Development of an Edge-AI-Enabled Object Detection System for Real-Time Monitoring

Amitava Das
PhD Scholar
Academy of Scientific and Innovative Research(AcSIR), Ghaziabad-201002, India, CSIR-CMERI,
Durgapur-713209, India,
Email: amitavasdas@gmail.com

Swetasree Chattoraj and Deepro Sen
Institute of Engineering & Management (IEM),
Kolkata-700091, India.
Email:Deepro.Sen@iem.edu.in

Ravi Kant Jain
Senior Principal Scientist and Professor, *AcSIR,*
Intelligent System Engg. Group *CSIR-CMERI,*
Durgapur-713209, India
*Email: rjain.cmeri@csir.res.in; drravikantjain@gmail.com (ORCID:0000-0002-8106-1256)
*Corresponding author

***Abstract*- This paper presents a novel, low-cost, and scalable AI-powered object detection framework for real-time environmental monitoring and intelligent decision-making. The proposed system integrates an ESP32-CAM module for edge-level image acquisition and live video streaming with a lightweight HTTP-based web application that enables platform-independent camera control, real-time visualization, and image capture through any browser-enabled device. The key novelty of this work lies in its hybrid edge–desktop intelligence architecture, where computationally constrained embedded hardware is efficiently combined with a Python-based graphical user interface (GUI) and a custom-trained YOLOv5s deep learning model for localized real-time object detection, minimizing continuous cloud dependency. Experimental evaluation demonstrates that the proposed approach achieves a maximum detection accuracy of 88.95%. The integrated design offers an efficient solution for edge data acquisition, secure cloud storage, and localized deep learning inference. The results highlight the potential of embedded vision systems for applications such as mobile robotics, smart city infrastructure, and intelligent monitoring systems.**

***Keywords*-** Edge AI, Object Detection, ESP32-CAM, Cloud Platform, real time monitoring etc.

I. INTRODUCTION

Nowadays, AI-based real-time object detection on embedded systems is rapidly advancing, driven by the convergence of computer vision, edge AI, and the Internet of Things (IoT) [1,2]. IoT-based robotic surveillance systems capable of object recognition and obstacle avoidance and another autonomous IoT-enabled mobile robot employing Convolutional Neural Networks (CNN) and Q-learning have been proposed by Anitha et al. [3] and Saravanan et al. [4] for path planning. Further, Lopez-Alfaro et al. [5] and Medjdoubi et al. [6] have utilised AI/ML libraries and image processing techniques for object detection and smart surveillance robot for face recognition using edge computing respectively. Iqbal et al. [7] have integrated IoT and autonomous robots with Building Information Modelling (BIM) for construction monitoring, connecting an ESP32 microcontroller to Google Cloud for data storage and visualization. This integration plays a crucial role in addressing contemporary challenges in surveillance, automation, and robotics. Recent advancements in embedded hardware have facilitated the deployment of deep learning models on resource-constrained devices, thereby significantly reducing dependence on continuous cloud connectivity. Chang et al. [8] have presented an ESP32-based edge computing model to minimize cloud server load and latency through localized inference. This shift supports localized, low-latency decision-making, which is essential for many real-world applications. However, there remains a strong need for compact, cost-efficient vision systems capable of performing real-time, multi-object detection while minimizing resource usage, bandwidth, and latency like Shaker et al. [9] have focused on the implementation of a real-time object detection using YOLO models on Raspberry Pi for navigation in complex environments. To address this need, a compact and cost-effective vision system built around the ESP32-CAM microcontroller and a Python-based android mobile/web application is proposed. During the development of the system, a custom-trained YOLOv5 object detection model is optimized for real-time multi-object identification on devices with limited resources. The ESP32-CAM, equipped with an onboard camera and Wi-Fi module, serves as the primary edge device for live video streaming. A custom mobile-friendly web interface is developed using HTML, CSS, and JavaScript which enables users to control the ESP32-CAM, stream live video, capture frames, and securely upload selected images to Google drive using OAuth 2.0 authorization.

A hybrid novel architecture is proposed that unites edge data capture, secure cloud storage, and deep learning inference. The Python-based desktop client downloads images from Google Drive and processes them using the YOLOv5s model trained on the MS COCO 2017 dataset, which supports 80 object classes. Detection results are presented through an intuitive web-based graphical interface, displaying bounding boxes, class labels, confidence scores, FPS, and timestamps. Additionally, the system allows

saving of annotated images for further analysis, enhancing its utility in real-time AI-driven analytics.

YOLOv5 model provides several advantages that make it suitable for real-time object detection applications. It performs object localization and classification in a single forward pass, resulting in low latency and high inference speed. The model achieves a strong balance between detection accuracy and computational efficiency through optimized network architecture and improved training strategies. Additionally, YOLOv5 model offers lightweight and scalable variants, enabling deployment on both resource-constrained edge devices and high-performance systems.

The key contributions of this paper are as follows:

i. **Design and development of a novel and efficient edge AI-based object detection system** utilizing the ESP32-CAM microcontroller, tailored for real-time applications in resource-constrained environments.

ii. **Implementation of a hybrid novel system architecture** that integrates direct data acquisition from the edge device, a custom web-based interface for camera control, secure image upload to Google Drive using OAuth 2.0, and localized deep learning inference.

iii. **Deployment of a Python-based android mobile/web application by** incorporating a custom-trained YOLOv5s model for accurate, low-latency object detection, with real-time visual output rendered through an intuitive Tkinter graphical user interface (GUI).

The paper is organized as follows: Section II presents brief state of art report in the areas of edge AI, embedded systems, and real-time object detection. Section III proposes the novel system architecture and design. Section IV describes the implementation of the ESP32-CAM module and the custom web interface. The Python-based android mobile/web application is developed and the YOLOv5s model is implemented. Section V discusses the experimental results and performance evaluation. Finally, Section VI concludes the paper and future work are also mentioned.

II BRIEF STATE OF ART REPORT ON EDGE AI AND EMBEDDED SYSTEMS FOR OBJECT DETECTION

In recent years, extensive research has been conducted on IoT-enabled technologies for applications in environmental monitoring, surveillance, and mobile robotics. Jiang et al. [10] introduced an improved YOLOv4-tiny model to enhance feature extraction while reducing computational complexity. Fu et al. [11] have introduced a fog computing approach with a YOLOv3-based recognition system for improving perception which reduces hardware complexity. Arani et al. [12] have provided an in-depth analysis of real-time object detection networks, emphasizing robustness, energy efficiency, and deployment feasibility on edge platforms. Redmon et al. [13] have introduced the YOLO framework along with object detection as a single-pass regression problem for real-time inference. Zou et al. [14] have attempted on object detection using deep learning-based methods. Zhao et al. [15] have carried out work a YOLOv5-based system for wheat grain detection using sparse pruning and hybrid attention mechanisms, while Zhao et al. [16] have demonstrated YOLOv5's superior accuracy and performance for contraband detection in public security scenarios. Focusing on the ESP32-CAM platform, Imron et al. [17] have developed a cloud-enabled object detection system using Firebase and FOMO models for image capture and storage. Das et al. [18] have focused high-accuracy real-time face recognition on the ESP32-CAM under varying conditions. Dharshini et al. [19] have implemented object detection using OpenCV and YOLOv3 on ESP32-CAM for security and autonomous driving. Narwaria et al. [20] have developed a Python-integrated ESP32-CAM-based object detection system for surveillance and automation. Sulistiyowati et al. [21] have also developed a real-time sorting system with ESP32-CAM using Python and OpenCV for color detection and object counting. Pullivarthi et al. [22] have built a vision-based navigation system for colored object tracking using the ESP32 AI camera. Kawawaki et al. [23] have focused on multiple object tracking sytem based on fusion method for intelligent and real-time image processing where object tracking has been analysed. Patel et al. [24] have attempted on an IoT-enabled mobile robot for AI-powered object detection. Despite the substantial progress across individual components such as embedded vision, edge AI, and IoT connectivity. There remains a gap in the development of a unified, end-to-end IoT cloud-based system architecture. Specifically, existing work has not fully explored that integrates ESP32-CAM-based edge video acquisition, a mobile web interface for real-time interaction and cloud storage, and a Python based andriod mobile/web application for YOLOv5s-based object detection and analytics.

To bridge this gap, we propose a novel system that consists of the ESP32-CAM's built-in Wi-Fi and camera capabilities for efficient real-time data capture. The proposed architecture enables flawless data flow from edge AI device to cloud and desktop-based analytics which offers an integrated solution for supporting secure storage, localized inference, and AI-driven object recognition. The proposed approach demonstrates a scalable, low-latency, and resource-efficient platform suitable for diverse applications in remote monitoring, automation, and smart environments.

III. PROPOSED HYBRID SYSTEM ARCHITECTURE AND MATHAMTICAL MODEL FOR AI-BASED OBJECT DETECTION SYSTEM

The hybrid system architecture is designed as a hybrid model by combining edge data acquisition, cloud-based storage, and localizing AI based image processing method as shown in **Figure 1**. This section details the main components and their interconnections as given below;

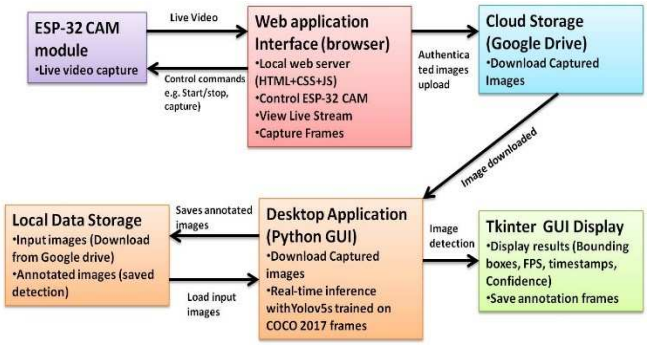

Figure 1. Block diagram for proposed hybrid edge AI system architecture

i. **ESP32-CAM Module:** It serves as the edge device for live video streaming and capturing individual frames. It wirelessly transmits data and receives control commands via the Wi-Fi network.

ii. **Mobile Device & Web Application:** It hosts a lightweight HTTP server to serve a custom HTML/CSS/JS web application. This interface allows users to control the ESP32-CAM, view the live stream, capture images, and securely upload selected frames to Google Drive.

iii. **Cloud Storage (Google Drive):** Google drive offers a secure and readily available solution for storing images. It allows verified uploads directly from a mobile device and enables convenient image downloads to a desktop for subsequent processing.

iv. **Android/Web Application (Laptop):** It runs the Python-based application code with a Tkinter GUI. This module is responsible for downloading images from Google Drive (or loading from local storage), performing real-time object detection using the YOLOv5s model, and displaying the annotated results.

v. **Local Data Storage:** It acts as a local repository on the desktop PC/laptop for both input images (downloaded from Google drive) and output images (annotated frames saved after detection).

Building upon the individual components described above, the flow chart for data acquisition, processing, and visualization are carried out as shown in Figure 2. This process maintains a real-time responsiveness and ensuring the reliable transfer of data across different operational environments. It effectively provides the strengths of each component, from the localized data capture at the edge to the centralized, powerful processing on the desktop/laptop which is facilitated by secure cloud.

For developing the mathematical model, given an input image, the goal is to detect objects in the image and classify them into predefined categories while localizing their positions using bounding boxes as given below;

a) Convolutional Layer: The YOLOv5s model, like other Convolutional Neural Networks (CNNs) [25], primarily uses convolutional layers to extract hierarchical features from the input image. These layers apply various filters to detect patterns, edges, textures, and ultimately, high-level semantic information crucial for object recognition.

b) Bounding Box Regression: Each bounding box is parameterized by a four-dimensional vector:

$$b=(x,y,w,h)$$

where x and y denote the center coordinates of the bounding box, and w and h represent its width and height, respectively.

Bounding box coordinates are predicted as given below:

$$\hat{b} = W_x + b \quad (1)$$

Where $\hat{b}$ is predicted bounding box, W_x and b are learnable network parameters.

The loss function for bounding box regression [26] is given by:

$$L_{reg}\left(\hat{b}, b\right) = \begin{cases} \frac{1}{2}\left\|\hat{b} - b\right\|^2, & if\ \frac{1}{2}\left\|\hat{b} - b\right\|^2 < 1, \\ \left\|\hat{b} - b\right\| - \frac{1}{2}, & \text{Otherwise} \end{cases} \quad (2)$$

c) Object Detection Loss Function: The overall loss function combines classification and localization losses:

$$L = L_{cls} + \lambda L_{reg} \quad (3)$$

Where L_{cls} denotes the classification loss, L_{reg} represents the bounding box regression loss, and λ is a weighting factor.

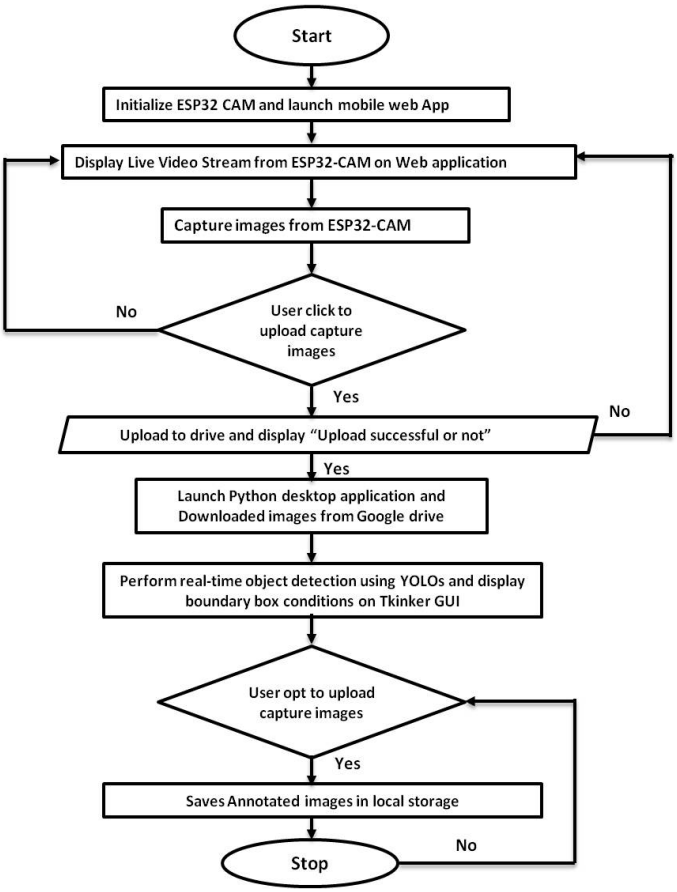

Figure 2. Flowchart for operates for data acquisition, processing, and visualization

III.PYTHON ANDROID/WEB APPLICATION AND YOLOV5S MODEL INTEGRATION

The Desktop PC/Laptop hosts the Python-based desktop application with its Tkinter GUI, handling image retrieval, object detection inference, and display of annotated results. Key libraries and frameworks include:

- **Python:** The primary programming language, chosen for its extensive libraries and suitability for machine learning.
- **Tkinter:** Python's standard GUI toolkit is used for building the interactive desktop application that displays detection results and provides user controls.
- **OpenCV (Open-Source Computer Vision Library):** Utilized for various image processing tasks such as loading and manipulating images, drawing bounding boxes, adding labels, and displaying the visual output.
- **PyTorch:** A powerful open-source machine learning framework, serving as the underlying engine for loading and running the deep learning model.

For image acquisition, the Python desktop application is designed to retrieve captured images from a local directory on the desktop PC. These images are manually transferred to this local storage (e.g., after being downloaded by the user from Google Drive or other sources). The training workflow generally involved in the following steps:

I. **Dataset Preparation:** Utilizing COCO 2017 dataset tailored for specific object detection needs.
II. **Pre-Processing & Data Augmentation:** Images underwent resizing, normalization, and various augmentation techniques to increase data diversity and improve model robustness.
III. **Model Training:** The pre-trained YOLOv5s model was trained for 100 epochs, optimizing its weights and biases to minimize the overall loss function.
IV. **Model Export:** The trained model was then saved in an optimized format (as a .pt file for PyTorch inference) suitable for efficient deployment and real-time object detection on the desktop application.

IV. IMPLEMENTATION OF THE ESP32-CAM MODULE AND CUSTOM WEB INTERFACE

The AI Thinker ESP32-CAM module serves as the primary edge device responsible for capturing visual data within the system as show in **Figure 3**. It integrates several key components critical for its operation:

- **ESP32 Microcontroller:** This powerful Wi-Fi and Bluetooth System-on-Chip (SoC) forms the core of the module. It handles all processing tasks.
- **OV2640 Camera Module:** A 2-megapixel (UXGA) CMOS image sensor responsible for converting light into digital image data.
- **Wi-Fi Antenna:** Integrated onto the PCB, this antenna facilitates wireless communication, enabling the ESP32-CAM to connect to the local Wi-Fi network and transmit data.
- **PSRAM (Pseudo Static RAM):** Typically 4MB of external PSRAM is included to provide crucial additional memory for buffering higher-resolution images.
- **Built-in Flash LED:** A small LED flash is often included for illumination in low-light conditions or for indication purposes.

Further, a mobile web application is applied using following components.

- **HTML (HyperText Markup Language):** Structures the content and layout of the user interface.
- **CSS (Cascading Style Sheets):** Styles the web page.

Figure 3. AI Thinker ESP32-CAM module

- **JavaScript:** Handles the dynamic functionalities, including sending control commands to the ESP32-CAM, displaying the live video stream, managing image capture events, and performing client-side logic for authentication and image upload via the Google Drive API.
- **Local HTTP Server App:** A lightweight HTTP server application (e.g., Simple HTTP Server for Android or similar) runs on the mobile device to host and serve these web files locally, allowing the user to access the interface through a browser on the same device.
- **Google Drive API:** It enables the mobile web application to upload captured images to a designated Google Drive folder and allows the Python desktop application to download these images for processing.
- **Google OAuth 2.0** is used for secure use authorization.

V. RESULTS AND DISCUSSIONS

This subsection primarily features visual evidence of the system in operation, illustrating its end-to-end functionality from live stream monitoring to object detection results. **Figure 4** showcases the mobile interface displaying a live video stream from the camera module. Crucially, this image also demonstrates the successful execution of an image capture and upload, confirmed by the "Upload successful!" notification. This highlights the mobile application's role in facilitating both monitoring and secure data transfer to cloud storage. Once an image is captured via the mobile application and made available (uploaded to Google Drive and then manually downloaded to the desktop), it is then loaded into the Python desktop application for object detection.

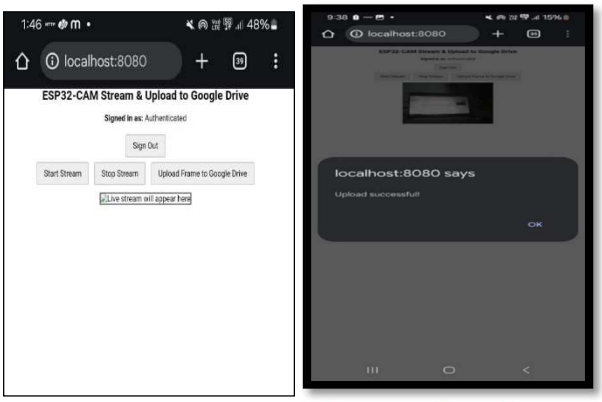

Figure 4. Mobile web application for successful image detection

The system demonstrated robust performance in identifying a wide range of common objects from the COCO dataset with high accuracy. Some of the detected objects are shown in **Figure 5.**

(a) Person, chair, bottle detected indoors.

(b) Computer/Laptop devices detected indoors

(c) Car, dog out detected outdoors

Figure 5. Detected objects

The experimental data of different object detections are summarised in Table 1. The inference speed has analysed as shown in **Figure 6.** The following graphs illustrate the system's frame per second (FPS) performance during operation. as shown in **Figure 7.** This scatter plot illustrates the relationship between the FPS at which an image was processed and the corresponding detection accuracy percentage. This analysis helps determine if there is a correlation between the system's operational speed and the precision of its object detections. The maximum accuracy is attained upto 91.20%.

Table 1. Experimental Data for System Performance Evaluation

SL. No.	Object class	FPS	Accuracy (%)	Capture duration (s)
1	Car	3.1	62.10	3.23
2	Chair	4.1	62.70	2.44
3	Person	5.0	63.10	2.00
4	Laptop	5.8	69.10	1.72
5	Keyboard	3.4	71.20	2.94
6	Bottle	3.3	80.10	3.03
7	Dog	3.1	86.80	3.23
8	Umbrella	3.4	88.20	2.94
9	Mouse	4.8	91.20	2.08

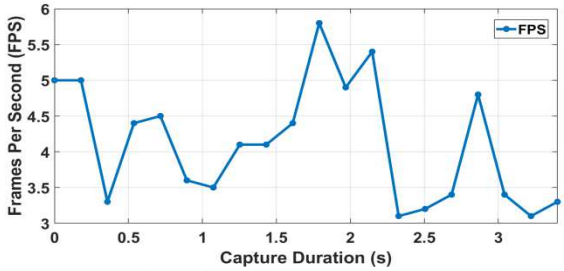

Figure 6. Observed FPS fluctuation over capture duration.

The detected average FPS rate per object is shown in **Figure 8.** This bar chart provides insight into the average frames per second achieved when various object classes are detected. Object dection accuracy % with FPS is also plotted

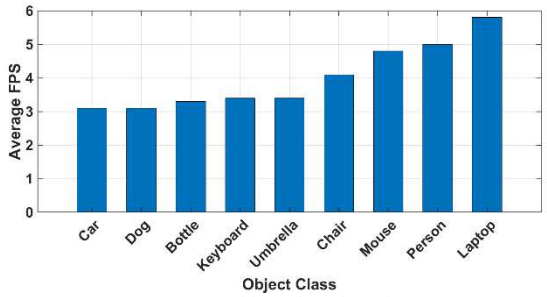

Figure 7. Average object class detection FPS rate

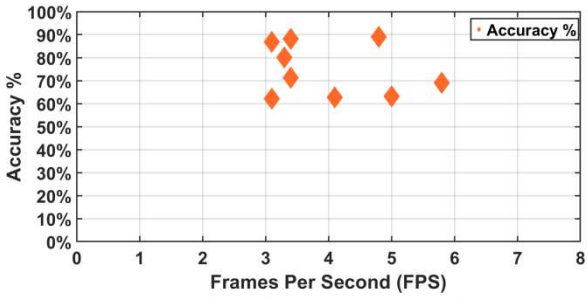

Figure 8. Object detection accuracy rate in FPS

A comparative study of the proposed model and other object detection models is given in Table 2.

Table 2. A comparative study of the proposed model and other object detection models

Features	Proposed AI model	YOLO v8 [27]	MobileNet-SSD [28]
Architecture type	CNN- based	CNN-based with an improved head	MobileNet backbone with SSD
Model size	Small (14 MB)	Medium (10–25 MB)	Very small (5–7 MB)
Detection accuracy	91.20%	90-92 %	70-75%
Performance	Faster (FPS: 3–5.8)	High (FPS: 10–30)	Very high (FPS: 15–60)
Inference speed	Faster	Faster	Very fast
Advantage	Faster speed and high detection accuracy	Improved accuracy and efficiency	Minimal resource consumption

The implemented system successfully integrates an end-to-end data flow consisting of an ESP32-CAM for live streaming and image capture, a mobile web application for user interaction and secure cloud storage via Google Drive, and a Python desktop/android/web application for robust object detection using the YOLOv5s model.While the system effectively met its primary objective of reliable object identification, challenges included optimizing data transfer efficiency and achieving higher inference speeds for true real-time applications on current hardware, aspects that could be refined in future iterations. During experimementation, it is observed that the latency rate is less during image transfer from ESP32CAM to IoT cloud.

The novelty of the proposed work is highlighted below;

a. Enables secure, low-cost deployment using ESP32-CAM with AI cloud integration.

b. Introduces an asynchronous AI processing architecture for object detection.
c. Reduces dependency on continuous cloud inference by enabling localized detection.

VI. CONCLUSION

In this paper, a novel AI-based object detection system is successfully designed and implemented which integrates low-cost embedded vision hardware with localized deep learning intelligence. The proposed architecture combines an ESP32-CAM module for real-time video streaming and image acquisition with an Android/web-based application that enables intuitive user interaction and secure cloud storage using Google Drive. A Python-based desktop application incorporating a fine-tuned YOLOv5s model facilitates robust object detection across diverse object categories, including persons, chairs, bottles, plants, cars, and animals, under varying environmental conditions. Experimental results confirm reliable system performance, achieving a maximum detection accuracy of 88.95% with consistent inference speed. The novelty of this work lies in its hybrid edge–desktop intelligence framework, which effectively balances computational constraints and detection accuracy while reducing reliance on continuous cloud connectivity. The proposed solution demonstrates strong potential for scalable deployment in applications such as surveillance, automation, robotics, and intelligent monitoring systems.

ACKNOWLEDGMENT

The authors express their gratefulness to the Director of CSIR-CMERI, Durgapur, India, for granting consent to publish this paper. Miss Swetasree Chattoraj was involved in this work during her summer internship at CSIR-CMERI, Durgapur.

REFERENCES

[1] Y.Y. Chen, Y.H. Lin, Y.C. Hu, C.H. Hsia, Y.A. Lian, and S.Y. Jhong, "Distributed real-time object detection based on edge-cloud collaboration for smart video surveillance applications", *IEEE Access*, vol. 10, 2022, pp. 93745-93759.

[2] S.H. Hozhabr and R. Giorgi, "A survey on real-time object detection on FPGAs", *IEEE Access*, vol. 11, 2025, pp. 38195-38238.

[3] K. Anitha, P. Deepthy, and S. Kambhampati, "IoT-based intelligent robot design and analysis for real-time monitoring and control", *Aut Aut Res. J.*, vol. XII, no. XII, Dec. 2021, pp. 233-236.

[4] M. Saravanan, P.S. Kumar, and A. Sharma, "IoT enabled indoor autonomous mobile robot using CNN and Q-Learning", in *Proc. IEEE Inter. Conf. on Industry 4.0, Artif. Intell. and Comm. Tech. (IAICT)*, Bali, Indonesia, 2019, pp. 7-13.

[5] G.A. Lopez-Alfaro, L.A. Hernandez-Fernandez, U. Vidal-Espitia, L. M. Rodriguez-Vidal, J.P. Serrano-Rubio, and C.M. Hernández-Mendoza, "Mobile robot for object detection using an IoT system", in *Proc. IEEE Inter. Autu. Meet. on Power, Electro. and Comput. (ROPEC)*, Ixtapa, Mexico, 2020, pp. 1-6.

[6] A. Medjdoubi, M. Meddeber, and K. Yahyaoui, "Smart city surveillance: Edge technology face recognition robot deep learning based," *IJE Trans. A: Basics*, vol. 37, no. 1, pp. 25–36, 2024. doi: 10.5829/ije.2024.37.01a.03.

[7] F. Iqbal, S. Ahmed, F. Amin, S. Qayyum, and F. Ullah, "Integrating BIM–IoT and autonomous mobile robots for construction site layout printing", *Buildings*, vol. 13, no. 9, Art. no. 2212, pp. 1-17, 2023. doi: https://doi.org/10.3390/buildings13092212

[8] Y.H. Chang, F.C. Wu, and H.W. Lin, "Design and Implementation of ESP32-Based Edge Computing for Object Detection", *Sensors*, vol. 25, no. 6, Art. no. 1656, 2025. doi: 10.3390/s25061656.

[9] A.M. Shaker, "Design and implementation of real-time AI object detection for mobile robot applications", M.S. thesis, Dept. Mechatronics Eng., Univ. of Mosul, Iraq, May 2024.

[10] Z. Jiang, L. Zhao, S. Li, and Y. Jia, "Real-time object detection method based on improved YOLOv4-tiny", *Journal of Network Intelligence*, vol. 7, no. 1, 2022, pp. 59-72. Available: https://doi.org/10.48550/arXiv.2011.04244.

[11] M. Fu, S. Sun, K. Ni, and X. Hou, "Mobile robot object recognition in the Internet of Things based on fog computing", in *Proc. 2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, Lanzhou, China, Nov. 18-21, 2019, pp. 1838–1842.

[12] E. Arani, S. Gowda, R. Mukherjee, O. Magdy, S. Kathiresan, and B. Zonooz, "A comprehensive study of real-time object detection networks across multiple domains: A survey", *arXiv preprint arXiv:2208.10895*, Aug. 2022. doi: 10.48550/arXiv.2208.10895.

[13] Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, Apr. 8, 2018.

[14] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object detection in 20 Years: A survey", *arXiv preprint arXiv:1905.05055*, 2019. doi: 10.48550/arXiv.1905.05055.

[15] W. Zhao, S. Liu, X. Li, X. Han, and H. Yang, "Fast and accurate wheat grain quality detection based on improved YOLOv5", *Comput. Electron. Agric.*, vol. 202, Art. no. 107426, 2022. Available: https://doi.org/10.1016/j.compag.2022.107426.

[16] Y. Zhao and S. Wang, "Research on real-time object detection based on Yolo algorithm", *Highlights in Science Engineering and Technology*, vol. 7, pp. 323–331, Aug. 2022. doi: 10.54097/hset.v7i.1091.

[17] I. Imron, B. Satria, S. Karim, and F. Ramadhani, "Cloud storage for object detection using ESP32-CAM", *TEPIAN*, vol. 5, no. 2, 2024, pp. 50–57. doi: 10.51967/tepian.v5i2.2994H.

[18] B. Das and K.K. Halder, "Face recognition using ESP32-CAM for real-time tracking and monitoring", in *Proc. 2024 Int. Conf. Advances in Computing, Communication, Electrical, and Smart Systems (iCACCESS)*, Mar. 2024, pp. 1-6. doi: 10.1109/iCACCESS61735.2024.10499606.

[19] S. P. Dharshini, R. Saranya, and S. Sneha, "ESP32-CAM based object detection and identification with OpenCV", *Data Analytics and Artificial Intelligence*, vol. 2, no. 4, 2022, pp. 166-171. doi: 10.46632/daai/2/4/31.

[20] R.P. Narwaria, A. Ahirwar, A.K. Prajapati, A. Kumar, and A.K. Tiwari, "Smart object detection using ESP32-CAM based on YOLO algorithm", in *Proc. 2024 Second International Conference on Intelligent Cyber Physical Systems and Internet of Things (ICoICI)*, Coimbatore, India, 2024, pp. 817-820. doi: 10.1109/ICoICI62503.2024.10696374.

[21] I. Sulistiyowati, H.M. Ichsan, and I. Anshory, "Object sorting conveyor with detection color using ESP-32 camera python based on open-CV", *Journal of Electrical Engineering and Computer Sciences*, vol. 9, no. 1, 2024, pp. 61–68,. doi: 10.54732/jeecs.v9i1.7.

[22] D. Pullivarthi, S. Raut, S. Maurya, and P. Dhole, "Multiple object detection, object tracking, lane tracking, and motion detection shadow robot based on computer vision", *School of Engineering and Science*, MIT ADT Univ., Pune, India, 2023.

[23] Y. Kawawaki, Y. Yamakawa, high-speed multiple object tracking based on fusion of intelligent and real-time image processing", *Sensors*, vol. 25, 2025, p. 3400. doi: https://doi.org/10.3390/s25113400.

[24] K.S. Patel, A. Das, S. Bose and R.K. Jain, "An IoT-Enabled Mobile Robot for AI-Powered Object Detection", *IEEE 3rd International Conference on Computer, Electronics, Electrical Engineering and their applications (IC2E3 2025)*, NIT Uttarakhand, India, 15-16 May 2025. doi:10.1109/IC2E365635.2025.11166763.

[25] Y. Li, Y. Chen, J. Li and H. Zhang, "A robust approach for industrial small-object detection using an improved faster R-CNN-based Model", *Sci. Rep.*, vol. 11, 2021, p. 2805, https://doi.org/10.1038/s41598-021-02805-y

[26] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, "Distance-IoU loss: Faster and better learning for bounding box regression," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 7, pp. 3703–3715, Jul. 2022, doi: 10.1109/TPAMI.2020.3017041.

[27] Liu, W. et al., SSD: Single Shot MultiBox Detector", In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, vol 9905. 2016. Springer, doi: https://doi.org/10.1007/978-3-319-46448-0_2.

[28] N. P. Jouppi et al., "In-datacenter performance analysis of a tensor processing unit," *IEEE Micro*, Vol. 38, No. 2, 2018, pp. 10-19. doi: 10.1109/MM.2018.022071134