# A Real-Time Hand Gesture Based Fruit Learning Platform to Support Deaf Communication Using MediaPipe, SVM, and Django

Ms. Pawar Poornima Gangadhar[1], Mr. Naik Smit Sujit[2], Mr. More Parth Manoj[3],
Mr. Patil Soham Hemant[4], Mr. Narvekar Ronit Mahesh[5]
[1,2,3,4,5] *FAMT, Ratnagiri*

***Abstract*—Deaf people often find it hard to communicate with hearing individuals because they don't share the same way of expressing themselves. To help with this, this project suggests a web-based learning tool that teaches American Sign Language (ASL) signs for different fruits. It uses instructional videos to show how to make the signs, and then tests the learner's ability by using a live camera. The homepage of the platform introduces ten fruit signs through tutorials, and then lets users try making them. A module based on Google MediaPipe tracks hand movements and identifies 21 key points on each hand in real time. These points are turned into a detailed 117-dimensional feature vector, which is then classified by a Support Vector Machine with an RBF kernel. When the system recognizes a sign with enough confidence, it displays the fruit's name on the screen, shows a picture of the fruit on top of the video, and also speaks the name aloud using a text-to-speech feature. The training data was collected by having one team member perform each sign, then breaking the videos into grayscale frames and applying a six-fold data augmentation method. The final classifier shows good and reliable performance across all ten fruit signs under normal conditions, making it a useful tool for teaching sign language to deaf individuals.**

***Index Terms*—ASL Gesture Recognition, MediaPipe Hand Landmarks, Support Vector Machine, Django Web Framework, Deaf Learning Platform, Text-to-Speech, Real-Time Computer Vision.**

## I. INTRODUCTION

People talk and write a lot in everyday life, but these methods are hard for those with serious hearing loss to use. Sign language helps the deaf community communicate, but not many hearing people know even the basics of signing. This gap in understanding creates a big problem when it comes to learning, working, and connecting with others, affecting millions of people each year.Using automatic hand gesture recognition could help break this problem.A good system can watch a person's hands through a regular webcam, figure out the sign, and turn it into text or speech right away. That way, people don't need a trained interpreter for everyday conversations.But most existing systems just identify the sign, they don't help someone learn it.This project's system is different. Before the real-time detection starts, the app shows short videos and pictures explaining each of the ten ASL fruit signs in its set. Only after the user has seen these materials do they move to the live practice part, where the camera checks their signing in real time.The system uses Google MediaPipe to track body movements, creates a special 117-dimensional feature set, and uses an SVM classifier with an RBF kernel.
It connects everything through a Django web server, making it work in a web browser. When a sign is recognized correctly, the user gets three types of feedback at once: a picture of the fruit, a label on the screen, and an audio message. This helps build both visual and hearing memory.

Specific contributions of this work include:
• A dataset of hand-recorded, self-labeled American Sign Language (ASL) gestures for fruits, created from segmented video clips.
• A tutorial section on the homepage that includes instructional videos and gesture guides for each fruit.
• A 117-dimensional feature descriptor that captures joint angles, extension ratios, palm shape, and depth information from 21 MediaPipe landmark points.

• A six-fold data augmentation method that helps overcome the small size of the dataset while keeping the recognition accuracy high.
• A system that provides three types of output at the same time—visual overlay, text label, and spoken announcement—without stopping the camera feed.

## II. RELATED WORK

### 2.1 Deep Learning Approaches

Convolutional neural networks have been widely used with sign language datasets and can achieve very high top-one accuracy.However, one major drawback is the need for a lot of resources. Training a dependable CNN usually requires tens of thousands of labeled images and often a specialized graphics processor. Because of this, it's not practical to use such a model in a lightweight browser-based educational app without having access to server-side GPU support.

### 2.2 Skeleton-Based Methods with MediaPipe

Instead of working with raw pixel data, extracting hand skeletons helps avoid issues related to large data volumes.Once a dependable skeleton is obtained, even a simple traditional classifier can accurately recognize gestures. MediaPipe Hands from Google provides a 21-point skeleton at full video frame rate on standard consumer hardware without needing a GPU. Researchers have shown that models like SVM and gradient-boosted trees, trained on normalized MediaPipe keypoints, can perform as well as or better than CNNs when the number of gesture classes is limited to a few dozen.

### 2.3 Educational and Assistive Platforms

Most published work on assistive sign language tools has focused mainly on the recognition engine, with the learning process handled by separate apps or printed resources.Studies in education show that when learners get structured instruction first and then practice interactively, they remember the skills better than just practicing on their own. This system addresses that by combining the teaching material and the practice part all within the same web application.

### 2.4 Multimodal Feedback in Learning Systems

Research on multimedia learning shows that when a visual image is paired with an audio cue at the same time, it helps the brain remember information better than using just one of them.Applying this idea to a tool that teaches through gestures means that when someone makes the right movement, showing a picture, writing the word, and saying the word aloud all at once should help learners pick up new vocabulary faster than just seeing the word written down.

## III. SYSTEM DESIGN & ARCHITECTURE

The platform has two main parts: a Learning Module that appears when you open the app, and a Gesture Recognition Module that starts on a different screen.

### 3.1 Learning Module

When you first open the app, you see a grid of fruit cards on the home page.
Each card shows the fruit's name, a picture of the fruit, a correct ASL hand shape image, and a short video that shows the gesture from different angles. This part lets you learn at your own pace and get used to the gestures before moving on to the live camera screen. The design keeps the learning and testing parts separate so you can focus on understanding the gestures without feeling stressed or rushed.

### 3.2 Dataset Construction

Since there wasn't a public dataset available for the specific set of ten fruits used in this project, the team created their own.
One team member repeated each hand gesture multiple times in front of a regular webcam, and the whole session was recorded as video. The videos were then split into individual frames and saved as grayscale images. After removing blurry or dark frames, each of the ten fruit categories — Apple, Grapes, Mango, Pineapple, Pear, Cherry, Coconut, Avocado, Peach, and Orange — had between thirty and forty good quality images. This created a raw dataset of about three hundred to four hundred images in total.

### 3.3 Hand Landmark Detection

Each video frame is sent to MediaPipe Hands, which identifies the hand area and gives the normalised (x, y, z) coordinates of 21 key points on the hand — four per finger plus the wrist.
When extracting features from the dataset, the module runs in static-image mode, where accuracy is more important than speed. During real-time prediction, it

switches to continuous video mode, and a tracking-confidence gate helps avoid incorrect detections between frames. If two hands are visible at once, feature vectors are created for each hand separately, and their probability results are averaged before making a final prediction.

3.4 Feature Vector Construction

The raw coordinates of the hand landmarks include information about where the hand is located and its size, which doesn't tell us about the specific gesture being made.

During the feature extraction process, we remove this unnecessary information and create a 117-dimensional feature vector. This vector is made up of the following parts:

- The x and y positions of all 21 landmarks, measured from the wrist and adjusted so the size of the hand doesn't affect the results (42 values in total).
- The depth of each landmark, adjusted by subtracting the depth of the wrist and then divided by the standard deviation of the depth values (21 values).
- The straight-line distance from the wrist to each other keypoint after the hand has been normalized (21 values).
- The signed angle of each of the 20 finger segments, calculated using the arctangent function (21 values, including one for the palm).
- The ratio of the distance from the fingertip to the knuckle compared to the distance from the wrist, which shows how curled each finger is (5 values).
- The angle between consecutive fingertip vectors, which shows how spread out the fingers are (4 values).
- The distance from each fingertip to the center of the palm, along with the overall width of the palm and the ratio of width to height (4 values).

3.5 Augmentation Strategy

With only thirty to forty images per class, there was a real risk of overfitting.

To make the training data bigger and more varied without needing more recordings, each real feature vector was copied five times with some controlled random changes. Small Gaussian noise was added to the coordinate features, multiplicative noise was used for distance features, uniform random shifts were added to angle features, and additive noise was applied to extension ratios. This made the final dataset six times larger than the original while keeping the basic structure of the gestures the same.

3.6 Classifier

A Support Vector Machine with an RBF kernel was chosen because it works well with high-dimensional data and can still perform well even when there aren't too many training samples.

The model is inside a scikit-learn Pipeline that first scales the data to have zero mean and unit variance before classification. The regularisation parameter C was set to 50 to create a decision boundary that is tight but not too strict. Gamma was set to 'scale' so the kernel width automatically adjusts based on the spread of the features. Balanced class weighting was also used to make sure no single class dominates the training process. Probability estimates are enabled so that the model can provide a confidence score for each class during predictions.

3.7 Prediction Smoothing

The raw predictions for each frame can be unclear when the hand gesture changes.

To make the label more stable and easy to read, the system keeps the last fifteen frames in a circular buffer. The gesture is then identified by the class that appears most often in those frames. There is also a confidence threshold of 0.40. If the top class doesn't meet this threshold, the system shows a neutral 'Searching' message instead of possibly wrong fruit names.

3.8 Trimodal Output

• Image overlay: A picture of the identified fruit appears in the top-right corner of the video frame, gradually becoming visible as the hand moves closer and then fading away when the hand is lowered.

This effect is created using a method called weighted alpha blending.

• Text label: The name of the fruit is displayed in large, high-contrast letters at the bottom-left of the frame, ensuring it is easy to read even on smaller screens.

• Spoken announcement: As soon as the image overlay becomes fully visible, a text-to-speech engine says the name of the fruit.

This process runs in a separate thread so it doesn't interfere with the camera processing.

3.9 Web Application

The entire system is built as a Django project.

The main page provides the tutorial information from Section 3.1. Another page starts the device's camera and shows the video feed with annotations in the web browser. No special software needs to be installed on the user's computer—just a modern web browser—making the system work on any desktop or laptop, regardless of the operating system.

## IV. COMPARISON WITH EXISTING APPROACHES

| Approach | Method | Output & Learning | Platform |
|---|---|---|---|
| CNN Classifier | Deep CNN | Text only, no tutorial | Desktop / GPU |
| MP + Random Forest | Skeleton + RF | Text only, no tutorial | Local script |
| MP + KNN | Skeleton + KNN | Text only, no tutorial | Local script |
| Proposed System | Skeleton + SVM | Text + Image + Voice + Tutorial | Django Web App |

## V. RESULTS & DISCUSSIONS

After augmentation the training set contained roughly 1,800 to 2,400 samples. The SVM achieved a high training accuracy on this set, indicating that the RBF kernel successfully separated the ten gesture classes in the 117-dimensional descriptor space.

Live-camera trials across varied room lighting and different hand sizes confirmed that all ten fruit gestures trigger correct, stable recognition. The fifteen-frame smoothing window eliminated the rapid label flipping that occurred with single-frame inference, and the 0.40 confidence gate prevented tentative or partially formed gestures from triggering premature feedback.

The text-to-speech thread ran without measurable impact on frame processing time, confirming that the background-thread design achieved its intended goal. Alpha blending produced a visually smooth fade-in and fade-out effect for the fruit photograph, and the label reset mechanism ensured that the same fruit name was announced again whenever the user lowered and re-raised their hand.

Informal trials with users who had no prior ASL knowledge showed that watching the home-page video tutorials before attempting live practice noticeably reduced the number of attempts required before achieving a first successful recognition, validating the value of the integrated learning module.

| Fruit | Gesture Shape | Avg. Confidence | Detection |
|---|---|---|---|
| Apple | Curved index + fist | >85% | Stable |
| Grapes | Clustered fingertips | >80% | Stable |
| Mango | Cupped open palm | >82% | Stable |
| Pineapple | Spiked spread hand | >78% | Stable |
| Pear | Oval palm cup | >80% | Stable |
| Cherry | Two finger pinch | >83% | Stable |
| Coconut | Rounded two hands | >79% | Stable |
| Avocado | Curved flat palm | >81% | Stable |
| Peach | Soft loose cup | >80% | Stable |
| Orange | Closed round fist | >84% | Stable |

Following images depict some of the demonstrated fruit gestures used to test the actual working of the system


Fig 5.1: ASL Gesture for Mango


Fig 5.2: ASL Gesture for Apple


Fig 5.3: ASL Gesture for Pineapple

## VI. CONCLUSION

This project created a web-based platform that helps learners of deaf communication start from their first encounter with a set of ASL fruit gestures and progress all the way to practicing independently, all within one app. The home page tutorial fills in a gap that basic recognition systems don't cover, while the MediaPipe-plus-SVM system behind the scenes offers accurate, fast gesture classification without needing special equipment.

The app gives three types of feedback at once—visual overlay, on-screen text, and spoken confirmation—which uses multiple senses and is expected to help people remember gestures better than just one type of feedback.Early user observations support this idea, but a more detailed study with more people is planned as the next step.

Future work will expand the gestures to include the entire ASL fingerspelling alphabet and common objects, explore recognizing whole sentences using sequence models, and make the interface work well on mobile devices so learners can practice anywhere.

## VII. ACKNOWLEDGMENT

The project team records their gratitude to Prof. Poornima G. Pawar, whose patient guidance and timely feedback shaped every stage of this work. The team also thanks the faculty and management of the Department of CSE (AIML), Hope Foundation's Finolex Academy of Management & Technology, Ratnagiri, for the laboratory access and academic support that made this project possible.

## REFERENCES

[1] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C. L. Chang, M. G. Yong, J. Lee, W. T. Chang, W. Hua, M. Georg, and M. Grundmann, "MediaPipe: A framework for perceiving and processing reality," Workshop on Perception for Mobile Robots, 2019.

[2] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in Proc. 5th Annual ACM Workshop on Computational Learning Theory, Pittsburgh, PA, 1992, pp. 144–152.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort et al., "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011.

[4] N. Neverova, C. Wolf, G. W. Taylor, and F. Nebout, "Hand gesture recognition with jointly calibrated Leap Motion and video camera," Multimedia Tools and Applications, vol. 75, no. 22, pp. 14367–14392, 2016.

[5] O. Köpüklü, A. Gunduz, N. Kose, and G. Rigoll, "Real-time hand gesture detection and classification using convolutional neural networks," in Proc. IEEE Intl. Conf. on Automatic Face and Gesture Recognition, 2019.

[6] Django Software Foundation, Django Documentation, Version 4.2, https://docs.djangoproject.com, 2024.

[7] Google LLC, MediaPipe Hands Solution, https://developers.google.com/mediapipe/solutions/vision/hand_landmarker, 2024.

[8] R. E. Mayer, Multimedia Learning, 2nd ed. Cambridge University Press, 2009.

[9] World Health Organization, "Deafness and Hearing Loss," WHO Fact Sheet, March 2023. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss

[10] S. S. Rautaray and A. Agrawal, "Vision based hand gesture recognition for human-computer

interaction: A survey," Artificial Intelligence Review, vol. 43, no. 1, pp. 1–54, 2015.