

Hand Sign Detection Using Python And Deep Learning

Mr. Vetrivel S

Assistant Professor

Department of IT and Cognitive Systems Sri Krishna Arts and Science College

Yanis Reginald S, Mythika S, Subaharini S

Department of IT and Cognitive Systems Sri Krishna Arts and Science College Coimbatore, India

Abstract— Sign-based gestures form a rich, non-verbal mode of interaction and hold particular value for people who are deaf, hard of hearing, or speech-impaired and who use sign language as part of everyday communication. This work describes a real-time handgesture recognition pipeline that classifies gestures captured live from a webcam by combining classical computer-vision techniques with a deep convolutional network. Video frames are read and processed using OpenCV, while hand localization and the extraction of twentyone skeletal landmark points are handled by the CVZone library, which itself wraps Google's MediaPipe hand-tracking solution. Recognition of the gesture class is carried out by a TensorFlow classifier. Prior to inference, each frame is cropped around the detected hand, resized to a fixed square canvas, padded against a neutral background, and adjusted so that the original aspect ratio of the hand is preserved. Evaluation across unit, integration, and end-to-end system tests confirmed stable, low-latency operation under everyday indoor lighting. Beyond its immediate accessibility use case, the pipeline is well suited to touch-free interfaces more broadly, with possible extensions into education, robotics, and ambient/smart-device control. The paper closes by outlining planned extensions covering continuous (dynamic) gesture tracking, broader multilingual signlanguage coverage, and porting the system to mobile and IoT targets.

Index Terms—Hand Gesture Recognition, Computer Vision, MediaPipe, TensorFlow, Deep Learning, OpenCV, Sign Language, MobileNetV2

I. INTRODUCTION

Automatic hand-gesture recognition sits at the intersection of computer vision and applied machine learning, and it gives machines a way to interpret a person's hand movements as they happen. People who are deaf, hard of hearing, or unable to speak

have traditionally depended on human interpreters or purpose-built assistive hardware, resources that are frequently expensive, scarce, or simply unavailable at the moment they are needed. Modern vision techniques have changed this picture: an ordinary laptop or phone camera is now enough to recognize a meaningful set of gestures accurately, with no gloves, markers, or depth sensors required.

The pipeline built for this project takes a continuous video stream, isolates the user's hand within each frame, derives a compact feature representation, and feeds that representation to a trained network that returns the most likely gesture label, all rendered back to the user with minimal delay. Ahead of classification, every frame passes through a short preprocessing chain — cropping to the hand region, resizing, background flattening, and aspect-ratio correction — so that the network always receives inputs in a consistent shape and scale. The vocabulary the system is designed to eventually support spans Hello, Thank You, Yes, No, Okay, Peace, Thumbs Up, Thumbs Down, Stop, Victory, I Love You, Rock On, and Pointing, opening the door to uses in sign-language translation, home automation, virtual-reality interfaces, gaming, robotics, healthcare, and classroom tools.

A. Problem Statement

Many existing gesture-recognition solutions lean on extra hardware — instrumented gloves, depth-sensing cameras, or dedicated motion trackers — which drives up cost and limits how portable a system can be. Purely classical imageprocessing pipelines also tend to falter once the background gets busy, the



hand rotates away from a canonical pose, or the user moves quickly, all of which hurt recognition reliability.

This project instead relies entirely on software running against ordinary webcam input. A single camera feed is captured, computer-vision routines locate and crop out the hand region, and a TensorFlow-based classifier interprets the isolated gesture before the outcome is shown through a lightweight onscreen interface.

B. Objectives

- Achieve reliable, real-time recognition of hand gestures directly from live webcam footage.
- Automatically isolate the hand region from the rest of the scene ahead of classification.
- Improve the quality of each frame through resizing, color normalization, and background suppression before it reaches the model.
- Use a TensorFlow-trained deep-learning model to determine the meaning of each captured gesture.
- Deliver a touch-free interaction style that feels natural for everyday human-computer communication.

II. SYSTEM ARCHITECTURE AND TECHNOLOGIES

The pipeline moves through six stages: capturing input, preprocessing each frame, detecting the hand, extracting landmark features, classifying the gesture, and finally displaying the outcome. OpenCV handles frame capture, and hand localization together with extraction of the twenty-one MediaPipe landmark points is delegated to CVZone's handtracking module. The cropped hand region is resized and normalized before a TensorFlow model assigns it a gesture class; the predicted label, its associated confidence value, and a bounding box around the hand are all overlaid on the live video. Because the stages are loosely coupled, new gestures, speechoutput support, additional languages, or integration with external devices can all be added without reworking the rest of

the pipeline.

Tech.	Category	Role
Python	Prog. Language	Ties every module together; implements gesture logic.
OpenCV	CV Library	Video capture, preprocessing, result rendering.
CVZone	Hand Tracking	High-level hand detection built on MediaPipe.
MediaPipe	Landmark Det.	Supplies 21 hand landmark points.
TensorFlow	Deep Learning	Loads trained classifier; predicts gesture.
NumPy	Numerical Comp.	Array operations, image-processing support.

TABLE I. TECHNOLOGY STACK

A. Model Architecture

Gesture classification rests on a MobileNetV2 backbone pretrained on ImageNet. During training, the convolutional layers of MobileNetV2 remain frozen, so learning is confined to the newly attached layers on top. Incoming images are resized to match MobileNetV2's expected input dimensions before passing through this frozen feature extractor.

A 128-unit dense layer follows the feature-extraction stage, paired with ReLU activation and dropout to curb overfitting and keep the model from becoming overconfident in its predictions. Its output feeds into a final classification layer that produces the predicted gesture.

MobileNetV2 was chosen over training a network from the ground up because transfer learning trains faster and generalises better when training data is limited, which matched the constraints of the gesture dataset used here. This makes transfer learning on top of MobileNetV2 a practical, efficient choice for the classification task at hand.

III. RESULTS AND DISCUSSION

The complete pipeline was implemented and evaluated in Python using OpenCV, CVZone, MediaPipe, TensorFlow, and NumPy. At runtime, the application pulls a frame from the



webcam, locates the hand through CVZone's tracking module, and normalizes the hand region to a fixed size and shape ahead of classification.

Once trained, the system consistently recognizes the gestures included in its training set. On a successful detection, the predicted label appears on screen next to the live camera feed, with the full capture-to-prediction cycle completing quickly enough that the user sees the result essentially instantly.

Testing covered the following scenarios:

- Capturing frames from the webcam.
- Locating the hand within each captured frame.
- Resizing and normalizing the isolated hand region.
- Loading the trained classification model.
- Correctly identifying the gesture being shown.

The pipeline was further checked end-to-end to confirm that each stage runs in the correct order, including edge cases such as no hand being present in frame or lighting conditions shifting mid-session. All of these checks passed.

Level	Scope	Result
Unit	Webcam capture, hand detection, preprocessing, model loading, prediction	Pass
Integration	Webcam + OpenCV + CVZone + preprocessing + TF model	Pass
System	Full pipeline under varied lighting and sustained use	Pass

TABLE II. TESTING SUMMARY

Accuracy is highest when the hand is fully visible against favorable lighting, and it dips somewhat when the background is cluttered, lighting is poor, or the hand is partially blocked from view. Even in these harder conditions, the target gestures were still recognized reliably. Taken together, these results suggest that pairing MediaPipe's landmark detection with a MobileNetV2-based classifier is a solid approach to real-time gesture recognition that needs no specialized hardware, and that

the resulting system holds up reasonably well even when conditions are not ideal.

IV. CONCLUSIONS

This work shows that combining computer-vision preprocessing with a deep-learning classifier can deliver reliable, real-time hand-gesture recognition using nothing more than an ordinary webcam. Removing the dependency on sensor gloves or dedicated motion-tracking equipment makes the approach both affordable and practical for gesture-driven communication. The completed pipeline captures video, locates the hand, extracts landmark features, prepares the image, and classifies the gesture through a trained TensorFlow model, returning results instantly while sustaining real-time performance. Potential applications span sign-language interpretation, healthcare, education, robotics, gaming, and home automation, with particular relevance as an accessibility aid for people with hearing or speech impairments.

A. Future Scope

- Growing the recognized gesture set to cover full words and complete sign-language alphabets.
- Adding support for dynamic, continuously moving gestures alongside the current static poses.
- Enabling detection of two hands simultaneously and linking the system to text-to-speech and speech-to-sign components.
- Porting the system to mobile platforms (Android and iOS) and to cloud deployment for broader reach.
- Connecting the system to smart-home devices so gestures can drive real-world control actions.
- Extending coverage to regional and additional sign languages.

REFERENCES

- [1] G. Bradski and A. Kaehler, Learning OpenCV: Computer Vision with the OpenCV Library. Sebastopol, CA, USA: O'Reilly Media, 2008. [2] I.



Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016. [3] F. Chollet, Deep Learning with Python, 2nd ed. Shelter Island, NY, USA: Manning Publications, 2021.

[4] OpenCV Documentation, "Open Source Computer Vision Library," 2025. [Online]. Available: <https://opencv.org/>

[5] TensorFlow Documentation, "TensorFlow Machine Learning Framework," 2025. [Online]. Available: <https://www.tensorflow.org/>

[6] MediaPipe Documentation, "Google MediaPipe Hands Solution," 2025. [Online]. Available: <https://developers.google.com/mediapipe>

[7] CVZone Documentation, "CVZone Computer Vision Library," 2025. [Online]. Available: <https://github.com/cvzone/cvzone> [8] A. Rosebrock, Deep Learning for Computer Vision with Python. PyImageSearch, 2019.