumb–blue, index–green, middle–red, ring–cyan, little–pink).
Post cleaning and augmentation, dataset expanded to 120,000 total images.

### 7.2 Dynamic Sign Data Preparation

Dynamic gestures were stored as NumPy coordinate arrays representing temporal motion of hands and face landmarks.
7.2.1 **One-Hand Gestures:** Simple movement signs (e.g., you, me, why, I) saved in .csvformat for easier inspection.
7.2.2 **Two-Hand Gestures:** Cook, car, and finish gestures were all captured using double hand avatars and stored in .npy files for further precision of temporal seg- mentation.

Another observed problem during the data collection arose from the default MediaPipe coordinates, where the origin is placed at the **top-left corner** of the image frame:
Origin: The top-left corner of the camera frame/image **[Fig4]**.
The x-axis increases to the right. The y-axis increases downward.
The z-coordinate represents depth, with the origin ($z = 0$) at the wrist. Smaller $z$ values indicate proximity to the camera.
This caused variations depending on where the signer placed their hands within the frame. To address this, the **left wrist was redefined as the local origin**, and a variable (wrist distance) was added to the .npy dataset to represent the spatial distance between both wrists.
Such adjustment makes it possible to perform gestures anywhere on the screen without compromising visual accuracy. In addition to this, the model guarantees that all interactions by both hands, e.g., touching, overlapping, or symmetric motion, will be captured and respected by the model.

**Fig. 4:** Coordinate normalization.
*Fig.4 explains how we used local coordinates and stored data in it, rather than using global coordinates.*
7.2.3 **Hand-and-Face Gestures:** (e.g., boy, girl) involve motion near face landmarks such as forehead, ears, or lips; planned for future work.
Each CSV/.NPY file stored landmarks as:
[hand_0_x, hand_0_y, hand_0_z,
..., hand_20_z, ear_L_x, ..., shoulder_R_z]
This format ensured consistent representation across gesture types.

## 8 Model Training

### 8.1 Static Sign Recognition Training

Using Tensor Flow and Keras, static models were built. The Finger Detection and Silhouette datasets were consid- ered separately and then fused to create a hybrid inference model.

**Table 1:** Model Training Summary

| Model | Dataset Size | Train / Test Accuracy |
|---|---|---|
| Finger Detection CNN | 84,000 images | 98% / 94% |
| Silhouette CNN | 36,000 images | 96% / 89% |
| Combined Hybrid Model | Fusion (0.6 : 0.4) | – / 91% |

**Training Details:**

- Optimizer: Adam (lr = 0.001)
- Batch size: 32
- Epochs: 30
- Validation Split: 20%
- Loss Function: Categorical Crossentropy

Inference speed achieved: **15–20 FPS** on CPU, making it feasible for real-time use.

### 8.2Dynamic Sign Recognition Training

The phase of dynamic recognition made use of time-series landmark data stored as NumPy arrays. Each gesture se- quence consisted of about 10 frames per sign that were normalized by the mean probability to represent the con- tinuous motion of a handsign.

**One-Hand Dynamic Gestures:**

- Trained on normalized frame sequences extracted from CSV data.
- Achieved an average accuracy of **66%** during testing.
- Data diversity and frame consistency significantly affected model stability.

**Two-Hand Dynamic Gestures:**

- Model trained with limited dataset, high inter-frame noise, and overlap.
- Accuracy during testing reached **32%**.
- Despite poor performance, it served as an experimen- tal baseline for future LSTM integration.

**Hand-and-Face Dynamic Gestures:**

- Dataset collection and preprocessing completed.
- No model training performed yet due to insufficient data volume.

In general, the dynamic gesture recognition stage concluded that a larger dataset requires better sequential stability and more reliable temporal modeling, especially for gestures associated with the use of two hands and linked to face recognition.

## 9 Results and Discussion

Static gesture recognition works very well with dual CNNs. After being integrated for hybrid inference, the model could correctly identify 91% of gestures during real-time testing scenarios.

On the other hand, dynamic recognition was unreliable

- the two-hand/face-linked gesture recognition, in particu- lar - because of noise and limited data availability. CSV files were lightweight, yet visually confirming them was harder, and .npy structured the data much better but added computation overhead.

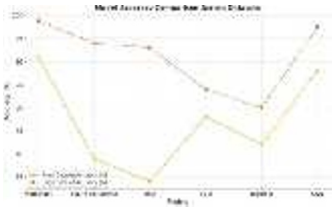

**Fig. 5:** Models Comparison.

*Fig.5 compares different types of model on the same dataset to check how each models perform.*

### 9.1Strengths

- Fully vision-based system, no sensors required.
- Automated annotation reduced manual workload.
- Real-time capable and resource-efficient.

### 9.2Limitations

- Dynamic gestures (especially multi-hand) had incon- sistent accuracy.
- Temporal model training on CSV data was complex due to poor visual interpretability.
- Noise filtering and outlier removal were challenging and not always effective.

## 10 System Architecture

Pipeline Overview:

1. Camera feed input
2. MediaPipe hand/face detection
3. Data preprocessing and coordinate extraction
4. Model selection (Static CNN / Dynamic Sequence Model)
5. Real-time prediction and on-screen display

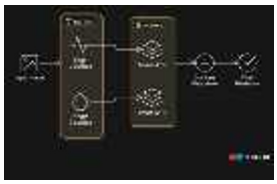

**Fig. 6:** Static block diagram.

*Fig.6 represents simplified block diagram of static hand sign prediction.*

## 11 Conclusion and Future Work

The presented framework for Real-Time Sign Language Translation verified gesture interpretation through a camera without the aid of any specialized hardware. The dual CNN model showed 91% accuracy on static gestures while the dynamic gesture pipeline has laid a base for temporal modeling using the coordinates derived from landmarks.

In any case, all routes towards being completely func- tional face hurdles regarding precision on multi-hand and face-assisted signs. Some of the enhancements in the future could be:

- Expanding dynamic datasets with better diversity and balance.
- Implementing temporal deep learning (LSTM, GRU, or 3D CNN).
- Incorporating face-hand fusion for contextual under- standing.
- Finding way to combine static and dynamic model together.

## References

1. Kodandaram et al. (2021). Sign Language Recognition. DOI:10.13140/RG.2.2.29061.47845.
2. Prashant Verma et al. (2022). Real-Time Sign Language Detection using TensorFlow, OpenCV

and Python. DOI:https://doi.org/10.22214/ijraset. 2022.43439
3. S. Kamble, SLRNet: a real-time LSTM-based sign language recognition system, arXiv preprint arXiv:2506.11154, 2025. [Online]. Available: https://arxiv.org/abs/2506.11154
4. Abougarair et al. (2022). Smart glove for sign language translation. International Journal of Robotics and Automation. 8. 109-117. 0.15406/iratj.2022.08.00253.
5. Mishra, et al. (2023). A Mediapipe-Based Hand Ges- ture Recognition Home Automation System. 1-6. 10.1109/INCOFT60753.2023.10425411.
6. Tejaswini Ananthanarayana et al. 2021. Deep Learning Methods for Sign Language Translation. ACM Trans. Access. Comput. 14(4), Article 22 (2021), 30 pages. https://doi.org/10.1145/3477498
7. Liang, et al. A Survey of Approaches and Techniques. Electronics, 12, 2678 (2023). https://doi.org/ 10.3390/ electronics12122678.
8. Z. Wang et al., Hear sign language: A real-time end-to-end sign language recognition system, in IEEE Transactions on Mobile Computing, 21(7), 2398-2410, doi: 10.1109/TMC.2020.3038303.
9. Kothadiya et al. Sign Language Detection and Recognition Using Deep Learning. Electronics, 11, 1780 (2022). https://doi.org/10.3390/ electronics 11111780
10. Verma, Prashant, and Khushboo Badli. Real-Time sign language detection using TensorFlow, OpenCV and Python. International Journal for Research in Ap- plied Science Engineering Technology 10.V (2022): May2022.
11. Sumit Kumar, Ruchi Rani, Ulka Chaudhari 2024, Real-time sign language detection: Empowering the disabled community, MethodsX, Volume 13, 2024, 102901, ISSN 2215-0161, https://doi.org/10.1016 j.mex.2024.102901.
12. D. Biyani etl "Real time sign language recognition using Yolov5," 2023 IEEE World Conference on AIC, Sonbhadra, India, 2023, pp. 582-588, doi: 10.1109/AIC57670.2023.10263913.
13. R. R. Milinda etl "Sign language detection in real-time applications," 2024 ICCIRT, Coimbatore, India, 2024, pp: 1-4, doi: 10.1109/ICCIRT59484. 2024. 10921810.
14. K. Sharma etl "Recent trends in sign language detec- tion system using machine learning algorithms," 2024 ICICAT, Gorakhpur, India, 2024, pp. 1449-1455, doi: 10.1109 /ICICAT62666. 2024.10923446.
15. Monisha etl "Sign language detection and classifica- tion using hand tracking and deep learning in real- time." (2023).
16. N. Snehalatha etl "Sign language detection using ac- tion recognition LSTM deep learning model," 2024 NMITCON, Bengaluru, India, 2024, pp. 1-6, doi: 10.1109/NMITCON62075.2024.10699301.
17. P. More and S. Sangamkar, "Vision Aid For Blind People Using YOLOV8," 2024 2nd International Con- ference on Networking, Embedded and Wireless Sys- tems (ICNEWS), Bangalore, India, 2024, pp. 1-8, doi: 10.1109/ ICNEWS60873. 2024.10731132.
18. More, P., More, R., Sonavane, S., Kalasgonda, M., More, S. (2025). Optimizing Road Sign Detection with Convolutional Neural Networks (CNN). In: Has- sanien, A.E., Anand, S., Jaiswal, A., Kumar, P. (eds) Innovative Computing and Communications. ICICC 2025.
19. A. Sonawane, V. Dandam, K. Khamkar, T. Wawge and P. More, "Leveraging YOLO for Real-Time Hu- man Detection and Pose Estimation in Live Stream Environments," 2025 International Conference on Computing and Communication Technologies (IC- CCT), Chennai, India, 2025, pp. 1-5, doi: 10.1109/IC- CCT63501.2025.11020018.