



Sign Language to Text and Audio Converter

Dr. C. P. Divate¹, N. R. Bhokare², Sankalp Shivprasad Takawade³, Shreya Sagar Shette⁴, Pirkhan Malik Pathan⁵, Arsalan Asif Sharikmaslat⁶

¹Dean, Department of Computer Engineering, ATS. Sanjay Bhokare Group of Institute, Faculty of Polytechnic, Miraj, Maharashtra, India.

²HOD, Department of Computer Engineering, ATS. Sanjay Bhokare Group of Institute, Faculty of Polytechnic, Miraj, Maharashtra, India.

^{3,4,5,6}Department of Computer Engineering, ATS. Sanjay Bhokare Group of Institute, Faculty of Polytechnic, Miraj, Maharashtra, India.

To Cite this Article: Dr. C. P. Divate¹, N. R. Bhokare², Sankalp Shivprasad Takawade³, Shreya Sagar Shette⁴, Pirkhan Malik Pathan⁵, Arsalan Asif Sharikmaslat⁶, "Sign Language to Text and Audio Converter", Indian Journal of Computer Science and Technology, Volume 05, Issue 03 (September-December 2026), PP: 413-416.



Copyright: ©2026 This is an open access journal, and articles are distributed under the terms of the [Creative Commons Attribution License](#); Which Permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Abstract: Communication between sign-language users and people who do not understand sign language is difficult when an interpreter is not available. This paper proposes the Sign Language to Text and Audio Converter, a low-cost, camera-based assistive system that recognizes static hand gestures for English alphabet and number signs and converts them into readable text and spoken audio. In the proposed design, a live webcam feed is processed with OpenCV and MediaPipe to detect and isolate the hand region. The extracted region is resized, converted to grayscale and normalized, and then classified by a Convolutional Neural Network (CNN) built with TensorFlow/Keras and trained on a labelled sign-language image dataset. Recognized symbols are buffered into words and sentences, displayed in a simple interface, and converted into speech by a text-to-speech (TTS) engine, so that the user can choose text or voice output. The system needs only a standard webcam and avoids sensor gloves or special hardware. It is organized into four modules: study and dataset preparation, hand detection and preprocessing, CNN training and classification, and text/audio output with testing. This paper presents the architecture, methodology and planned evaluation of the system; the implementation is in progress, and no experimental results are claimed.

Key Word: Sign Language Recognition, Convolutional Neural Network (CNN), Computer Vision, Hand Gesture Detection, OpenCV, MediaPipe, Text-to-Speech (TTS), Assistive Technology

I. INTRODUCTION

Communication is essential for education, employment, healthcare and social participation. People who communicate mainly through sign language often face difficulty when they interact with people who do not know it, particularly when no interpreter is available. This communication gap is an area where computer vision and machine learning can provide practical assistive solutions.

Sign language recognition is a computer-vision task in which visual information from hand gestures is captured and analyzed to determine the intended sign. Recent advances in deep learning, particularly Convolutional Neural Networks (CNNs), allow a model to learn visual features from hand-sign images automatically and classify them. Earlier studies have used CNN-based models with webcam input and image preprocessing for alphabet recognition [1], [2], and have combined recognition with text and speech output [3], [4].

This paper proposes the Sign Language to Text and Audio Converter. A camera captures the user's hand gestures, computer-vision techniques isolate the hand in each frame, and a trained CNN classifies the gesture into an English alphabet letter or a number. The recognized symbols are displayed as readable text and can also be spoken through a text-to-speech engine. The primary interaction needs only a standard camera and computing device, rather than specialized sensor gloves, which makes the approach affordable and deployable in homes, schools and public service points.

The main contributions of this work are: (i) a proposed modular, webcam-based pipeline from gesture capture to text and audio output; (ii) a design that combines MediaPipe hand detection with CNN classification and a text buffer so that letters form words and sentences; and (iii) a planned evaluation covering accuracy, precision, recall, F1-score, confusion matrix and real-time latency under varied lighting and backgrounds. The scope is limited to static alphabet and number gestures, not continuous sign-language translation.

The rest of the paper is organized as follows. Section II reviews the literature. Section III describes the methodology as four modules. Section IV discusses the expected benefits, planned evaluation and feasibility. Section V concludes.

II. LITERATURE REVIEW

Triwijoyo et al. [1] proposed a deep learning approach for alphabetic sign-language recognition using a seven-layer CNN trained on an American Sign Language (ASL) alphabet dataset of 27 categories and 87,000 hand-gesture images. The images were preprocessed with background correction and resizing, the data was split into training and validation sets, and the model was tested on hand signs captured through a webcam. The study shows that preprocessing and CNN classification can support real-time alphabet recognition, but it produces only the recognized sign and gives no speech output.

Adhikari et al. [2] compared a customized CNN with four pre-trained models (VGG16, InceptionV3, ResNet50 and DenseNet121) for ASL classification. MediaPipe was used to detect hand landmarks and extract the hand region from live camera frames, followed by resizing and normalization. The customized CNN gave the highest testing accuracy among the models evaluated. This supports the choice of a compact CNN together with MediaPipe for live-camera recognition.

Palla [3] developed a sign-language recognition and text-to-speech translation system, studying different CNN configurations and evaluating them with accuracy, precision, recall and F1-score. The work discusses real-world challenges such as differences in sign-language dialects, lighting conditions, uncontrolled environments and variation between users, and stresses real-time translation and accessibility.

Fonya and Worrell [4] presented a real-time assistive system that translates static sign gestures into text and audible speech. A CNN trained on the Sign Language MNIST dataset classifies gestures captured by a webcam, OpenCV handles video capture and image processing, MediaPipe provides hand detection and landmark tracking, and pyttsx3 produces the speech. The authors also note lighting variation, background clutter, hand orientation and differences in hand size and signing style as practical challenges. This pipeline is the closest to our proposed system.

Wilson and Steephen [5] proposed a real-time system that recognizes ASL gestures captured by a webcam and converts them into text using a CNN, targeting the communication gap between deaf or hard-of-hearing people and others.

Research gap. The reviewed studies show that webcam-based CNN recognition is feasible, but they differ in what they provide: some stop at recognized text, some add speech, and several report that lighting, background and signing style affect performance. The proposed work does not claim a new recognition technique. Its aim is a complete, low-cost and modular prototype that joins hand detection, preprocessing, CNN classification, word-level text buffering, a choice of text or audio output and a user interface, and that is planned to be tested explicitly under varied lighting, backgrounds and hand positions.

III. METHODOLOGY

The proposed system follows a modular pipeline in which each stage processes the output of the previous one (Fig. 1). It combines computer vision, image preprocessing, CNN-based deep learning, real-time recognition and text-to-speech, and is organized into four modules (Table I).

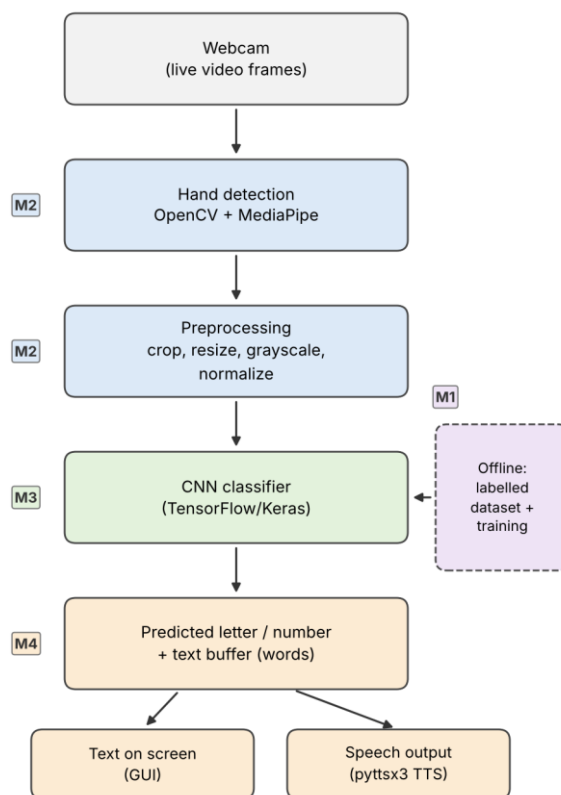


Fig. 1. Block diagram of the proposed Sign Language to Text and Audio Converter and its four modules (M1–M4). M1 prepares the dataset and trains the model offline; M2–M4 run in real time.

Module	Function	Key tools
1. Study	Literature study, dataset collection, requirement analysis	Python, TensorFlow/Keras
2. Detection	Webcam capture, hand detection, image preprocessing	OpenCV, MediaPipe, NumPy
3. CNN model	Training, validation and real-time classification of gestures	TensorFlow/Keras
4. Output	Text buffer, speech output, interface, testing	pyttsx3 or gTTS, Tkinter/PyQt

Table I. Summary of the Four Modules

A. Module 1: Literature Study, Dataset Collection and Requirement Analysis

This module establishes the research basis and the technical requirements of the system. A literature study covers sign-language recognition, computer vision, CNN-based image classification, hand detection and text-to-speech conversion.

Image dataset is selected, focusing on English alphabet and number signs. It is examined for the number of classes, the number of images per class, image size and format, image quality, label consistency, variation in hand position and orientation, and suitability for CNN training. Datasets such as Sign Language MNIST, which holds labelled grayscale hand-sign images, have been used for static gesture classification [4].

Requirement analysis: Functional requirements are capturing gestures through a webcam, detecting the hand, processing the image, classifying the gesture, displaying the recognized symbol as text, converting text to audio and providing a simple interface. Non-functional requirements are real-time or near-real-time response, reasonable recognition accuracy, a simple interface, low-cost hardware and reliable operation under suitable lighting.

Hardware and software: The system needs a computer or laptop with a webcam and speakers or headphones. The software environment consists of Python, OpenCV, MediaPipe, TensorFlow/Keras, NumPy and a TTS library such as pyttsx3.

B. Module 2: Hand Detection and Image Preprocessing

Camera-based image capture: The webcam continuously supplies video frames, which OpenCV reads and processes one at a time. The general flow is: webcam, video frame, hand detection, hand-region extraction, preprocessing. Processing continues until the user stops the camera or exits the application.

Hand detection: MediaPipe is used to detect the hand and track its landmarks [6]. This separates the hand from other objects in the frame and gives a bounding region that is cropped for classification.

Image preprocessing: The CNN expects input in the same format as its training data, so each cropped hand region is resized, converted to grayscale, normalized and reshaped according to the selected dataset and CNN input size.

Handling practical variations: Webcam images are affected by lighting, background objects, hand orientation, hand position, distance from the camera and differences between users, which the literature identifies as the main challenges for real-time recognition [3], [4]. Cropping to the detected hand and normalizing the image reduce the effect of background and brightness, and testing will cover these conditions explicitly.

C. Module 3: CNN Training and Gesture Classification

Dataset preparation: The preprocessed dataset from Module 1 is divided into training, validation and testing portions, so that performance is measured on images the model has not seen during training.

CNN architecture: A CNN is suitable for image classification because it learns visual features directly from images. The proposed network contains convolution layers with activation functions, pooling layers, a flattening layer, fully connected layers and an output layer with one unit per gesture class. The exact number of layers and filters will be fixed during experimentation.

Model training: The model learns the relation between each input hand image and its class label through repeated prediction, loss calculation and weight update. Adam optimization [7] and a categorical cross-entropy loss are planned for multi-class classification. Training and validation accuracy and loss are monitored during training, and test accuracy and loss are recorded afterwards.

Real-time classification: The trained model is integrated with the webcam application. Each suitable frame passes through Module 2, and the CNN classifies the processed hand image, giving the chain: webcam, MediaPipe/OpenCV, preprocessing, CNN, gesture class, alphabet or number. This module is the core of the system.

D. Module 4: Text and Audio Output, Testing and Report

Text output: The class predicted by the CNN is mapped to its alphabet letter or number and shown on the interface. A buffer accumulates successive recognized symbols so that they form words and sentences, with a delay or stability check so that one held gesture is not entered many times.

Audio output: The recognized text is passed to a TTS engine such as pyttsx3 or Google Text-to-Speech, which generates computer speech. The overall flow is: hand gesture, CNN, recognized symbol, text, TTS, audio. The user can choose text only or text with voice.

User interface: A lightweight interface built with Tkinter or PyQt, or a web front end, shows the live camera feed, the detected gesture and the recognized text, with controls to start the camera, speak the text, clear it and exit. It is kept simple, because the main purpose of the project is gesture recognition and communication.

System testing: After integration, the system is tested for functions of each module, recognition of different alphabet and number gestures, continuous real-time webcam operation, correctness of text and audio output, and performance under different lighting, backgrounds, hand positions and orientations. The results will be compiled in the final project report.

The system is planned for implementation in Python using OpenCV, MediaPipe, TensorFlow/Keras, NumPy and a TTS library, following an Agile approach in which detection accuracy and output quality are tested and refined iteratively.

IV. EXPECTED BENEFITS AND EVALUATION PLAN

A. Expected Benefits

Based on the design, the system is expected to: recognize alphabet and number gestures from live webcam input; display them as readable English text; provide speech output as an additional communication channel; run at low cost with a standard webcam and open-source software; and reduce the communication barrier between sign-language users and people who do not know sign language in education, workplaces and public service points. These are design goals, not measured results.

B. Planned Evaluation

After implementation, the system will be evaluated on the following measures:

- recognition accuracy on a held-out test set of alphabet and number gestures;
- precision, recall and F1-score for individual gesture classes;
- a confusion matrix to show which signs are confused with one another;
- real-time response, i.e., the time taken to detect, classify and display a gesture;
- correctness of the generated text and speech output;
- performance under different users, lighting, backgrounds and hand positions.

C. Feasibility and Scope

The project is technically feasible because it uses established libraries (Python, OpenCV, MediaPipe, TensorFlow/Keras, NumPy) and economically feasible because it needs only a computer, a webcam and free software. It is operationally feasible because the interface needs only basic computer knowledge, and time feasible because it is limited to static English alphabet and number gestures rather than continuous sign-language translation. Recognition may still vary under poor lighting, complex backgrounds, unusual hand orientations and individual signing styles, and the system does not yet cover continuous signing or full sentences in sign grammar.

V. CONCLUSION

This paper proposed the Sign Language to Text and Audio Converter, a webcam-based system that detects hand gestures with OpenCV and MediaPipe, classifies English alphabet and number signs with a CNN, and converts the result into readable text and speech. The four-module design lets each stage be developed, tested and improved independently, and uses only commonly available hardware and open-source software, so it can serve as a practical assistive prototype.

Future work includes completing the implementation, evaluating the system as planned in Section IV, and extending it to larger sign vocabularies, continuous gesture recognition, better robustness to environmental conditions and more advanced assistive communication features.

Acknowledgment

The authors thank their project guides, Dr. C. P. Divate and Ms. N. R. Bhokare, and the Department of Computer Engineering, Sanjay Bhokare Group of Institute, Miraj, for their guidance and support.

REFERENCES

1. B. K. Triwijoyo, L. Y. R. Karnaen, and A. Adil, "Deep learning approach for sign language recognition," *Jurnal Ilmiah Teknik Elektro Komputer dan Informatika (JITEKI)*, vol. 9, no. 1, pp. 12–21, 2023, doi: 10.26555/jiteki.v9i1.25051.
2. S. K. Adhikari, P. Neupane, S. Mainali, U. Regmi, and P. Chapagain, "American sign language classification using CNNs: A comparative study," in *Proc. KEC Conference*, 2024, doi: 10.3126/injet.v1i2.66704.
3. V. Palla, "Sign language recognition and text-to-speech translation," M.Sc. research project, Dept. of Artificial Intelligence, National College of Ireland, 2023.
4. B. Fonya and C. Worrell, "Real-time sign language gestures to speech transcription using deep learning," *arXiv:2508.12713*, 2025.
5. Wilson and L. Steephen, "Sign language to text conversion using CNN," *Indian Journal of Data Mining*, vol. 4, no. 1, pp. 9–12, May 2024, doi: 10.54105/ijdm.A1634.04010524.
6. F. Zhang et al., "MediaPipe Hands: On-device real-time hand tracking," *arXiv:2006.10214*, 2020.
7. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2015.