IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151178

# Sign Language Translation Using Machine Learning and Artificial Intelligence

**Srinidhi K S[1], Nayan Kumar J S[2], Nikhil S[3], Nikhil M[4], Mohith S Gowda[5]**

Assistant. Professor, Dept. of CSE, SJBIT, Bengaluru, Karnataka, India[1]

Dept. of CSE, SJBIT, Bengaluru, Karnataka, India[2-5]

**Abstract:** Communication between sign language speakers and non-sign language speakers relies on human interpretation and/or sensors, both restricting frequent usage. In this paper, we describe the design of a webcam-based application capable of recognizing signs in sign language through a Convolutional Neural Network (CNN) and showing them as text and also speech as a planned feature. The architecture consists of dataset formatting and labeling, pre-processing of images, training of CNNs with the help of the Adam optimizer and categorical cross-entropy loss function and recognition process with OpenCV, TensorFlow and Keras running on regular Windows laptop. Signs or gestures are to be put together into simple sentences with planned features such as suggesting the next character and conversion into speech. The project is in an intermediate phase where 5 Kaggle datasets are examined for appropriateness and around half of the training process is complete. There are no performance measurements yet: only the evaluation procedure with accuracy, precision, recall, F1 score and confusion matrices is stated and the other limitations are discussed.

**Keywords:** Sign Language Recognition, Computer Vision, Convolutional Neural Network, Deep Learning, Gesture Recognition, Text to Speech.

## I. INTRODUCTION

This is a language that uses vision where the meaning is conveyed by hand shapes, movements, orientations, placements, facial expressions, and body postures [3]. To many individuals who are either deaf or hearing impaired, this is their natural form of communication, but the majority of people with hearing ability have no knowledge of sign language. This makes interactions within the medical, legal, and education sector need an interpreter. This increasing need has seen the introduction of video remote interpretation, which is however quite imperfect [1].

The automatic recognition process goes through two paths. The first type uses sensor-based systems, where data gloves or motion trackers can be used, which can provide accurate finger information but are uncomfortable to use. In vision-based systems, normal cameras can be used, and information can also be used from color and texture, but they need to deal with lighting conditions, segmentation, background noise, and occlusions. This has been made possible due to the emergence of Convolutional Neural Networks [3].

The reason why the project was initiated is because there is a need for an affordable recognizer that can run on a typical computer using a web camera and not require special tools like gloves, depth camera or even a workstation equipped with a graphics processing unit. The aim of the project is to create a computer vision based program that would capture sign language gestures through a web camera, classify them through CNN and then present the output in text form as well as verbally at some point.

The document provides three main elements: a comparative study of eight vision-based papers and a survey that concludes in a specific research gap; a four-module approach to webcam-based CNN recognition producing textual output; and an evaluation strategy determined before any results are acquired. The second section provides an overview of related work, the third is on methodology, the fourth is on implementation and evaluation, the fifth section deals with limitations, responsible usage and future work, while the sixth one concludes the paper.

## II. RELATED WORK

Table I summarizes the articles discussed above; figures in this table are results presented by the cited authors from their data, and not results of our project. The following discussion is focused on what each of the studies suggests to us about our design.

 This work is licensed under a Creative Commons Attribution 4.0 International License 

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151178

TABLE I. Overview of studies reviewed (published results of the mentioned authors)

| Ref. | Technique | Data / task | Reported result | Main limitation |
|---|---|---|---|---|
| [2] | Skin colour, edges, Fourier descriptors, vector quantization, template matching | Self-made videos: numbers, alphabets, phrases | 92.91% overall (conclusion) | Hand-crafted features; similar signs confused |
| [4] | Haar-like AdaBoost, skin colour, frame difference, tracking | Hand detection in three test videos | Up to 99.9% detection rate | Fails on skin-coloured background; weak in poor light |
| [5] | Embedded board, Canny edges, convex hull, HMM matching | Numbers and finger alphabets | No accuracy given | Hand-crafted contour features |
| [6] | CLAHE and HSV segmentation for ROI, CNN | Numerals 0–9 | 99.56% same subject; 97.26% low light | Thresholds need retuning; numerals only |
| [7] | Two-layer CNN, SGD vs Adam | Six finger-count classes | 99.12% (SGD); 99.51% (Adam) | Very small class set |
| [8] | Fine-tuned pretrained CNNs, fuzzy text matching | 32 Yemeni sign classes (Kaggle) | 98.78% best model | Isolated signs; small vocabulary |
| [1] | Inception CNN plus LSTM | 100 ASL signs, one signer | 90–93% (softmax features) | Drops for unseen skin tone, clothing |
| [9] | MediaPipe key points, VGG-19, Bi-LSTM | Recognition and translation, ISL sentences | Above 95% classification | GPU workstation required |
| [10] | Pretrained denoising language model | Gloss-to-text benchmarks | +4.37 BLEU over baseline (Phoenix2014T) | Copying errors; noisy-input sensitivity |

*A. Classical and Template-Based Pipelines*

In their work, Badhe and Kulkarni [2] (2015) recorded their own videos of Indian Sign Language, followed by hand tracking using cropping, frame differencing, YCbCr skin color detection, and Canny edge detection, after which they compared Fourier descriptor codebook entries based on Euclidean distance. They point out that the errors mainly arise due to lookalike signs like C and O, and M, N, and R. The pipeline clearly explains how crucial it is for the accuracy of sign language recognition, and yet every step in the pipeline is done by hand-crafting and is tuned according to the recording environment. Nath and Anu [5] (2017) developed the same pipeline on ARM Cortex-A8 with USB camera and OpenCV along with a speech to text application for Android phones via Bluetooth connectivity.

*B. Hand Detection and Region of Interest*

Wu & Nagahashi [4] (2013) integrated Haar-based AdaBoost, HSI model of skin, YCbCr model of skin, and frame difference methods to detect and track both hands. The detector performed better in handling cluttered background than a naive AdaBoost method, however it failed to work in situations where hands occluded other skin areas, on skin-colored background, and poor illumination conditions. This research highlights the importance of an explicit ROI selection step with caution of color-based ROI.

*C. CNN-Based Static Gesture Recognition*

Sajanraj and Beena [6] (2018) used a skin segmentation ROI (CLAHE in LAB space, blurring, HSV thresholds, largest contour) and Keras CNN for ISL numerals. The performance degraded with decreasing brightness levels, while threshold parameters had to be tweaked manually according to changes in the light intensity. Suresh et al. [7] (2019) maintained the very simple structure of their CNN (two convolutional layers and dropout), comparing SGD and Adam; Adam provided slightly better results, and precision, recall, F1 score and confusion matrix for every class are given in the article, thus this type of evaluation is adopted in our work. Mosleh and Gumaei [8] (2024) fine-tuned five CNNs pretrained on ImageNet and tested on the Yemeni sign language dataset hosted on Kaggle in an Android application with fuzzy mapping from text to sign language, also showing how computationally expensive large networks are.

*D. Temporal, End-to-End and Language-Level Methods*

The authors of the study by Bantupalli and Xie [1] (2018) used an LSTM in which they fed Inception features to classify sequences of frames. Accuracy decreased if the skin color or clothing of signers was different from the training

 This work is licensed under a Creative Commons Attribution 4.0 International License 

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151178

data, and they suggested hand isolation with an ROI. The study by Natarajan et al. [9] (2022) involved MediaPipe keypoints, VGG-19, and Bi-LSTM, creation of sign video, and the glove-free ISL sentence dataset. This research was performed on a GPU workstation. The study conducted by Chen and Wang [10] (2026) revealed that the fine-tuning of a denoising language model leads to better gloss-to-text translation while at the same time pointing out the problem of copying errors and vulnerability if the gloss comes from a noisy recognizer. According to the literature review by Kamal et al. [3] (2019), continuous signing is not only a sequence of signs, annotated corpora are rare, and low-cost systems should work with regular cameras.

*E. Research Gap*

Combined, these papers provide evidence that CNN classifiers work well for classifying static gestures when the conditions are controlled [6]–[8], that their performance drops if there are changes in lighting, background, skin color or signers [1], [4], [6], and that producing sentence-level outputs requires temporal or linguistic modeling in addition to classification [3], [9], [10]. It is less common to find literature describing a low-cost, end-to-end system connecting preprocessing, CNN classification, text generation, and speech using webcams with the diversity of datasets and the variability of the environment as key inputs into the design. This is what this project aims to fill the gap in.

## III. PROPOSED SYSTEM AND METHODOLOGY

*A. System Architecture*

Figure 1 depicts the process that converts a video frame taken by the webcam to speech through eight steps. Solid boundaries represent steps which are currently being worked on and the dotted boundaries indicate functions that are planned for inclusion. The pipeline is modular in nature such that the ROI Extraction step, the Classifier step and the Text Layer step can each be swapped independently of one another.

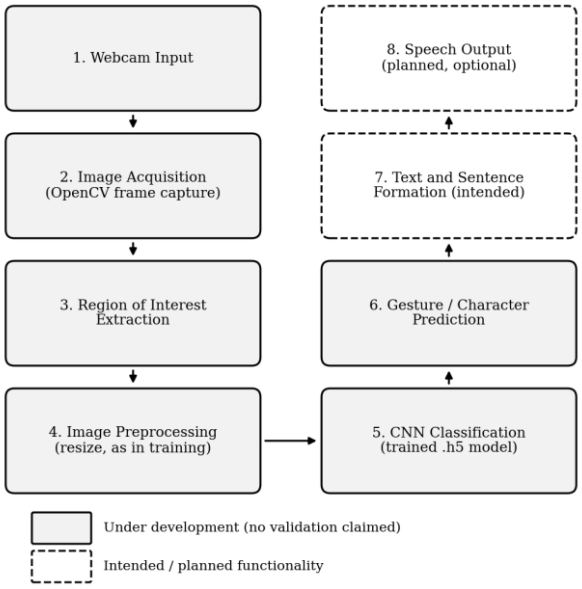



Figure 1. Suggested process from video capture to speech synthesis.

*B. Module 1: Dataset Creation*

There are two sources for the gesture images: webcam images or publicly available data. There are five publicly available datasets on Kaggle, which are currently being analyzed for use. Because they have no verified classes, image sizes, background scenes, and signers, these datasets are not considered to be mergeable. Every chosen dataset is cropped to the desired area, resized to a certain size and put into a folder corresponding to each class.

*C. Module 2: Preprocessing and Label Encoding*

The loader goes through the dataset folders and extracts the images while assigning their labels by using the names of the folders. A label encoder transforms the string labels into integers and these integers are then transformed into one hot encodings for categorical cross entropy loss. The images and labels are collected into arrays and the encoder mapping is saved with the model.

*D. Module 3: CNN Model Training*

CNN architecture matches well with hand gesture images since the kernels used for local edges and contour recognition have smaller sizes with shared weights, while pooling allows for some shift in the image. The function of each layer is

DOI: 10.17148/IJARCCE.2026.151178

summarized in Table II. The number of filters, sizes of kernels, and size of the input image are yet to be determined during training. Convolution operation is computed by the following formula:

TABLE II. Function of each layer in the CNN suggested

| Layer | Operation | Purpose |
|---|---|---|
| Input | Fixed-size image tensor | Same size for training and live frames |
| Convolution + ReLU | Learned kernels, max(0, x) | Local edge and contour features |
| Max-pooling | Maximum over small windows | Down-sampling; shift tolerance |
| Fully connected | Dense weighted sums | Combine features per class |
| Softmax output | Normalized exponentials | One probability per class |

$$F_m = \max(0, \Sigma_n K_{m,n} * X_n + b_m) \quad (1)$$

Where $K$ is kernel, $*$ denotes convolution, and $b$ is bias. The output layer computes for the logit vector $z$ that spans $C$ classes as

$$\hat{y}_k = \exp(z_k) / \Sigma_{j=1\ldots C} \exp(z_j) \quad (2)$$

Training uses the Adam optimizer to minimize categorical cross-entropy as described for this project. For an example with a one-hot label vector y,

$$L = -\Sigma_{k=1\ldots C} y_k \log \hat{y}_k \quad (3)$$

After each epoch, there is a validation step. Solutions that can be used in the case that the validation error starts to deviate from the training error include dropout [7], data augmentation, and early stopping [8]. An 80-20 train-test ratio is suggested, where a fraction of the training set is held out for validation.

*E. Module 4: Real-Time Recognition and Text Construction*

The ROI is extracted for each frame using OpenCV, followed by resizing and preprocessing as done on the training images, and the trained model determines the class.

Accepted characters would be appended to a text buffer. For next-character suggestion, a word list would be matched against the current prefix and the likeliest completions displayed; the word list has not yet been chosen. Speech output would pass the finished text to a Python text-to-speech engine yet to be selected. These functions are intended, not implemented. Because the CNN sees one frame at a time, the system performs static gesture classification; dynamic signs need temporal models such as LSTMs [1], [9]. Hence the system is character/gesture recognition with text construction, not sign language translation in the full sense.

## IV. IMPLEMENTATION AND EVALUATION

*A. Environment and Status*

The stated hardware is an Intel Core i5 processor, 8 GB RAM, a 256 GB SSD, an HD webcam (720p/1080p) and optional Intel Iris Xe graphics for training. Software comprises Windows, Python/Anaconda and Visual Studio Code, with OpenCV for acquisition and preprocessing and TensorFlow with Keras for the network. Compared with the GPU workstation of [9], this is modest, so a compact CNN is preferred.

Completed so far is the collection of five Kaggle datasets. Under way are the suitability study and CNN training, which was reported about 50% complete at the time of the project presentation. Real-time testing, suggestions, sentence formation and speech remain intended.

*B. Dataset Audit and Split Policy*

Before any merging, each Kaggle dataset will be audited for its class inventory, image size and colour mode, background type, number of signers, duplicate images, label noise and licence terms. Classes with the same meaning but different names will be unified, and datasets whose label sets cannot be reconciled will be used separately. To avoid leakage, near-duplicates will be removed before splitting, augmentation will be applied to the training portion only, and, where signer information exists, the split will keep signers apart.

*C. Evaluation Metrics*

Evaluation uses a held-out test set unseen during training. With $N_{correct}$ correct predictions out of $N_{total}$:

$$\text{Accuracy} = (N_{correct} / N_{total}) \times 100\% \quad (4)$$

$$P_c = \text{TP} / (\text{TP} + \text{FP}), \quad R_c = \text{TP} / (\text{TP} + \text{FN}) \quad (5)$$

$$F1_c = 2P_cR_c / (P_c + R_c) \quad (6)$$

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151178

for class *c*, with TP, FP and FN the true positives, false positives and false negatives. Class values are macro-averaged. Table III defines each metric's purpose; nothing has been measured yet.

TABLE III. Planned evaluation metrics and current status

| Metric | Purpose | Status |
|---|---|---|
| Accuracy | Overall share of correct predictions | Not yet evaluated |
| Precision | Reliability of each predicted class | Not yet evaluated |
| Recall | Coverage of each true class | Not yet evaluated |
| F1-score | Balance of precision and recall | Not yet evaluated |
| Confusion matrix | Shows which gestures are confused | Not yet evaluated |
| Latency | Time per frame on target hardware | Not yet evaluated |

*D. Experiment Plan*

Robustness is judged under controlled variation, summarized in Table IV, because the reviewed work shows strong sensitivity to illumination, background and signer [1], [4], [6]. Latency will be timed per stage (capture, ROI, preprocessing, inference, rendering) with a high-resolution clock, and the mean and 95th percentile over many frames will be reported. Real-time suitability will be judged against a frame budget set after measurement; for reference, [6] had to skip frames to keep up.

TABLE IV. Planned experiments (none completed)

| ID | Condition varied | Purpose |
|---|---|---|
| E1 | Random 80:20 split, single dataset | Baseline accuracy and loss |
| E2 | One signer held out entirely | Signer-independent performance |
| E3 | Bright, normal and dim lighting | Sensitivity to illumination |
| E4 | Plain and cluttered backgrounds | Sensitivity to ROI failure |
| E5 | Hand distance and position | Robustness to placement |
| E6 | Train on one Kaggle set, test on another | Cross-dataset generalization |
| E7 | Per-stage timing on stated hardware | Real-time feasibility |

## V.RESULTS

The results are presented through a qualitative assessment of the implemented system, covering hand landmark detection, gesture classification, text construction, and speech output. The following tables summarize the functionality of each module. This assessment describes the system’s operation; quantitative recognition accuracy and real-time performance remain to be validated.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151178

TABLE I. Dataset and Evaluation details

| **Parameter** | **Value** |
|---|---|
| Sign classes | 35: A–Z and digits 1–9 |
| Total valid samples | 41,792 |
| Features per sample | 84 landmark coordinates |
| Classification model | Fully connected neural network (MLP) |
| Evaluation subset size | 8,359 samples |
| Subset selection | Stratified 20% split, random seed 42 |

TABLE II. Preliminary Sign Classification Performance

| **Metric** | **Result** |
|---|---|
| Accuracy | 99.9880% |
| Macro precision | 99.9881% |
| Macro recall | 99.9878% |
| Macro F1-score | 99.9880% |
| Correct predictions | 8,358 |
| Incorrect predictions | 1 |

## VI. DISCUSSION, LIMITATIONS AND FUTURE WORK

*A. Benefits and Feasibility*

If it works as designed, the system would be a low-cost, glove-free aid giving immediate text feedback on common hardware, and its modular structure lets individual stages be improved separately. Feasibility on a Core i5 laptop with 8 GB RAM depends on the measured latency and on keeping the network compact.

*B. Limitations and Threats to Validity*

Lighting, hand position, background and signer variation can shift the input distribution and break ROI extraction [4], [6], and look-alike gestures will probably be confused, as in [2]. Vocabulary is capped by the datasets, whose diversity is not yet known. Isolated gestures differ from continuous signing, which carries motion and grammar [3]. Building grammatical sentences from recognized characters is hard, and language-model approaches stay sensitive to recognition errors [10]. A random split of images from the same signers overstates performance, which is why held-out signers and cross-dataset tests (E2, E6) are planned. Results may also depend on the chosen ROI method and on the specific webcam.

*C. Responsible Use*

A recognizer of this kind should not replace interpreters in medical or legal settings, where errors carry high cost. Deaf and hard-of-hearing users should be involved in testing and in judging usability. Webcam frames show faces and surroundings, so images should be processed locally and not stored without consent.

*D. Future Work*

Planned steps are: (1) finishing training and quantitative testing; (2) better gesture segmentation and robustness; (3) a larger vocabulary; (4) temporal models such as CNN-LSTM if dynamic gestures are required [1], [7]; (5) NLP-based word suggestion and sentence formation; and (6) building and evaluating text-to-speech.

## VI. CONCLUSION

We have outlined a webcam-based sign recognition system built on a CNN trained with Adam and categorical cross-entropy, using OpenCV, TensorFlow and Keras on ordinary hardware. Four modules cover dataset creation, preprocessing, training and live recognition with text output, while sentence formation and speech remain intended.



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151178

Datasets are still under study and training is incomplete, so the contribution at this stage is the design and a complete evaluation plan rather than performance figures. The immediate task is to finish training and run experiments E1–E7.

## REFERENCES

[1] K. Bantupalli and Y. Xie, "American Sign Language recognition using deep learning and computer vision," in Proc. IEEE Int. Conf. Big Data, 2018, pp. 4896–4899.
[2] P. C. Badhe and V. Kulkarni, "Indian Sign Language translator using gesture recognition algorithm," in Proc. IEEE Int. Conf. Comput. Graph., Vis. Inf. Security (CGVIS), 2015, pp. 195–200.
[3] S. M. Kamal, Y. Chen, S. Li, X. Shi, and J. Zheng, "Technical approaches to Chinese sign language processing: A review," IEEE Access, vol. 7, pp. 96926–96935, 2019, doi: 10.1109/ACCESS.2019.2929174.
[4] S. Wu and H. Nagahashi, "Real-time 2D hands detection and tracking for sign language recognition," in Proc. 8th Int. Conf. Syst. of Syst. Eng., Maui, HI, USA, Jun. 2013, pp. 40–45.
[5] G. G. Nath and Anu V. S., "Embedded sign language interpreter system for deaf and dumb people," in Proc. Int. Conf. Innovations Inf., Embedded Commun. Syst. (ICIIECS), 2017.
[6] T. D. Sajanraj and M. V. Beena, "Indian Sign Language numeral recognition using region of interest convolutional neural network," in Proc. 2nd Int. Conf. Inventive Commun. Comput. Technol. (ICICCT), 2018, pp. 636–640.
[7] S. Suresh, M. H. T. P., and M. H. Supriya, "Sign language recognition system using deep neural network," in Proc. 5th Int. Conf. Adv. Comput. Commun. Syst. (ICACCS), 2019, pp. 614–618.
[8] M. A. A. Mosleh and A. H. Gumaei, "An efficient bidirectional Android translation prototype for Yemeni sign language using fuzzy logic and CNN transfer learning models," IEEE Access, vol. 12, pp. 191030–191045, 2024, doi: 10.1109/ACCESS.2024.3512455.
[9] B. Natarajan et al., "Development of an end-to-end deep learning framework for sign language recognition, translation, and video generation," IEEE Access, vol. 10, pp. 104358–104374, 2022, doi: 10.1109/ACCESS.2022.3210543.
[10] S. Chen and Y. Wang, "Improving sign language-gloss translations with pretrained models," IEEE Access, vol. 14, pp. 39461–39470, 2026, doi: 10.1109/ACCESS.2026.3665295.

 This work is licensed under a Creative Commons Attribution 4.0 International License 