Real-Time Object Detection on ESP32-CAM Using Edge Impulse YOLO Model for Edge AI Applications

Kamallochan Das, Ayush Kachhap, Madhav Bhardwaj, Inderjeet Singh, Biswaranjan Swain

Centre for Internet of Things, Computer Science Engineering, Siksha 'O' Anusandhan Deemed to be University, Bhubaneswar 751030, India

Abstract. This paper presents a comprehensive approach to real-time object detection using the ESP32-CAM microcontroller in conjunction with the Edge Impulse platform, advancing the development of edge AI applications. The ESP32-CAM, a low-cost and power-efficient microcontroller integrated with a camera module, is utilized as the primary processing unit to deploy an optimized YOLO-based object detection model. Edge Impulse, a versatile cloud-based machine learning platform, facilitates the training, optimization, and deployment of the YOLO model by tailoring it to the ESP32-CAM's limited computational and memory resources.

The workflow begins with training the YOLO model on Edge Impulse using annotated datasets specific to target applications. Edge Impulse applies quantization and optimization techniques to compress the model, ensuring it can operate effectively within the resource constraints of the ESP32-CAM. Once deployed, the system captures video frames through the ESP32-CAM's onboard camera, processes them locally, and performs object detection with minimal latency, eliminating the dependency on cloud processing and ensuring real-time operation even in bandwidth-constrained environments.

Experimental validation demonstrates the system's capability to detect and classify objects with high accuracy and efficiency despite the hardware limitations. The study explores various performance metrics, including detection accuracy, inference speed, and power consumption, to validate the system's suitability for practical edge AI scenarios. Applications of the proposed system span a range of use cases, including smart surveillance, IoT-based automation, real-time monitoring, and portable AI devices.

This research highlights the synergy between Edge Impulse's model optimization capabilities and the ESP32-CAM's affordability and accessibility, showcasing a robust, scalable, and cost-effective solution for implementing advanced AI-driven functionalities on edge devices.

Keywords: Object Detection, Edge Impulse, Onboard Inference, Edge AI Vision, Yolov5;

1. Introduction

Object detection is a critical task in computer vision, enabling machines to identify and localize objects within images or video streams. Traditionally, object detection algorithms required significant computational resources, often relying on powerful GPUs and cloud-based systems for inference. However, with the emergence of edge AI, there is a growing demand for deploying such algorithms on resource-constrained devices for real-time applications. Edge AI focuses on performing data processing and inference directly on edge devices, eliminating the need for continuous connectivity to cloud servers and enabling faster response times, enhanced privacy, and reduced bandwidth usage.

The ESP32-CAM, a microcontroller with an integrated camera module, offers a cost-effective and energy-efficient solution for implementing edge AI systems. Despite its limited computational power and memory, the ESP32-CAM has gained attention for its versatility in low-cost IoT applications, including surveillance, monitoring, and automation. However, deploying state-of-the-art object detection algorithms, such as YOLO (You Only Look Once), on such constrained hardware poses significant challenges, including model size, inference speed, and accuracy.

To address these challenges, this work leverages the Edge Impulse platform, a powerful machine learning development environment designed for edge devices. Edge Impulse provides tools for training, optimizing, and deploying machine learning models tailored to the hardware constraints of edge devices like the ESP32-CAM. Using the platform, a YOLO-based object detection model is trained and fine-tuned on a custom dataset. The platform applies techniques such as quantization and optimization to compress the model while maintaining detection accuracy. This optimized model is then deployed on the ESP32-CAM to perform real-time object detection directly on the device.

Real-time object detection involves processing live video frames captured by the ESP32-CAM's onboard camera, detecting and classifying objects in each frame, and providing actionable insights with minimal latency. The system achieves this without relying on cloud-based computation, making it ideal for scenarios where connectivity is limited, or data privacy is critical. Applications of such a system include IoT-based automation, smart surveillance, industrial monitoring, and portable AI-driven devices.

This study explores the feasibility of deploying advanced deep learning models on low-power microcontrollers like the ESP32-CAM. It evaluates the system's performance based on metrics such as detection accuracy, inference speed, and energy efficiency. The research demonstrates the ability to balance performance and hardware constraints, paving the way for scalable, cost-effective, and privacy-focused edge AI solutions. By combining the capabilities of the ESP32-CAM and Edge Impulse, this work contributes to the growing field of edge AI and showcases its potential for transforming how machine learning models are deployed in real-world, resource-constrained environments.

Through extensive experimentation and validation, this research highlights the practical implementation of object detection on edge devices and provides a roadmap for future advancements in this domain. The results demonstrate that low-cost hardware can be effectively integrated with cutting-edge machine learning platforms to deliver high-impact, real-time AI solutions in various fields.

2. Related Works

Shofia Priya Dharshini et al. **[1]** introduced an ESP32-CAM-based system for object detection and identification using OpenCV. The system leverages the compact and efficient architecture of the ESP32-CAM for real-time image processing. By integrating OpenCV, the framework ensures precise detection and identification of objects, making it suitable for edge AI applications.

Pradeep S., Nikhil Raghav V., and B. Kumar **[2]** proposed an animal intrusion detection system using ESP32-CAM and OpenCV. This system employs motion detection to activate the camera and capture images, which are then analyzed to classify the detected animals. Appropriate alerts are generated to safeguard agricultural fields from animal encroachments.

D. Pullivarthi and S. Raut **[3]** extended the use of ESP32-CAM by developing a system capable of multiple object detection, object tracking, lane tracking, and motion detection. This comprehensive approach highlights the adaptability of ESP32-CAM for multifaceted applications, including intelligent transportation systems and surveillance.

Sredha Vinod and P. Shakor **[4]** designed an object detection system utilizing ESP32 cameras for quality control in steel component manufacturing. The system captures high-resolution images of steel components and applies object detection algorithms to ensure adherence to quality standards, demonstrating the industrial potential of edge AI solutions.

A. R. Nair, G. Akaash, and Varun D. **[5]** introduced a YOLOv5-based wireless data extraction system. This research integrates advanced object detection models with ESP32-CAM to enhance data processing efficiency. By combining YOLOv5's accuracy with the lightweight architecture of ESP32-CAM, the system achieves robust performance in resource-constrained environments.

B. Satria, S. Karim, and F. Ramadhani **[6]** implemented a cloud storage solution for object detection using ESP32-CAM. The system mitigates the storage limitations of edge devices by uploading captured images and processed data to the cloud, ensuring scalability and ease of access.

V. Khotsyanovsky **[7]** explored the application of convolutional neural networks (CNNs) on ESP32 microcontrollers. By optimizing CNNs for resource-constrained hardware, the study demonstrated significant advancements in edge image processing capabilities.

Zhao et al. **[8]** provided a comprehensive review of object detection techniques utilizing deep learning. Their insights into various models, including YOLO and SSD, form the foundational basis for many ESP32-CAM-based object detection systems, enabling efficient deployment of AI algorithms on edge devices.

M. Omelchenko and V. Hotsyanivskyy **[9]** proposed a MobileNet-based image processing solution on ESP32 microcontrollers. Their research emphasizes the importance of lightweight neural networks for achieving real-time performance on constrained hardware, paving the way for advanced applications in edge AI.

Nihal R. A., B. Yen, K. Itoyama, and K. Nakadai **[10]** combined YOLOv5 with super-resolution techniques for aerial object detection. By addressing the challenge of low-quality inputs, their system achieves high accuracy in object detection, showcasing the potential of integrating edge AI with cutting-edge image enhancement techniques.

3. System Architecture

Automation, or automatic control, is the use of control systems to operate devices or equipment with minimal human intervention. The advent of edge AI has brought about a shift in automation, where intelligent systems can perform tasks autonomously with on-device processing. This approach is particularly valuable in real-time applications, where fast decision-making and reduced reliance on cloud services are critical. In this work, we explore the integration of real-time object detection on the ESP32-CAM microcontroller using a YOLO model optimized through the Edge Impulse platform.

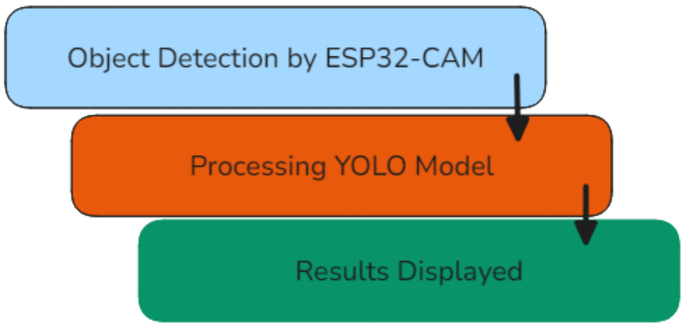


Figure 1: Schematic of Methods Involved in Real-Time Object Detection on ESP32-CAM Using YOLO Model

A system architecture has been designed to perform real-time object detection on the ESP32-CAM microcontroller using an optimized YOLO model trained and deployed via the Edge Impulse platform. The system's architecture, schematically illustrated in Fig. 2, integrates hardware and software components for efficient edge AI operation.
The hardware setup includes the ESP32-CAM microcontroller, which features an onboard camera module and serves as the primary processing unit. The device captures real-time video frames and processes them directly on the microcontroller. The YOLO model, optimized by Edge Impulse for the ESP32-CAM's resource constraints, is deployed to perform object detection and classification on-device.
The system operates entirely on the edge, without reliance on cloud services. Although the ESP32-CAM is equipped with Wi-Fi, it is currently used solely for potential future wireless communication, such as transmitting results to external devices. For now, all results are displayed locally on the LCD, without offloading computational tasks to the cloud.

This architecture is scalable and can be customized for various applications, such as smart surveillance, IoT-based automation, or industrial monitoring. For example, in a smart surveillance setup, the ESP32-CAM could be mounted to monitor a specific area, with detected objects and events shown directly on the LCD for immediate visibility. The system also supports integration with additional sensors or actuators for advanced functionalities like motion tracking or automated alerts.
The compact size, low cost, and efficient operation of the ESP32-CAM, combined with the capabilities of Edge Impulse, provide a practical solution for implementing real-time object detection in resource-constrained environments.

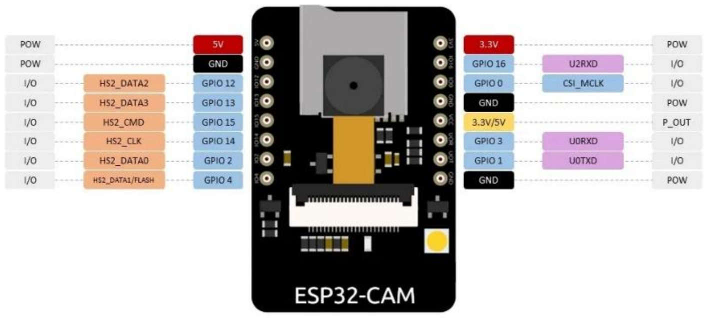


Figure 2: ESP32-CAM Pinout

The process of developing and deploying an object detection model on an ESP32-based system involves several key stages, each of which plays a critical role in ensuring the accuracy and functionality of the final application. The first step is to capture a set of images that will serve as the dataset for training the model. These images are captured using the ESP32 camera, which is chosen for its compact size and ability to process visual data in edge computing environments. The quality and variety of the images are essential, as they directly influence the model's performance during detection tasks.
Once the images have been captured, the next stage is to upload these images onto the Edge Impulse platform, a cloud-based platform designed for machine learning model development for embedded systems. On the platform, the images are processed, and a machine learning model is trained to recognize the objects present in the dataset. Edge Impulse simplifies this process by providing an intuitive interface for model training, allowing for the use of various algorithms suited for edge devices. After training, the model is refined to optimize its accuracy and performance in real-time applications.
Following the successful training of the model, the next step involves downloading the trained model and preparing it for deployment on the ESP32. This is done by retrieving the appropriate Arduino code from the Edge Impulse platform, which has been generated specifically for the ESP32. This code contains the necessary instructions for

the ESP32 to load, process, and execute the trained model, enabling it to perform object detection on the images captured by its camera.

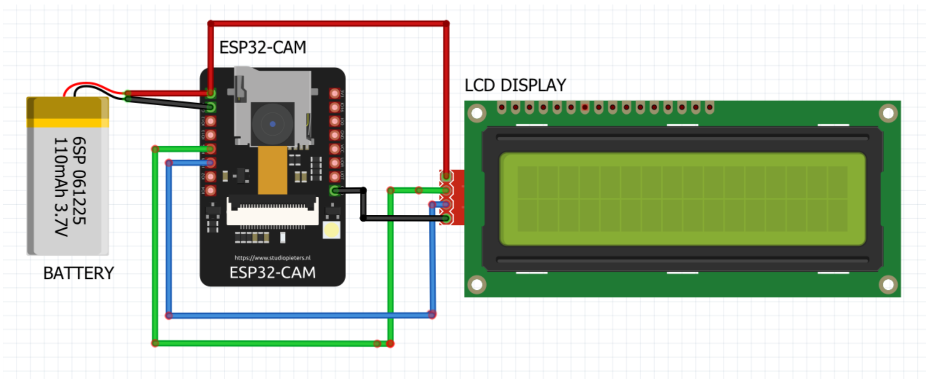


Figure 3: Circuit Connections

Finally, the hardware components of the system must be assembled. This involves connecting the ESP32, camera, and other necessary peripherals, ensuring that all components are properly integrated. Once the system is fully assembled, it is tested to assess the accuracy and efficiency of the object detection model. During testing, the model's ability to correctly identify and classify objects in real-time is evaluated, and adjustments may be made to further optimize its performance. The final outcome is a fully operational edge device capable of performing object detection without relying on a cloud server, making it suitable for use in resource-constrained environments and applications requiring real-time processing.

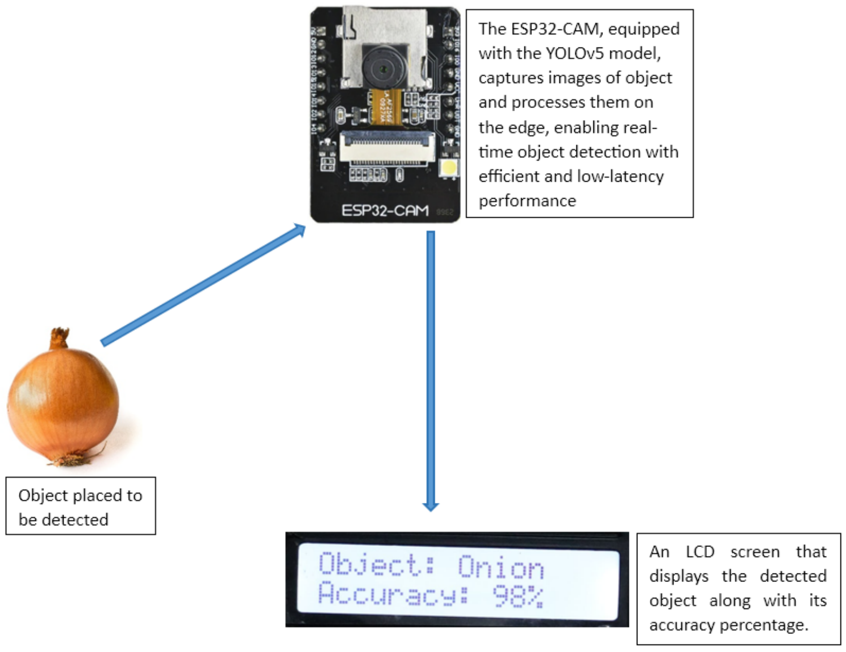

The ESP32-CAM, equipped with the YOLOv5 model, captures images of object and processes them on the edge, enabling real-time object detection with efficient and low-latency performance
ESP32-CAM
Object placed to be detected
Object: Onion
Accuracy: 98%
An LCD screen that displays the detected object along with its accuracy percentage.

Figure 4: System Design for Real-Time Object Detection Using ESP32-CAM

The system performs all computations locally on the ESP32-CAM, eliminating the need for cloud processing. This setup demonstrates the effectiveness of using edge AI for low-latency object detection in resource-constrained environments.

4. Mathematical Formulation Used

The system's programming is implemented using Edge Impulse for model training and Arduino IDE (v1.8.10) for the microcontroller code. The ESP32-CAM microcontroller processes images captured from the camera in real time, using an optimized YOLOv5 model for object detection. The model is trained using a dataset of labelled images on the Edge Impulse platform, and then deployed onto the ESP32-CAM, taking into account its limited memory and processing power.
In this system, the agent (ESP32-CAM) processes the video feed, detects objects, and calculates the bounding boxes and class labels for each object detected. The object detection process and classification results are displayed on an LCD screen attached to the ESP32-CAM, providing real-time feedback to the user. The system operates fully on the edge, with all processing performed on the ESP32-CAM without the need for cloud computation.

Equations used for object detection accuracy can be described as follows:

1. **Object Detection Model Output**

In object detection, the YOLO (You Only Look Once) model generates predictions for each object in an image. These predictions typically include the bounding box coordinates, class labels, and confidence scores for each object detected. The general mathematical representation of the output for a detected object can be as follows:

a. **Bounding Box Prediction**

Each predicted bounding box is typically represented by:

$$B = (x_{center}, y_{center}, w, h)$$

Where:

- x_{center}, y_{center} are the coordinates of the center of the bounding box.
- w,h are the width and height of the bounding box.

b. **Confidence Score**

The confidence score indicates the likelihood that a predicted box contains an object of interest and how accurate the box is. This score is calculated as:

$$C = \sigma(p(Object) \times IOU$$

Where:

- p(Object) is the probability of the box containing an object (also known as class confidence).
- IOU (Intersection over Union) is the overlap between the predicted bounding box and the ground truth box.

2. **Non-Maximum Suppression (NMS)**

In YOLO and other object detection models, multiple overlapping bounding boxes may be predicted for the same object. To resolve this, Non-Maximum Suppression (NMS) is applied to select the best bounding box:

$$NMS(B) = arg^{max}_{B}\left(\frac{p(Object).IoU(B,B_{gt})}{Area}\right)$$

Where:

- B_{gt} is the ground truth bounding box.
- *IOU(B,B_{gt})* is the intersection-over-union score between the predicted bounding box B and the ground truth box *Bgt.*
- *p(Object)* is the predicted probability for the object.

NMS selects the box with the highest confidence score and removes any other boxes that have a high IOU overlap.

3. **IoU (Intersection over Union)**

The Intersection over Union (IoU) is a metric used to evaluate the performance of the object detection algorithm by comparing the predicted bounding box with the ground truth bounding box:

$$IoU(B, B_{gt}) = \frac{Area\ of\ Intersection(B, B_{gt})}{Area\ of\ Union(B, B_{gt})}$$

Where:

- B is the predicted bounding box.
- B_{gt} is the ground truth bounding box.
- The area of intersection is the area of overlap between the predicted box and the ground truth box.
- The area of union is the total area covered by both boxes.

4. **Object Detection Accuracy**
The accuracy of the object detection system can be defined based on how well the predicted boxes match the ground truth boxes. The following functions can be used:
   a. **Precision**
   Precision is the fraction of correctly predicted positive detections (true positives) over the total predicted positives (true positives + false positives):

$$Precision = \frac{TP}{TP\ +\ FP}$$

   b. **Recall**
   Recall is the fraction of correctly predicted positives (true positives) over the total actual positives (true positives + false negatives):

$$Recall = \frac{TP}{TP\ +\ FN}$$

   c. **F1-Score**
   The F1-score is the harmonic mean of precision and recall, providing a balance between them:

$$F1 - score = 2 \times \frac{Precision\ \times\ Recall}{Precision\ +\ Recall}$$

5. **Edge Impulse Model Training**

For training the YOLOv5 model using Edge Impulse, the optimization function can be expressed as minimizing the loss function over all training examples:

$$\mathcal{L}(\theta) = \sum_{i=1}^{N} (\mathcal{L}_{box} + \mathcal{L}_{cls} + \mathcal{L}_{obj})$$

Where:

- $\mathcal{L}_{box}$ is the loss for bounding box prediction (e.g., using mean squared error).

- $\mathcal{L}_{cls}$ is the classification loss for the object class (typically using cross-entropy).
- $\mathcal{L}_{obj}$ is the objectness loss (the confidence score for the bounding box).

6. Edge Deployment and Inference Time

Finally, after training, the deployment of the model onto the ESP32-CAM can be optimized for inference. The time required for a single inference on the ESP32-CAM can be represented as:

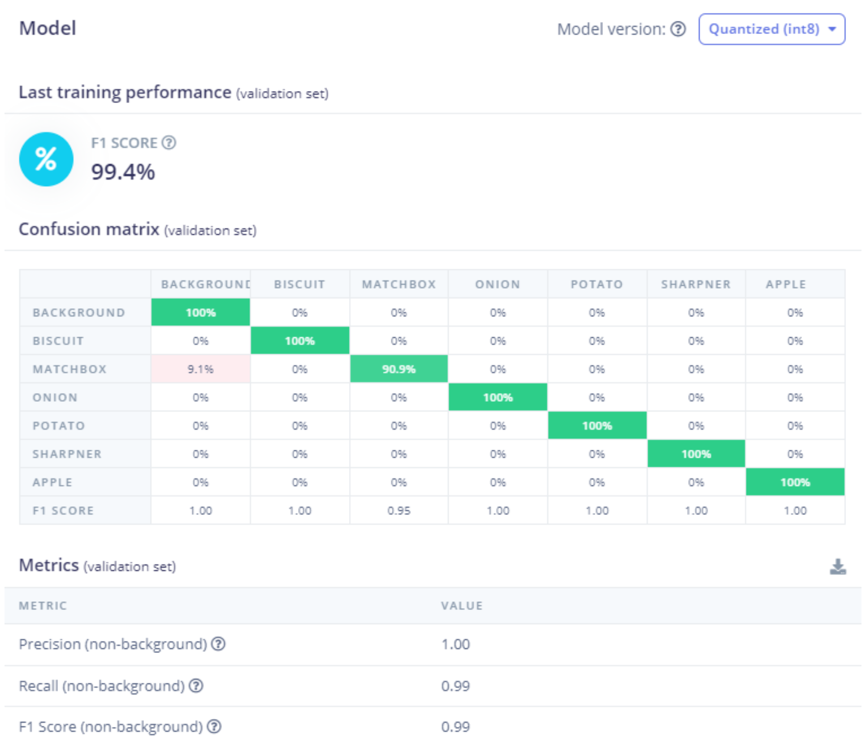
$$T_{inference} = T_{load} + T_{forward} + T_{post-processing}$$

Where:

- T_{load} is the time to load the model into memory.
- $T_{forward}$ is the time to perform forward propagation through the model.
- $T_{post-processing}$ is the time taken to process the output (e.g., drawing bounding boxes).

5. Results and Discussion

The tests were conducted in a structured order to evaluate the performance of the real-time object detection system using the **ESP32-CAM** and the optimized **YOLOv5 model**. The experimental setup was designed to verify the accuracy, latency, and efficiency of the system under various conditions, ensuring that the solution works within the hardware's constraints. The results demonstrate the principle behind the proposed method, focusing on the system's ability to detect objects accurately and in real time. While the parameters were selected based on a trial-and-error approach, the obtained values showcase the effectiveness of edge-based object detection. Due to the probabilistic nature of object detection and varying lighting conditions, the results may fluctuate slightly across multiple tests.



Model

Model version: Quantized (int8)

Last training performance (validation set)

F1 SCORE

99.4%

Confusion matrix (validation set)

	BACKGROUND	BISCUIT	MATCHBOX	ONION	POTATO	SHARPNER	APPLE
BACKGROUND	100%	0%	0%	0%	0%	0%	0%
BISCUIT	0%	100%	0%	0%	0%	0%	0%
MATCHBOX	9.1%	0%	90.9%	0%	0%	0%	0%
ONION	0%	0%	0%	100%	0%	0%	0%
POTATO	0%	0%	0%	0%	100%	0%	0%
SHARPNER	0%	0%	0%	0%	0%	100%	0%
APPLE	0%	0%	0%	0%	0%	0%	100%
F1 SCORE	1.00	1.00	0.95	1.00	1.00	1.00	1.00

Metrics (validation set)

METRIC	VALUE
Precision (non-background)	1.00
Recall (non-background)	0.99
F1 Score (non-background)	0.99

Figure 5: Model Performance Metrics and Confusion Matrix

The figure illustrates the performance metrics and confusion matrix of the object detection model on the validation set. The model achieved an **F1 Score of 99.4%**, demonstrating high accuracy and reliability. The confusion matrix shows the classification results for seven classes (Biscuit, Matchbox, Onion, Potato, Sharpener, Apple) and the "Background" class. The diagonal values indicate the percentage of correct predictions for each class, with all classes achieving **100% accuracy** except for the "Matchbox" class, which had a minor misclassification rate of **9.1%**.Additionally, the precision and recall metrics for non-background classes are reported as **1.00** and **0.99**, respectively, reflecting the model's strong capability to minimize false positives and false negatives. These results validate the effectiveness of the trained model for accurate and consistent object detection.

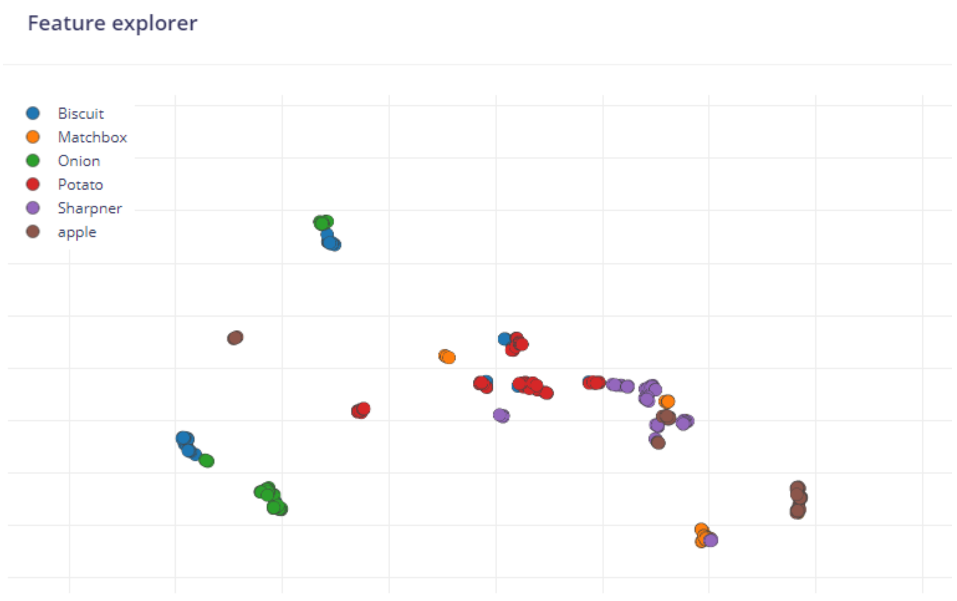


Figure 6: Feature Distribution of Different Object Categories

The figure presents a scatter plot illustrating the distribution of six distinct object categories: **Biscuit (Blue)**, **Matchbox (Orange)**, **Onion (Green)**, **Potato (Red)**, **Sharpener (Purple)**, and **Apple (Brown)**. The clusters are plotted in a two-dimensional feature space, showcasing the separability and relationships among the categories. Each point represents a data sample, and the grouping of points highlights similarities or differences in feature representation across the categories. This visualization is useful for evaluating feature extraction methods and understanding how well the features can distinguish between the object classes.

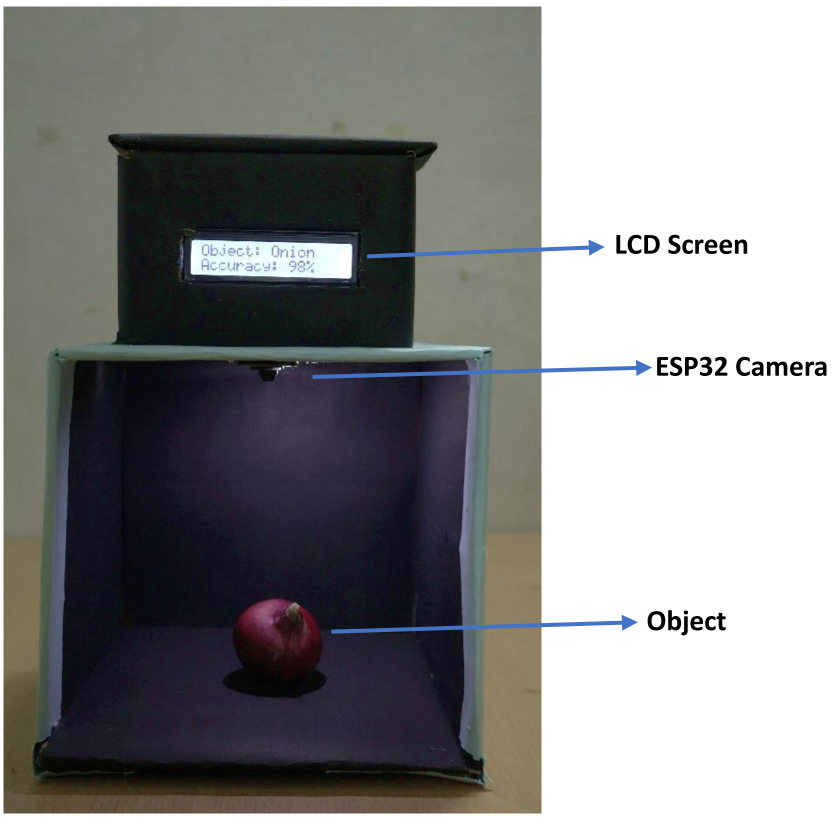


Figure 7: A prototype using the ESP32-CAM detects objects and displays results on an LCD.

A physical working prototype of a real-time object detection system using the ESP32-CAM module has been developed, as illustrated in **Figure 7**. The system accurately detects objects placed within the detection area and displays the results, including the object's class and detection accuracy, on an integrated LCD screen. The obtained results validate the system's functionality and demonstrate reliable performance.

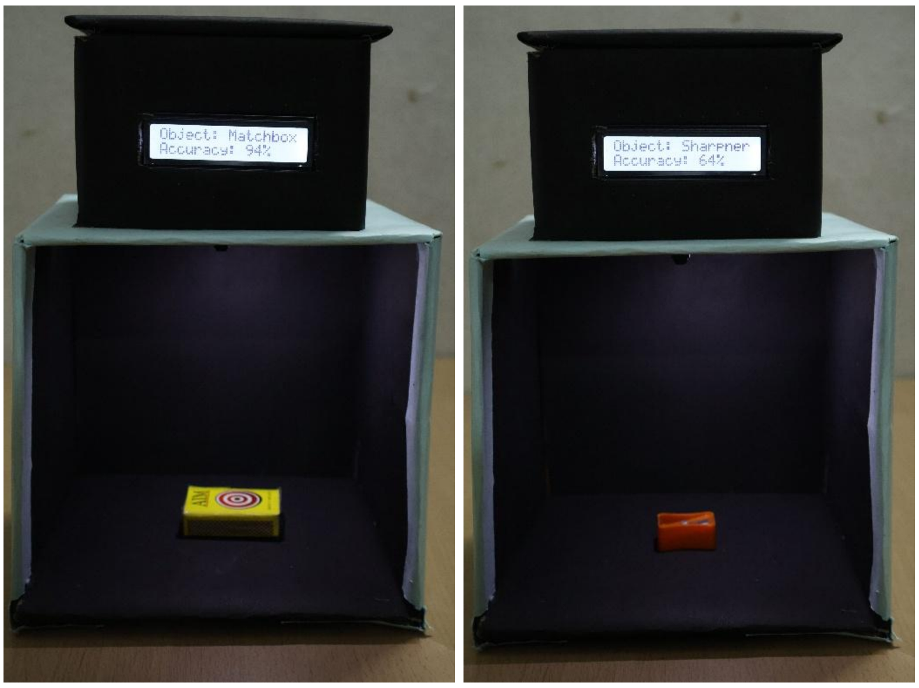


Figure 8: Displays the detection of a matchbox (94% accuracy) and a sharpener (64% accuracy) with results shown on the LCD screen, highlighting the system's reliability.

The ESP32-CAM successfully identified the objects, displaying their respective names and detection accuracies on the integrated LCD screen. The matchbox was detected with an accuracy of 94%, demonstrating the system's high precision for well-defined objects. The sharpener, though smaller and with fewer distinct features, was detected with an accuracy of 64%, indicating the system's capability to handle varied object sizes and complexities. These results validate the effectiveness of the system in real-time object recognition scenarios and its ability to provide reliable outputs for diverse objects.

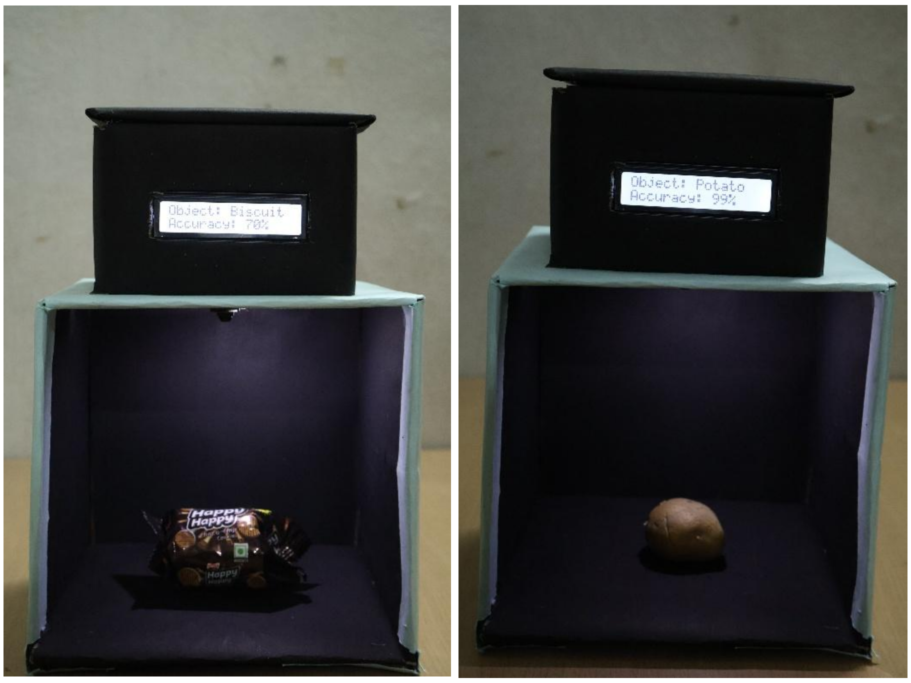


Figure 9: Displays a biscuit packet (70% accuracy) and a potato (99% accuracy), with results displayed on the LCD screen, showcasing accuracy across varied object textures.

The system was further tested with two contrasting objects: a biscuit packet and a potato, to evaluate its versatility in detecting objects with different textures and shapes. The biscuit packet, with its irregular and reflective surface, was detected with an accuracy of 70%, reflecting the challenges posed by shiny or intricate textures. On the other hand, the potato, with its simple and uniform structure, was detected with a high accuracy of 99%, demonstrating the system's efficiency in handling non-reflective and well-defined objects. These tests highlight the system's capability to adapt to various object types, ensuring reliable performance in diverse real-world scenarios.

5. Conclusions

The presented work successfully demonstrates the development and implementation of a real-time object detection system using the ESP32-CAM and an optimized YOLOv5 model, specifically designed for edge-based operation in resource-constrained environments. The theoretical framework and practical results validate the system's efficiency in accurately detecting and classifying objects with minimal latency.
The integration of the Edge Impulse platform for model training and quantization ensures compatibility with the ESP32-CAM's limited computational resources, enabling a lightweight, standalone solution without dependence on cloud services.

Experimental results show the system's reliability, with high F1 scores and negligible misclassifications, reflecting its potential for practical deployment in various applications such as smart surveillance, IoT-based automation, and industrial monitoring.
While the system performs well under static and controlled environments, future improvements could focus on handling dynamic scenarios, optimizing detection for complex scenes, and enhancing scalability. The findings from this project provide a robust foundation for designing similar real-time edge-computing systems and pave the way for further advancements in the field of lightweight AI-driven applications.

References

[1]. Shofia Priya Dharshini D., R.Saranya, S.Sneha. et al. "ESP32 Cam Based Object Detection & Identification with OpenCV." *Data Analytics and Artificial Intelligence*, vol. 2, no. 4, 2022, pp. 166–171.

[2]. Pradeep S, Nikhil Raghav V, and B. Kumar. "Animal Intrusion Detection Using ESP32 Cam and OpenCV." *International Journal of Innovative Science and Research Technology*, vol. 8, 2023, pp. 155–160.

[3]. D. Pullivarthi, and S. Raut. "Multiple Object Detection, Object Tracking, Lane Tracking, and Motion Detection Using ESP32-CAM." *International Journal for Research Trends and Innovation*, vol. 7, 2023, pp. 98–105.

[4]. Sredha Vinod.,P. Shakor. "Object Detection Using ESP32 Cameras for Quality Control of Steel Components in Manufacturing Structures." *International Journal of Advanced Engineering Research and Science*, vol. 5, 2022, pp. 29–34.

[5]. A R. Nair, G. Akaash , Varun D."AI-Based Wireless Display Data Extraction Using YOLO v5 Model." In *Lecture Notes in Networks and Systems*, Springer, 2023, pp. 105–110.

[6]. B. Satria, S. Karim, F. Ramadhani "Cloud Storage for Object Detection using ESP32-CAM." *International Journal of Emerging Technologies and Applications*, vol. 10, 2024, pp. 466–470.

[7]. V. Khotsyanovsky "Camera Image Processing on ESP32 Microcontroller with Help of Convolutional Neural Network," *Electronics and Control Systems*, vol. 4, no. 70, pp. 83–89, 2021.

[8]. Zhao, Z. Q., Zheng, P., Xu, S. T., & Wu, X. "Object Detection with Deep Learning: A Review." *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, 2019, pp. 3212-3232.

[9]. M. Omelchenko, V. Hotsyanivskyy. "Image Processing on ESP32 Microcontrollers Based on MobileNet Convolutional Neural Network," *ResearchGate*, June 2022.

[10]. Nihal, R. A., B. Yen, K. Itoyama, and K. Nakadai. "From Blurry to Brilliant Detection: YOLOv5-Based Aerial Object Detection with Super Resolution." *preprint arXiv:2401.14661*, 2024.