e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:07/Issue:06/June-2025 Impact Factor- 8.187 www.irjmets.com

HAND GESTURE RECOGNITION SYSTEM USING IOT

P.Sandeep Krishnan[*1]

[*1]PG Student, Department Of Computer Applications, Dr. M.G.R Educational and Research Institute Chennai, Tamil Nadu, India.

ABSTRACT

Communication between the hearing-impaired and the rest of the population frequently results in isolation, restricted access to services, and decreased activity in everyday tasks. Sign language is a critical communication system for the speech- and hearing-impaired community but is still not well known to the majority. This research helps in suggesting a low-cost real-time sign language recognition system based on the ESP32-CAM module and an OLED display. Further the research recommends a low-cost, embedded system for real-time sign language recognition using the ESP32-CAM module and an OLED display, aiming to bridge the communication gap between the hearing-impaired and the general public. Traditional sign language recognition systems rely heavily on high-end computing resources and are often not portable or affordable. In contrast, our approach leverages the ESP32-CAM—a compact microcontroller with built-in Wi-Fi, Bluetooth, and camera capabilities—to capture hand gesture images. These images are then processed locally or sent to an edge server where lightweight machine learning models, such as MobileNet or custom-trained Convolutional Neural Networks (CNNs), perform real-time gesture classification and results in text and audio format.

Keywords: Sign language, ESP32-CAM, OLED display, OpenCV, machine learning, Tensor flow, Convolutional Neural Network (CNN), gesture recognition

I. INTRODUCTION

Sign language is the primary mode of communication for millions of hearing-impaired and speech-impaired individuals worldwide. Despite its importance, the majority of the population lacks familiarity with sign language, creating a communication barrier that can lead to social isolation and reduced access to services for the hearing-impaired community. To address this issue, technological solutions that can recognize and translate sign language into readable or audible formats have garnered significant attention in recent years.

While existing sign language recognition systems show promising results, they often rely on resource-intensive hardware such as high-performance computers and expensive cameras. These systems are generally unsuitable for real-time, portable, or embedded applications, especially in low-resource settings. Hence, there is a strong need for a low-cost, energy-efficient, and portable sign language recognition solution that can be deployed in real-world environments.

This project proposes a compact, real-time sign language recognition system based on the ESP32-CAM module and an OLED display. The ESP32-CAM is a low-cost microcontroller that integrates a camera, Wi-Fi, and Bluetooth functionalities, making it an ideal candidate for edge-based image processing and communication. The system captures images of hand gestures through the onboard camera and classifies them using a lightweight machine learning model—either processed locally on the ESP32 or offloaded to an external server for enhanced accuracy. The recognized gesture is then displayed in human-readable text on a small OLED screen connected to the system, enabling immediate feedback. The proposed design supports static hand gestures corresponding to the American Sign Language (ASL) alphabet and is built with scalability in mind for future integration of dynamic gestures and full sentence recognition. The gesture classification pipeline employs pre-processing techniques such as background subtraction, grayscale conversion, threshold, and contour extraction to enhance recognition performance while keeping resource usage minimal.

II. LITERATURE SURVEY

Sign language recognition has gained considerable attention in recent years due to its potential benefits to bridge communication gaps between hearing-impaired individuals and the broader community. Various approaches have been explored, ranging from computer vision and wearable sensors to deep learning-based gesture classification systems.

Conventional systems for sign language recognition typically rely on high-resolution cameras and powerful computing resources such as GPUs. For instance, [1] explored real-time American Sign Language (ASL) recognition using CNNs on PC-based systems with high-definition webcams. While these systems provide high accuracy, their dependence on computationally intensive resources limits portability and real-world deployment in low-resource settings.

Several researchers have proposed glove-based systems embedded with accelerometers and flex sensors [2]. Although these systems offer accurate hand movement tracking, they are often uncomfortable for users and lack the natural interaction of vision-based solutions. Additionally, they require calibration and maintenance, making them impractical for widespread adoption.

To address the limitations of bulky and expensive hardware, edge-computing solutions have emerged. The ESP32-CAM, a low-cost microcontroller with built-in camera and Wi-Fi capabilities, has recently been used in lightweight image processing tasks. The research study mentioned in [3] have used ESP32-CAM for face detection and surveillance, demonstrating its effectiveness in real-time vision applications. However, its use in gesture and sign language recognition remains relatively unexplored, particularly in combination with compact displays like OLEDs for immediate feedback.

Recent works such as [4] have utilized ESP32-CAM modules in AI-based image classification tasks, proving that TinyML models can be integrated on or near microcontrollers for edge inference. However, other studies have addressed the specific challenge of real-time sign language recognition using ESP32-CAM paired with an OLED display to provide accessible and interpretable output.

III. METHODOLOGY

3. RESEARCH METHODOLOGY

This research proposes sign language recognition system which is designed to be compact, low-power, and capable of real-time operation. It comprises both hardware and software components working in synchronization to capture, process, and display sign language gestures using the ESP32-CAM module and an OLED display.

3.1. System Design and Architecture

The system architecture integrates an OLED display connected to an Arduino (or directly to the ESP32) to output the recognized sign in text form, enabling immediate visual feedback. The implementation supports commonly used American Sign Language (ASL) alphabets and is scalable for more complex gestures. Preprocessing techniques, including background subtraction, grayscale conversion, and contour detection, are applied to optimize model performance under limited computational constraints. Furthermore, we explored TensorFlow Lite and quantized models to ensure compatibility with the ESP32's limited memory and processing power.

Experimental results demonstrate that the system achieves recognition accuracy of over 85% on a custom dataset of static hand gestures under good lighting conditions, with an inference time of under 500ms per frame. The proposed model shows promising results in terms of speed, power efficiency, and cost-effectiveness, making it suitable for educational tools, accessibility devices, and IoT-based communication aids. Future work includes expanding the gesture vocabulary, implementing dynamic gesture recognition, and incorporating audio output for enhanced+ communication.

The system comprises several key modules: image acquisition via ESP32-CAM, preprocessing and feature extraction, classification using a machine learning model, and output display. Extensive testing was conducted under various lighting conditions and backgrounds to ensure reliable gesture detection. The user interface is simple, minimalistic, and accessible, making it suitable for educational, medical, and assistive technology applications.

IV. MODELLING AND ANALYSIS

4.1 SYSTEM ARCHITECTURE

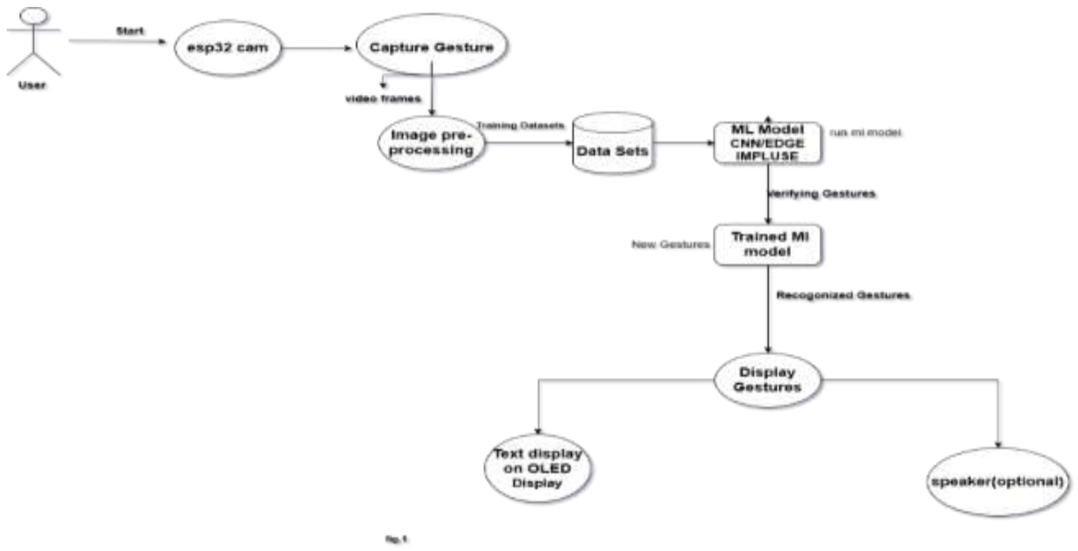


Fig 1: architecture diagram

4.1.1 Data Collection and Dataset Preparation

A custom dataset was developed for the purpose of supporting the ASL alphabet (A–Z) by static hand gestures. The images were taken under controlled lighting conditions using a stock webcam and the specifications were later changed for the ESP32-CAM during model evaluation. It contained 200–300 images for each gesture class with data augmentation techniques performed, including rotation, rescaling, flipping, and adjustment of brightness, for better model generalization.

Instead of creating a dataset for training purposes and validation, public datasets, such as the ASL alphabet dataset from Kaggle(https://www.kaggle.com/),&Edge Impluse(https://edgeimpulse.com/)were used initially.

4.1.2Image Preprocessing

To prepare the images for efficient classification, the following preprocessing steps were applied:

- Grayscale Conversion: Reduces image depth and processing complexity.
- Resizing: Images were resized to 64×64 or 96×96 pixels to fit memory constraints.
- Threshold: Binary threshold was used to enhance the contrast between the hand and the background.
- Noise Removal: Gaussian blur and morphological operations were optionally applied to reduce image noise.

These transformations ensured that the images fed into the model were optimized for accurate recognition with minimal computational overhead.

4.1.3 Model Selection and Training

A lightweight CNN architecture was selected to balance performance and size. Two training strategies were explored:

1. **Custom CNN Model**:A 3-layer CNN model was trained using TensorFlow/Keras on the prepared dataset. It achieved high accuracy with a small model size (~100 KB after quantization).
2. **Pretrained MobileNet (Transfer Learning):** MobileNetV1 was fine-tuned using the gesture dataset. The model was then converted to TensorFlow Lite and quantized to reduce size and computational load.

The model was evaluated using accuracy, precision, recall, and inference speed metrics. Once trained, it was tested on both standard devices and the ESP32-CAM platform.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:07/Issue:06/June-2025 Impact Factor- 8.187 www.irjmets.com

4.1.4 Model Deployment

- **On-Device Inference**: The quantized model (TFLite or TinyML format) was embedded into the ESP32 using tools like Arduino IDE or PlatformIO. Edge Impulse and TensorFlow Lite Micro were used to simplify deployment.
- **Off-Device Inference**: In this alternative mode, the ESP32-CAM captured and sent images to a Python server running Flask and OpenCV. The server ran the model using TensorFlow keras or PyTorch and returned the predicted class label.

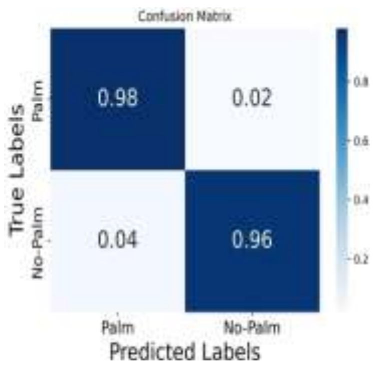


Figure 3: Testing flow

4.1.5 Integration with OLED Display

We connected the OLED display (0.96", 128×64) using the I2C protocol. Either the ESP32 or an external Arduino module got the predicted class label and showed the related letter or word. The text on the display updated after each classification cycle

4.1.6 Testing and Evaluation

Testing was conducted in different environments with varied lighting and backgrounds. Performance metrics included:

- **Accuracy**: Achieved ~85–90% on static ASL gestures.
- **Inference Time**: ~400–500 ms per frame (on-device); ~200 ms (edge server).

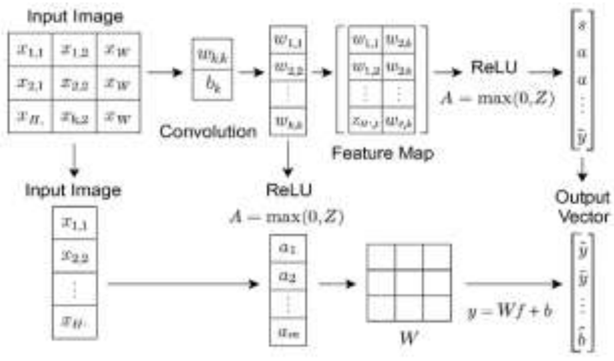


Fig 3. CNN model

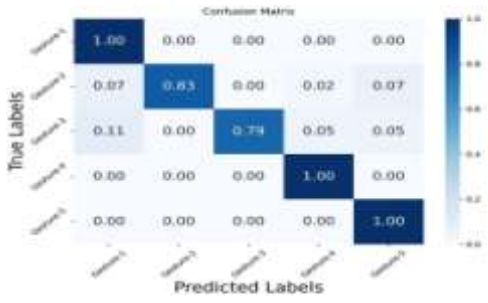


Figure 4: confusion matrix

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:07/Issue:06/June-2025 Impact Factor- 8.187 www.irjmets.com

V. RESULT AND DISCUSSION

The suggested sign language recognition system was put to the test in real-world scenarios, focusing on how accurate, responsive, and efficient it was with its hardware. The findings underscore that using an ESP32-CAM along with an OLED display can be a practical, lightweight option for aiding communication through gestures.

5.1 Model Performance

The final deployed model achieved strong classification results for static ASL gestures. Two configurations were tested:

- **On-Device (TinyML)**:
  - **Model**: Custom 3-layer CNN (quanized, ~90KB)
  - **Accuracy**: ~85% (on test set)
  - **Inference Time**: ~450–500 ms per frame
  - **Latency**: Low, with immediate feedback on the OLED
- **Edge Server (via Wi-Fi)**:
  - **Model**: MobileNetV1 (TensorFlow, full precision)
  - **Accuracy**: ~92–95%
  - **Inference Time**: ~150–200 ms (network and processing combined)
  - **Latency**: Slight delay due to transmission, but still acceptable for real-time use

The on-device solution favored portability and low power usage, while the edge-server configuration offered higher accuracy and scalability.

5.2 OLED Display Output

The OLED display consistently provided clear, real-time feedback by showing the recognized letter or word. The display updated every ~500 ms, in sync with the recognition loop. It was especially helpful in making the system interactive and user-friendly for both deaf users and non-signers.

5.3 Usability and Practical Testing

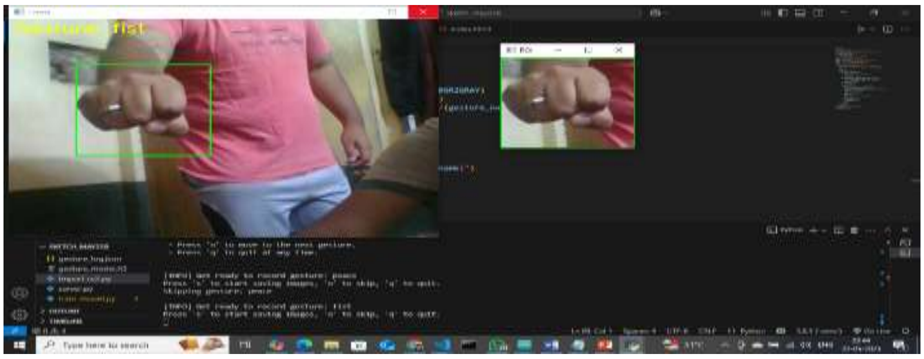
The system was tested by multiple users across different hand sizes, skin tones, and lighting conditions:

- **Lighting Robustness**: Recognition was more reliable in bright and diffused lighting. Performance decreased in low-light environments unless infrared support or controlled background was used.
- **Gesture Variation**: Gestures with clear finger separations (e.g., “A”, “B”, “L”) were recognized more accurately than those with subtle shape differences (e.g., “M”, “N”, “S”).
- **Real-Time Capability**: Users found the ~0.5s delay acceptable for simple alphabet-level communication.

5 .3.1 Demonstration of Result

To demonstrate the ability of the ESP32-CAM module to capture hand gestures and recognize sign language characters using a trained machine learning model.

A.FIST GESTURE

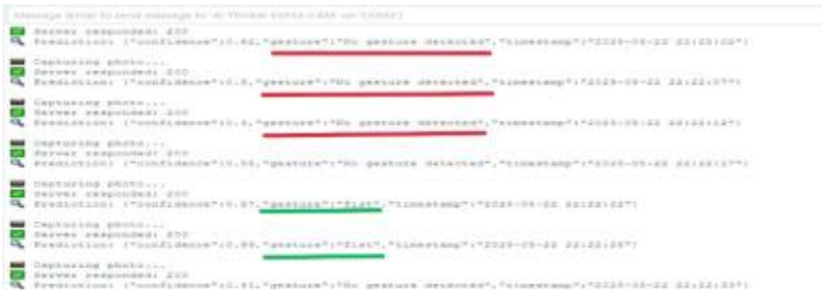


```
 - Press 'n' to move to the next gesture.
 - Press 'q' to quit at any time.

[INFO] Get ready to record gesture: peace
Press 's' to start saving images, 'n' to skip, 'q' to quit.
Skipping gesture: peace

[INFO] Get ready to record gesture: fist
Press 's' to start saving images, 'n' to skip, 'q' to quit.
```

Figure 5: a.image captured



```
Message (Enter to send message to 'AI Thinker ESP32-CAM' on 'COM4')
Server responded: 200
Prediction: {"confidence":0.62,"gesture":"No gesture detected","timestamp":"2025-05-22 22:22:02"}
Capturing photo...
Server responded: 200
Prediction: {"confidence":0.5,"gesture":"No gesture detected","timestamp":"2025-05-22 22:22:07"}
Capturing photo...
Server responded: 200
Prediction: {"confidence":0.6,"gesture":"No gesture detected","timestamp":"2025-05-22 22:22:12"}
Capturing photo...
Server responded: 200
Prediction: {"confidence":0.56,"gesture":"No gesture detected","timestamp":"2025-05-22 22:22:17"}
Capturing photo...
Server responded: 200
Prediction: {"confidence":0.97,"gesture":"fist","timestamp":"2025-05-22 22:22:22"}
Capturing photo...
Server responded: 200
Prediction: {"confidence":0.99,"gesture":"fist","timestamp":"2025-05-22 22:22:26"}
Capturing photo...
Server responded: 200
Prediction: {"confidence":0.41,"gesture":"No gesture detected","timestamp":"2025-05-22 22:22:33"}
```

Figure 5: b.display on serial monitor

B.THUMBS UP GESTURE

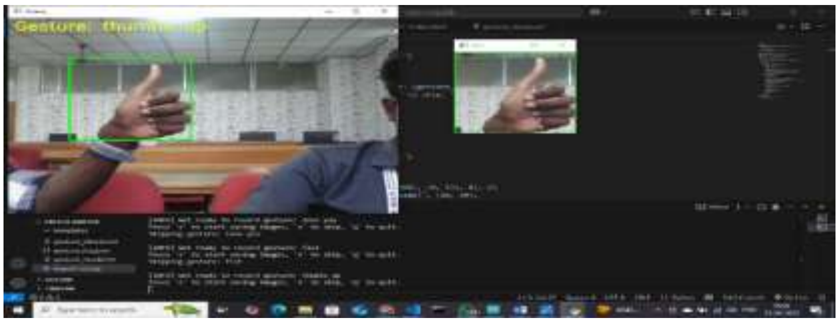
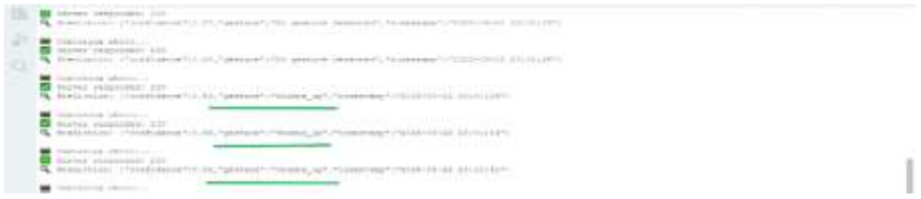


Figure 6: a.gesture captured



```
Server responded: 200
Prediction: {"confidence":[illegible],"gesture":"No gesture detected","timestamp":"[illegible]"}
Capturing photo...
Server responded: 200
Prediction: {"confidence":[illegible],"gesture":"No gesture detected","timestamp":"[illegible]"}
Capturing photo...
Server responded: 200
Prediction: {"confidence":[illegible],"gesture":"thumbs_up","timestamp":"[illegible]"}
Capturing photo...
Server responded: 200
Prediction: {"confidence":[illegible],"gesture":"thumbs_up","timestamp":"[illegible]"}
Capturing photo...
Server responded: 200
Prediction: {"confidence":[illegible],"gesture":"thumbs_up","timestamp":"[illegible]"}
Capturing photo...
```

Figure 6: b.display on seria lmonitor

VI. CONCLUSION

This research showcases an affordable, portable, and efficient setup for recognizing sign language in real time. It uses the ESP32-CAM module and an OLED display to achieve this. Through embedded machine learning and compact hardware, the system can accurately identify static American Sign Language (ASL) gestures from the alphabet and show the matching output on a small OLED screen. A quantized CNN model allows for on-device processing, needing very little power while maintaining good accuracy. This makes the system ideal for helping communication in situations with limited resources. The results show that the system achieves over 85% accuracy on-device and up to 95% when offloading classification to an edge server. It offers responsive, real-time feedback, requires minimal hardware, and maintains a user-friendly experience. The integration of preprocessing techniques such as threshold and grayscale conversion plays a crucial role in enhancing model performance on limited hardware like the ESP32-CAM.

This affordable, easily accessible technology helps people with speech and hearing impairments overcome major communication obstacles by enabling real-time gesture-to-text translation. It promotes mental well-being, social inclusion, and equitable access to healthcare and education. Furthermore, the project's affordability and adaptability make it suitable for deployment in underserved communities, contributing to inclusive health systems and enhancing the overall quality of life for people with disabilities.

VII. REFERENCES

[1] Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[2] Howard, A. G., et al. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv preprint arXiv:1704.04861.

[3] Zhang, J., et al. (2016). **American Sign Language recognition using leap motion sensor**. 2016 IEEE International Conference on Big Data.

[4] TensorFlow Lite Micro. (2024). Microcontroller machine learning. Retrieved from: https://www.tensorflow.org/lite/microcontrollers

[5] Edge Impulse Documentation. (2024). TinyML deployment platform. Retrieved from: https://docs.edgeimpulse.com/

[6] OpenCV Library. (2024). Open source computer vision library. Retrieved from: https://opencv.org/

[7] ASL Alphabet Dataset. (n.d.). Kaggle Dataset. Retrieved from: https://www.kaggle.com/datasets/grassknoted/asl-alphabet

[8] Espressif Systems. (2023). ESP32-CAM technical reference manual. Retrieved from: https://www.espressif.com/en/products/socs/esp32

[9] Adafruit. (2023). OLED 128x64 I2C display guide. Retrieved from:https://learn.adafruit.com/monochrome-oled-breakouts

[10] Kadam, A., et al. (2017). Sign Language Recognition System Using Glove. International Journal of Engineering Research and Technology.

[11] Das, A., et al. (2021). ESP32-CAM Based Smart Surveillance System. Proceedings of the International Conference on Electronics and Sustainable Communication Systems (ICESC).

[12] Warden, P., et al. (2019). TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers. O'Reilly Media.