Journal of Engineering and Technology for Industrial Applications

ITEGAM-JETIA

Manaus, v.11 n.55, p. 293-303. September/October, 2025.
DOI: https://doi.org/10.5935/jetia. v.11 n.55.2714

ISSN ONLINE: 2447-0228

RESEARCH ARTICLE OPEN ACCESS

COMPARATIVE ANALYSIS OF YOLOv3, MOBILENET-SSD, AND EFFICIENTDET FOR REAL-TIME PERSON DETECTION IN LOW-RESOLUTION IMAGES

Mhd. Idham Khalif[1] and Raden Deiny Mardian[2]

[1, 2] Electrical Engineering, Universitas Trisakti, Tomang, West Jakarta, Indonesia

[1]https://orcid.org/0009-0002-6911-9678, [2]https://orcid.org/0000-0001-5090-4604

Email: idham.khalif@trisakti.ac.id, deiny_wp@trisakti.ac.id

ARTICLE INFO

Article History
Received: September 19, 2025
Revised: October 7, 2025
Accepted: October 9, 2025
Published: October 31, 2025

Keywords:
Real-time detection,
Person detection,
Low-resolution camera,
ESP32-CAM

ABSTRACT

Object detection using low-resolution cameras is increasingly important for resource constrained and low-cost systems. This study evaluates the performance of three pre-trained machine learning models from the COCO dataset YOLOv3, MobileNet-SSD, and EfficientDet for human detection at three image resolutions: 160×120 (QQVGA), 320×240 (QVGA), and 640×480 (VGA). Low resolutions pose challenges due to limited visual and spatial details, which can reduce detection accuracy. Results show that YOLOv3 achieves the highest accuracy and consistency across all resolutions but demands higher computational resources, reflected in increased CPU usage, higher latency, and lower frame rates (FPS). MobileNet-SSD provides a balanced trade-off between accuracy and efficiency, making it suitable for real-time applications in power-constrained environments. EfficientDet, although the most lightweight in computation, shows lower detection accuracy. This study emphasizes empirical evaluation under real-world deployment constraints, following the zero-shot performance approach, where pre-trained models are applied directly to low-resolution edge data without retraining. The findings offer practical insights into the trade-offs among accuracy, efficiency, and latency, guiding model selection for IoT and edge computing applications in resource-limited settings. This work contributes to understanding suitable model choices for low-resolution human detection in real-world scenarios.

I. INTRODUCTION

The advancement of Artificial Intelligence (AI), Machine Learning (ML), and the Internet of Things (IoT) has accelerated rapidly, underpinning digital transformation across various sectors. These technologies support intelligent applications in embedded systems, automation, surveillance, and real-time decision-making, aligning with Indonesia's national strategy to accelerate digital transformation as outlined in Presidential Regulation No. 82 of 2023 [1].

Machine Learning enables systems to learn patterns from data and make decisions without explicit programming, supporting applications such as image classification, natural language processing, and anomaly detection. In IoT and edge computing, ML facilitates object detection, camera-based surveillance, and environmental monitoring, where decision intelligence contextualizes real-time data to support automated responses [2]. Its impact extends into economics, law, and healthcare, improving prediction, risk management, legal review, and personalized diagnosis [3].

In digital image processing, ML has significantly advanced automatic object detection and recognition, using techniques such as segmentation, classification, and feature extraction [4]. Popular models like You Only Look Once (YOLO), Single Shot MultiBox Detector (SSD), and Faster R-CNN offer different trade-offs between speed and accuracy [5]. Recent studies emphasize that low-resolution image recognition requires robust datasets and optimization strategies; for instance, YOLO can achieve real-time maritime object detection on edge computing when appropriately optimized [6], [7].

Low-resolution images (below 720p), such as QQVGA (160×120), QVGA (320×240), and VGA (640×480), challenge detection due to loss of fine details, yet they remain widely used in IoT devices and edge computing for power efficiency and bandwidth constraints [8-10]. Applying ML models under such constraints is challenging because most models require high-quality data and computational resources [11], [12]. Nevertheless, edge computing combined with low-resolution inputs can reduce inference latency and computational load while enabling real-time detection [12], [13].

Building on these insights, this study compares YOLOv3, MobileNet-SSD, and EfficientDet for real-time person detection using low-resolution images from an ESP32-CAM processed on an ARM-based edge computing. Evaluation metrics include average confidence, CPU utilization, frames per second (FPS), inference latency, and detection success, with lighting intensity as an additional factor. Unlike studies introducing new architectures, this work emphasizes empirical evaluation under resource-constrained conditions, reflecting real-world deployment and zero-shot performance scenarios [14].

II. MATERIAL AND METHODS

II.1 DATASET

The availability of datasets is a fundamental requirement for developing and applying computer vision-based methods [14]. The dataset used in this study is COCO (Common Objects in Context), which is one of the standard and widely adopted datasets for object detection tasks. The models employed in this research were pre-trained on COCO, with a specific focus on the person class to support the evaluation of human detection. Out of the 80 object categories available in COCO, this study utilized only the annotations for the person class, totaling 66,808 instances, in line with the research objective of emphasizing human detection in low-resolution images.

Figure 1: Sample dataset for person class images from the COCO dataset.
Source: [15].

Figure 1 shows a sample from the COCO dataset in the person class. This dataset provides a wide variety of poses, sizes, and backgrounds that reflect real world conditions, making it highly useful for training and evaluating person detection models in the context of edge computing and low-resolution cameras. By using only the person category, this study focuses on analyzing the performance of ML models for real-time human detection, reducing the complexity of multiclass classification and enabling more accurate evaluation of efficiency and accuracy under low-resolution conditions. The dataset also allows testing model robustness against variations in lighting, viewing angles, and object density, which are critical factors in implementing IoT-based camera systems.

II.2 MODEL PRE-TRAINED MACHINE LEARNING (ML)

The ML models used in this study are pre-trained models, which have been trained beforehand and can be implemented directly without the need for retraining. Pre-trained models not only unify semantic representations across multiple tasks and languages but also demonstrate high-level capabilities approaching human intelligence [16]. With this approach, the training and initial testing phases are unnecessary, allowing the study to focus directly on the implementation and evaluation of model performance in real time on edge computing with low-resolution inputs.

II.2.1 YOLOv3

YOLOv3 (You Only Look Once Version 3) is one of the most popular and widely used real-time object detection models in various computer vision applications. The model employs the Darknet-53 architecture as its backbone, consisting of 53 convolutional layers designed to balance accuracy and inference speed. As a one-stage detection model, YOLOv3 processes the entire image in a single inference, in contrast to two-stage approaches such as Faster R-CNN, making it highly efficient for real-time applications. YOLOv3 achieves high detection accuracy with fast inference time, making it suitable for real-time environments [17]. Previous studies have applied YOLOv3 for real-time object detection tasks, reporting strong performance; for example, with an input resolution of 512×512

pixels, the model achieved an accuracy of 74.1% [18] In the context of this study, YOLOv3 is used in the form of a pre-trained model trained on the COCO dataset. Despite its larger model size than the lightweight version, YOLOv3 remains relevant for mid-range edge computing devices due to its ability to detect objects with good precision.

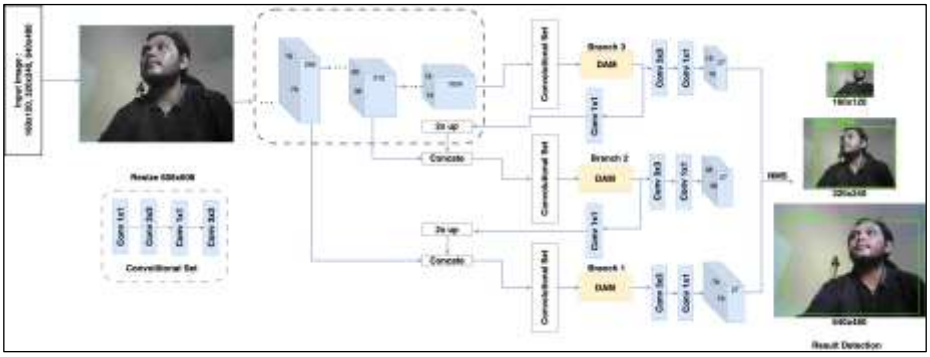

Figure 2: Architecture and result of YOLOv3 for human detection.
Source: Authors, (2025).

Figure 2 is illustrates the architecture and result of YOLOv3 applied in this study to detect human objects, starting from input images to final output results. The input images, originally in various resolutions such as 160×120 pixels, 320×240 pixels, or 640×480 pixels, are first uniformly resized to 608×608 pixels as part of a standard pre-processing step. These images are then processed by the network backbone, which extracts hierarchical features using a series of convolutional blocks composed of 1×1 and 3×3 filters, forming the basic convolutional sets. The network leverages the Feature Pyramid Network (FPN) principle by utilizing feature maps at three different scales—76×76 pixels, 38×38 pixels, and 19×19 pixels —to enable detection of objects at multiple sizes. Feature maps from deeper layers (low-resolution but rich in context) are up sampled and con-catenated with those from shallower layers (high-resolution but rich in spatial detail), enhancing the net-work's ability to extract comprehensive semantic and spatial information.

Following this multi-scale feature fusion, three detection branches (Branch 1, Branch 2, Branch 3) are employed, each equipped with convolutional layers and a Dilated Attention Module (DAM). The DAM helps the network focus on the most relevant areas, thereby improving the accuracy of bounding box pre-dictions, object confidence scores, and class probabilities. Finally, all predicted bounding boxes from the three branches are refined using Non-Maximum Suppression (NMS) to remove overlapping detections and retain the most relevant boxes for each detected object. The resulting output is a clear visualization of the detected human objects, along with the average confidence values for person detection using YOLOv3. Performance evaluations are based on confidence scores, mean Average Precision (mAP), and Precision-Recall (P-R) metrics for object detection [19].

II.2.2 MobileNet-SSD

MobileNet-SSD is the first TensorFlow-based deep learning model designed specifically for mobile devices. Due to its simpler and more computationally efficient architecture compared to other transfer learning models, such as VGG and ResNet, the name "Mobile" reflects its suitability for mobile applications [20]. In this study, human detection in images is performed using the MobileNet architecture combined with SSD (Single Shot MultiBox Detector) as the backbone of the object detection system. This combination enables the model to detect both the presence and location of humans in real-time, even on devices with limited computational resources. MobileNet-SSD, advantages in efficiency and speed make it highly suitable for various applications, including camera-based security surveillance systems, traffic monitoring, and augmented reality, all of which require fast and responsive human detection.

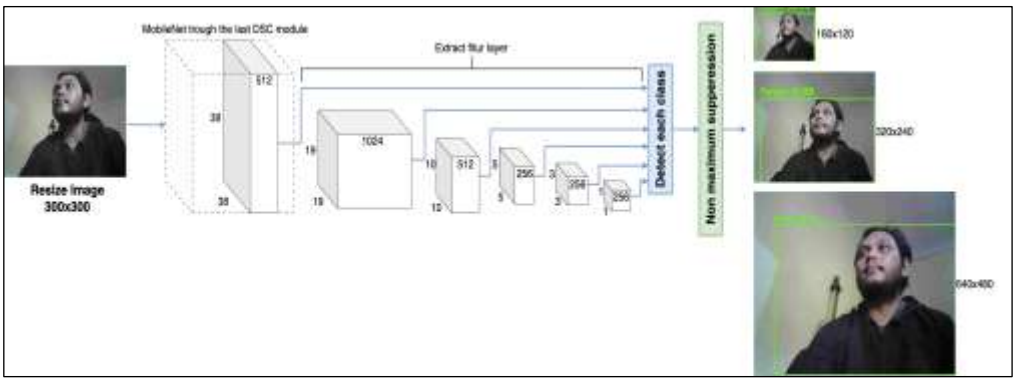

Figure 3: Architecture and results of MobileNet-SSD for human detection.
Source: Authors, (2025).

Figure 3, illustrates the workflow and results of the MobileNet-SSD architecture for performing object detection on devices with computational limitations, such as edge computing-based cameras. Similar to previous studies, three input resolutions—160×120 pixels, 320×240 pixels, and 640×480 pixels—are used. The process begins with resizing the input image to 300×300 pixels, which is then forwarded to the MobileNet-SSD backbone to extract visual features. MobileNet-SSD employs Depthwise Separable Convolution (DSC) up to the final block to produce a spatial feature representation of 38×38×512.

Furthermore, features from several layers (multiscale feature maps) are extracted at various spatial sizes—19×19, 10×10, 5×5, 3×3, and 1×1—with feature depths of 1024, 512, and 256 channels. This strategy enables MobileNet-SSD to perform object detection at multiple scales, from small to large objects, by utilizing high-resolution feature maps for spatial details and low-resolution feature maps for semantic context. Each of these features is then sent to the detection block (Detect Each Class), which predicts the bounding box location and object class directly in one step, without the region proposal stage used in Faster R-CNN. The detection results from all layers are combined and filtered through a Non Maximum Suppression (NMS) process to eliminate redundancy and retain only predictions with the highest confidence. The figure on the right shows the final result of human object detection.

II.2.3 EfficientDet

EfficientDet is a scalable and efficient object detection architecture that achieves state-of-the-art accuracy with significantly fewer parameters and FLOPs compared to previous detectors [21]. This model introduces two key innovations: BiFPN (Bidirectional Feature Pyramid Network), which enables efficient multi-scale feature fusion by allowing easy and weighted feature aggregation across different lev-els, and compound scaling, which systematically scales input resolution, network depth, and feature width in a balanced manner. These innovations allow EfficientDet to maintain high detection accuracy while substantially reducing computational demands, making it highly suitable for deployment in re-source-constrained environments, such as edge computing and IoT-based systems.

EfficientDet's design emphasizes both performance and efficiency, enabling it to operate effectively on devices with low power consumption and limited camera resolution. This makes it ideal for real-time ap-plications such as visual monitoring, autonomous vehicles, smart city infrastructure, and intelligent surveillance systems. Its ability to adapt to various hardware constraints without sacrificing detection quality has made it a preferred choice in many practical implementations. Moreover, EfficientDet supports a range of model sizes (D0 to D7), giving developers flexibility to choose the appropriate balance between speed and accuracy based on application requirements. This scalability ensures that EfficientDet can be optimized for both lightweight mobile devices and high-performance servers. Overall, EfficientDet represents a significant advancement in object detection, combining cutting-edge accuracy with practical efficiency for modern AI systems.

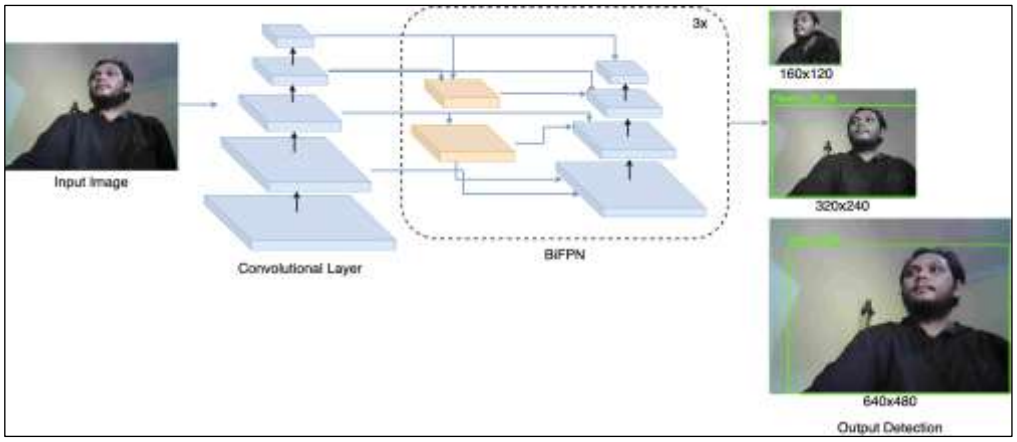

Figure 4: Architecture and results of EfficientDet for human detection.
Source: Authors, (2025).

Figure 4 shows the EfficientDet architecture and its results, demonstrating the model's ability to detect human objects efficiently and scalably through a combination of a lightweight backbone and an intelligent feature fusion mechanism. The process begins with an input image, which is then processed through several convolutional layers using the EfficientNet backbone to extract features at various resolution levels. These features are subsequently combined through the Bi-directional Feature Pyramid Network (BiFPN), a key component of the EfficientDet architecture that enables efficient feature fusion from top to bottom and vice versa.

BiFPN selects the most relevant feature information by weighting important information paths, resulting in a richer feature representation without significantly increasing computational complexity. After several iterations of BiFPN, the enriched features are used by the detection head to generate predictions of the locations and classes of objects in the image. In the architecture figure, the final result demonstrates successful object detection even when the input comes from a low-resolution camera, making EfficientDet highly suitable for applications in edge computing monitoring systems and IoT devices with limited power and computational resources.

II.3 MEASUREMENT AND HARDWARE SETUP

In this study, measurements and testing were carried out using real-time scenarios with low-resolution camera devices built on the ESP32-CAM. This device was chosen because it can transmit images directly via Wi-Fi, making it suitable for edge computing monitoring scenarios with limited resources. The camera captures images continuously, which are then analyzed using a lightweight ML-based object detection model to identify human presence in the test environment. Testing was conducted under various lighting conditions and perspectives to evaluate detection performance in situations that reflect real-world conditions. The system architecture is designed to maintain computing and power efficiency while providing accurate and responsive detection results in real time. The results of these tests are used to evaluate the effectiveness of the model in detecting objects from low-resolution images and to assess the stability of data transmission and overall system power efficiency.

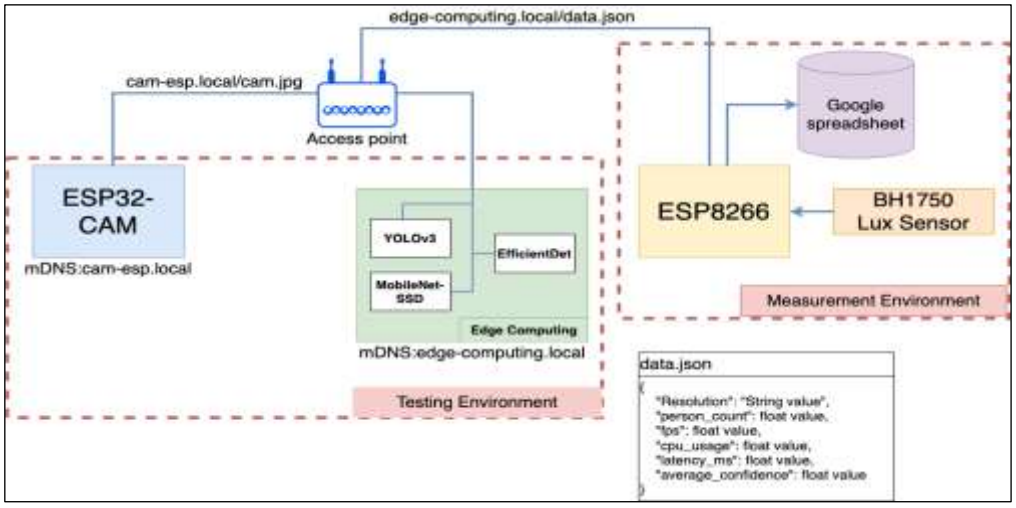

Figure 5: Architecture testing and measurement.
Source: Authors, (2025).

Figure 5, illustrates the architecture used in this study, consisting of two environments: testing and measurement. All devices are connected to a single access point within the same local network. In the testing environment, ESP32-CAM devices are configured to capture images in three resolutions: 160×120 (QQVGA), 320×240 (QVGA), and 640×480 (VGA), all in .JPG format. Edge computing devices are set up using the Multicast Domain Name System (mDNS) protocol, which allows dynamic device naming without relying on a unicast DNS server. mDNS simplifies device access even when IP addresses change across networks.

Captured images are continuously sent to the edge computing system, where they are processed using the selected ML models to detect human objects. Detection results are formatted in .JSON to facilitate communication between the testing and measurement systems. In the measurement environment, an ESP8266-based microcontroller measures ambient light intensity (lux) around the camera to evaluate the impact of lighting on model performance. Measurement data are transmitted to Google Spreadsheet via an internet connection in a time-series format for further analysis. Figure 6 is a show the setup of the testing device using ESP32-CAM and the measurement device using an ESP8266 microcontroller with a BH1750 lux sensor.

Figure 6: Camera and measurement device.
Source: Authors, (2025).

III. RESULT AND DISCUSSION

Testing was conducted in a controlled environment, specifically in a closed room with artificial lighting. The measurements showed that light intensity in the room ranged from 40 to 145 lux and was evenly distributed. Each test session lasted 30 minutes, with data collected periodically every minute. This procedure was repeated for each test scenario.

III.1 AVERAGE CONFIDENCE

During the deployment of object detection models, a confidence score threshold is set to reduce false positives and ensure that each predicted object is associated with an acceptable level of certainty [22]. This threshold is compared across ML models at various resolutions to assess stability and accuracy. In addition, tests measure human detection performance over a 30-minute period under resource constrained conditions.

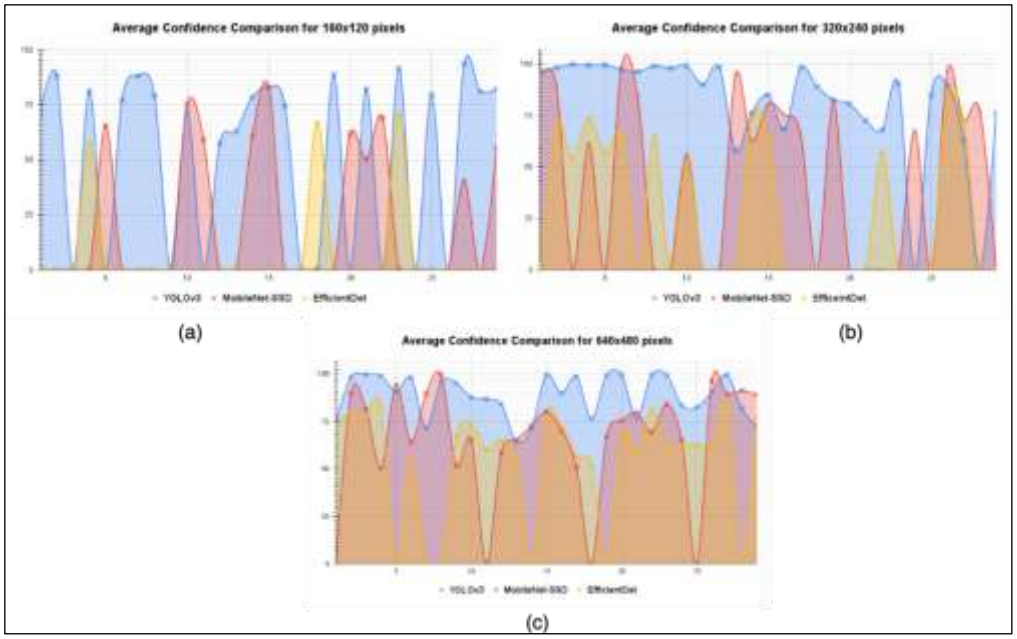

Figure 7: Comparison of average confidence values for each ML model: (a) 160×120 pixels (QQVGA). (b) 320×240 pixels (QVGA). (c) 640×480 pixels (VGA).
Source: Authors, (2025).

Figure 7a shows the results at 160×120 pixels (QQVGA). YOLOv3 achieved the highest average confidence (50.42%) and detection success rate (63.33%) during 30 minutes of testing. MobileNet-SSD followed with 20.7% confidence and a 33.33% success rate, while EfficientDet scored the lowest at 6.52% confidence and 10% success. These results indicate that YOLOv3, with its deep architecture and multiscale detection, can still recognize human objects reasonably well at very low resolutions. In contrast, MobileNet-SSD struggles due to limited feature extraction, while EfficientDet's lightweight design restricts its ability to detect small or low-quality objects. This highlights the importance of aligning model complexity with input quality, especially in low-resolution edge deployments.

At 320×240 pixels (QVGA), as shown in Figure 7b, YOLOv3 again led with 81.77% confidence and a 93.33% success rate. MobileNet-SSD achieved 44.04% confidence and 56.67% success, while EfficientDet reached 28.2% confidence and 43.33% success. YOLOv3's consistent performance stems from its robust feature extraction, making it effective even under varying lighting or object positions. MobileNet-SSD remains practical for real-time tasks but exhibits fluctuations in detection reliability. EfficientDet's lower performance reflects its trade-off between efficiency and feature depth, suggesting the need for model adaptation when applied to moderate-resolution inputs.

At 640×480 pixels (VGA), shown in Figure 7c, YOLOv3 maintained top performance with 88.9% confidence and a 100% success rate. MobileNet-SSD followed with 65.87% confidence and 86.67% success, while EfficientDet achieved 54.77% confidence and 80% success. These results confirm that YOLOv3 consistently delivers high detection accuracy across all resolutions, making it a strong candidate for applications that prioritize precision. MobileNet-SSD demonstrates reliable performance at medium and high resolutions, offering a good balance between accuracy and computational efficiency. In contrast, EfficientDet, while resource efficient, shows limited detection capabilities, particularly at lower resolutions. Model selection for edge compu-ting systems must therefore consider this trade-off: YOLOv3 for accuracy, EfficientDet for resource efficiency, and MobileNet-SSD as a middle ground. For detailed metrics, refer to Table 1.

Table 1: Summary of detection confidence metrics across models and resolutions.

Metric (Resolution)	YOLOv3 (%)	MobileNet-SSD (%)	EfficientDet (%)
Average (160 × 120)	50.42	20.07	6.52
Average (320 × 240)	81.77	44.04	28.2
Average (640 × 480)	88.9	65.82	54.77
Min (160 × 120)	00.00	00.00	00.00
Min (320 × 240)	00.00	00.00	00.00
Min (640 × 480)	64.62	00.00	00.00
Max (160 × 120)	93.46	82.92	70.7
Max (320 × 240)	99.92	98.56	85.16
Max (640 × 480)	99.87	99.73	83.98
Detection Success (160 × 120)	63.33	33.33	10.00
Detection Success (320 × 240)	93.33	56.67	43.33
Detection Success (640 × 480)	100.00	86.67	80.00

Source: Authors, (2025).

III.2 CPU USAGE

CPU Usage measures processor workload during inference, indicating computing power used in real-time detection. High usage risks delays, especially on edge computing device. Low usage improves efficiency and stability. Analyzing CPU usage is vital for model feasibility. In cloud systems, it ensures service quality [23].

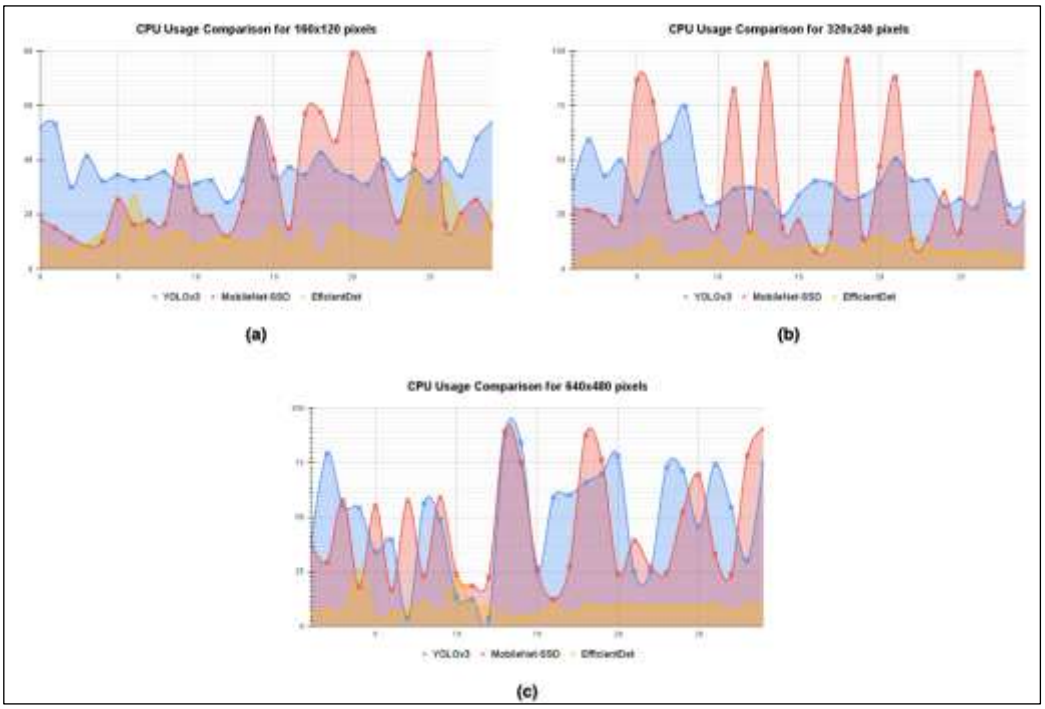

Figure 8: Comparison of CPU usage values for each ML model: (a) 160×120 pixels (QQVGA). (b) 320×240 pixels (QVGA). (c) 640×480 pixels (VGA).
Source: Authors, (2025).

Figure 8a shows the average CPU usage at 160×120 resolution. YOLOv3 recorded the highest usage (37.28%), followed by MobileNet-SSD (31.04%) and EfficientDet (13.23%). These results reflect the architectural complexity of each model. Despite the low resolution, YOLOv3 demands significant resources due to its multiscale detection and modules such as the Feature Pyramid Network and Non Maximum Suppression. MobileNet-SSD also incurs a moderate load from its multi-scale feature extraction, while EfficientDet remains the most efficient thanks to its lightweight structure and adaptive BiFPN-based feature fusion. At 320×240 resolution Figure 8b, YOLOv3 and MobileNet-SSD show increased CPU usage—40.46% and 39.43%, respectively—while EfficientDet remains steady at 13.23%. The rise in resource consumption for YOLOv3 and MobileNet - SSD is attributed to their deeper architectures and larger parameter sets, which increase the per-frame computational load. EfficientDet's consistently low CPU usage highlights its efficiency, largely due to EfficientNet and BiFPN. However, this efficiency may come at the cost of reduced detection accuracy, which must be considered in application design.

At 640×480 resolution Figure c, YOLOv3 peaks at 50.7% CPU usage, MobileNet-SSD at 43.58%, and EfficientDet drops further to 8.47%. Higher resolutions amplify computational demands, especially for YOLOv3, which employs complex processing to maintain detection precision. MobileNet-SSD also shows increased load but remains more efficient than YOLOv3. EfficientDet consistently consumes minimal CPU, though with potential trade-offs in accuracy. Overall, the results indicate that more accurate and complex models require significantly greater computational resources. YOLOv3 offers strong detection performance but at a high CPU cost, making it suitable for more powerful systems. EfficientDet delivers excellent efficiency with moderate accuracy, making it ideal for resource-constrained environments. MobileNet-SSD strikes a balance between detection quality and CPU load. Detailed CPU comparisons are provided in Table 2.

Table 2: Summary of CPU usage metrics across models and resolutions.

Metric (Resolution)	YOLOv3 (%)	MobileNet-SSD (%)	EfficientDet (%)
Average (160×120)	37.28	31.04	13.23
Average (320×240)	40.46	39.43	13.23
Average (640×480)	50.7	43.58	8.47
Min (160×120)	24.4	8.5	3.4
Min (320×240)	24.2	8.5	3.4
Min (640×480)	3.5	12.03	3.5
Max (160×120)	55.1	79.1	37.8
Max (320×240)	74.7	96.03	16.00
Max (640×480)	87.9	90.04	80.00

Source: Authors, (2025).

III.3 FRAME PER SECOND (FPS)

FPS indicates the number of frames a model processes per second, reflecting both inference speed and system efficiency. Object detection performance is commonly evaluated using FPS and FLOPs [24]. A higher FPS enhances video quality, and to achieve smooth playback at 10 FPS, processing must be completed within 100 ms [25].

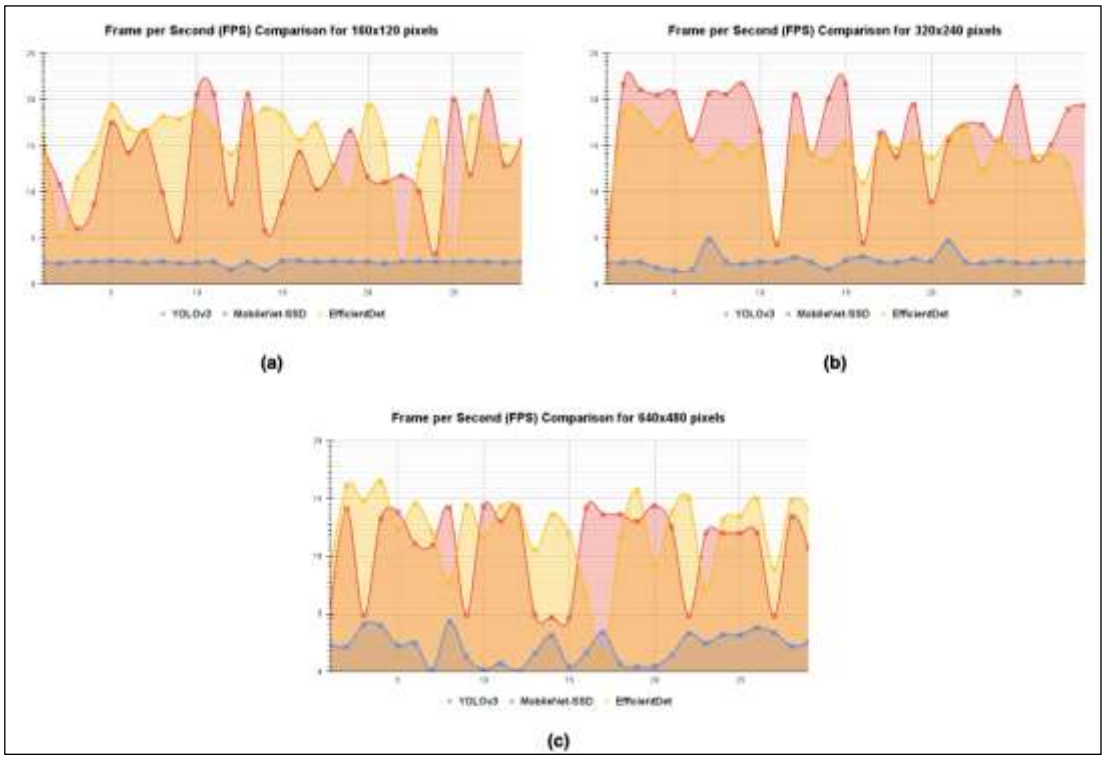

Figure 9: Comparison of FPS values for each ML model: (a) 160×120 pixels (QQVGA). (b) 320×240 pixels (QVGA). (c) 640×480 pixels (VGA).
Source: Authors, (2025).

Figure 9a compares the Frames Per Second (FPS) performance of YOLOv3, MobileNet-SSD, and EfficientDet at a resolution of 160×120. EfficientDet achieved the highest FPS (14.96), followed by MobileNet-SSD (12.98), while YOLOv3 lagged significantly (2.37). These results highlight EfficientDet's superior efficiency on low-resolution inputs due to its lightweight architecture and feature fusion mechanism. Although MobileNet-SSD is also efficient, its performance is slightly lower, likely due to its mul-ti-scale layer complexity. In contrast, YOLOv3's low FPS suggests that it is unsuitable for real-time applications on resource constrained devices, despite its high accuracy.

At 320×240 resolution (Figure 9b), MobileNet-SSD recorded the best performance (16.07 FPS), followed by EfficientDet (13.84 FPS) and YOLOv3 (2.52 FPS). These results confirm MobileNet-SSD's advantage in real-time processing, particularly for mid-resolution scenarios. While EfficientDet remains competitive, YOLOv3's relatively low FPS again underscores its heavier computational load, limiting its applicability in time critical edge computing systems.

At 640×480 resolution (Figure 9c), MobileNet-SSD maintained the lead with 12.27 FPS, followed by EfficientDet (10.54 FPS) and YOLOv3 (2.11 FPS). The overall decrease in FPS across all models re-flects the increased processing demands at higher resolutions. YOLOv3 consistently delivered the lowest FPS, though it may still be applicable in scenarios where real-time video output is not required. Overall, MobileNet-SSD and EfficientDet demonstrate better scalability in edge computing deployments. Detailed performance metrics are presented in Table 3.

Table 3: Summary of FPS metrics across models and resolutions.

Metric (Resolution)	YOLOv3 (FPS)	MobileNet-SSD (FPS)	EfficientDet (FPS)
Average (160×120)	2.37	12.98	14.96
Average (320×240)	2.52	16.07	13.84
Average (640×480)	2.11	12.27	10.54
Min (160×120)	1.54	3.31	1.52
Min (320×240)	1.51	2.06	4.67
Min (640×480)	0.01	1.27	4.67
Max (160×120)	2.56	21.02	19.45
Max (320×240)	4.87	21.71	18.60
Max (640×480)	4.34	16.5	14.40

Source: Authors, (2025).

III.4 LATENCY

Latency plays a crucial role in evaluating the performance of inference models, as it measures the time required to generate output from input. In the context of human detection, accuracy alone is insufficient; models must also operate in a timely manner. Deep learning models often face stringent latency constraints [26]. Lower latency significantly enhances the feasibility of deploying models in real-time systems, such as video streaming, which demands instantaneous responses, or edge computing, where processing must occur near the data source without significant delays. Therefore, latency optimiza-tion is a key factor in the design and selection of models for applications that require high speed and responsiveness.

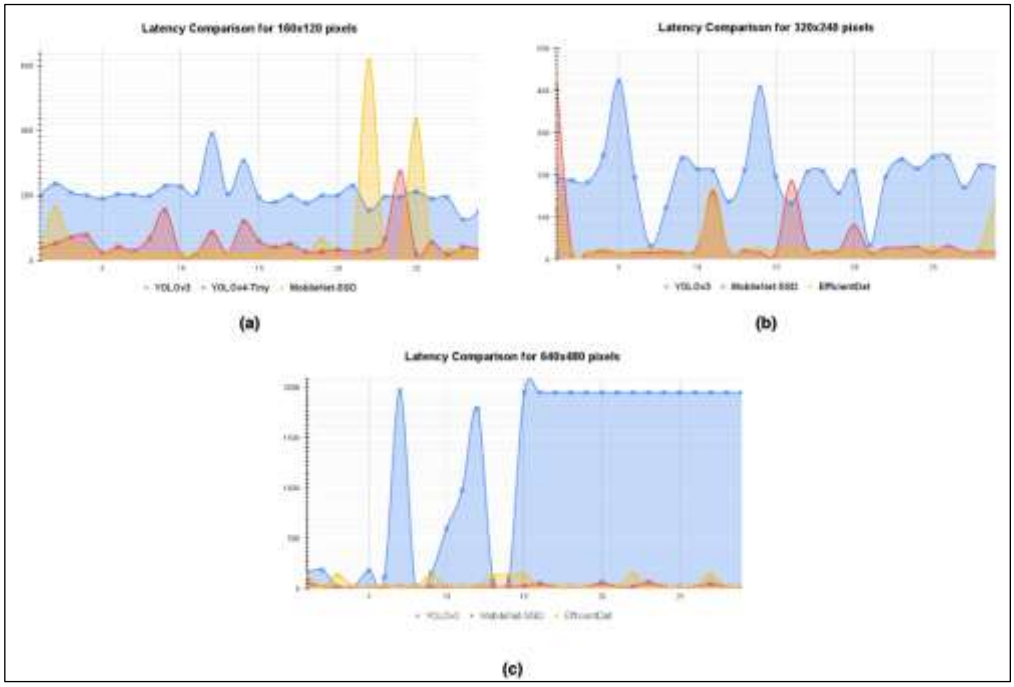

Figure 10: Comparison of latency values for each ML model: (a) 160×120 pixels (QQVGA). (b) 320×240 pixels (QVGA). (c) 640×480 pixels (VGA).
Source: Authors, (2025).

Figure 10a presents the latency results at 160×120 resolution, defined as the average time required to generate a detection output from a single image. MobileNet-SSD achieves the best performance with a latency of 55.71 ms, enabling efficient near real-time processing. EfficientDet follows with 63.01 ms, which remains suitable for responsive applications. In contrast, YOLOv3 records the highest latency at 207.85 ms, reflecting its heavier computational load. These results indicate that for low-resolution tasks requiring fast responses, MobileNet-SSD and EfficientDet are more suitable than YOLOv3.

At 320×240 resolution (Figure 10b), EfficientDet leads with the lowest latency of 35.46 ms an improvement compared to its performance at lower resolution—demonstrating strong scalability. MobileNet-SSD follows with 58.42 ms, maintaining consistent results. YOLOv3 again shows the highest latency at 202.24 ms, reinforcing its limitations for time-critical tasks. This resolution-dependent shift emphasizes the importance of evaluating detection models across different input sizes for optimal de-ployment.

At 640×480 resolution (Figure 10c), MobileNet-SSD achieves the lowest latency at 24.86 ms, followed by EfficientDet at 53.35 ms. YOLOv3, however, shows a dramatic increase to 1194.34 ms far beyond real-time thresholds. This trend confirms that although YOLOv3 may provide high accuracy, it is not latency-efficient, particularly at higher resolutions.

Overall, MobileNet-SSD performs best at low and high resolutions, while EfficientDet excels at medium resolution. Both models are better suited for real-time edge computing applications that demand fast response and energy efficiency. Detailed latency comparisons are presented in Table 4.

Table 4: Summary of latency metrics across models and resolutions.

Metric (Resolution)	YOLOv3 (ms)	MobileNet-SSD (ms)	EfficientDet (ms)
Average (160×120)	207.85	55.71	63.01
Average (320×240)	202.24	58.42	35.46
Average (640×480)	1194.34	25.86	53.35
Min (160×120)	126.31	16.77	15.96
Min (320×240)	33.21	16.07	16.19
Min (640×480)	17.22	16.42	15.96
Max (160×120)	390.13	272.35	615.36
Max (320×240)	424.4	438.1	170.43
Max (640×480)	1972.02	73.01	145.21

Source: Authors, (2025).

V. CONCLUSIONS

This study successfully applied and compared three machine learning models—YOLOv3, MobileNet-SSD, and EfficientDet—for human detection using ESP32-CAM devices with edge computing, based on Ubuntu ARM running on virtual machines. Testing was conducted at three resolutions: 160×120 pixels (QQVGA), 320×240 pixels (QVGA), and 640×480 pixels (VGA), under controlled lighting conditions (40–150 lux). The results show that YOLOv3 is superior in accuracy and detection consistency but has weaknesses in processing efficiency, with high CPU usage, latency, and low FPS. MobileNet-SSD offers a balance between accuracy and efficiency, making it an ideal choice for real-time systems with limited resources. EfficientDet is the most lightweight in terms of computation, suitable for edge computing de-vices with limited power, although at the cost of lower accuracy. This study recommends selecting mod-els based on system requirements: YOLOv3 for high accuracy, MobileNet-SSD for balanced performance, and EfficientDet for resource-constrained systems. Further research is suggested to evaluate these models on specialized edge computing devices such as NVIDIA Jetson, as well as to explore custom-trained lightweight models and assess their performance on edge computing such as NVIDIA Jetson or Coral TPU.

VIII. REFERENCES

[1] Government of Indonesia, "Presidential Regulation Number 82 of 2023 on the Acceleration of Digital Transformation and Integration of National Digital Services," 2023. [Online]. Available: https://peraturan.bpk.go.id/Details/279013/perpres-no-82-tahun-2023

[2] P. M. Jeyanthi, D. H. Polay, and T. Choudhury, "The Rise of Decision Intelligence: AI That Optimizes Decision-Making," in Decision Intelligence Analytics and the Implementation of Strategic Business Management, P. M. Jeyanthi, T. Choudhury, D. Hack-Polay, T. P. Singh, and S. Abujar, Eds. Cham, Switzerland: Springer, 2022. doi: 10.1007/978-3-030-82763-2_7.

[3] A. Rajkomar, J. Dean, and I. Kohane, "Machine learning in medicine," New England Journal of Medicine, vol. 380, no. 14, pp. 1347–1358, 2019. doi: 10.1056/NEJMra1814259.

[4] D. B. Olawade, O. Z. Wada, A. O. Ige, B. I. Egbewole, A. Olojo, and B. I. Oladapo, "Artificial intelligence in environmental monitoring: Advancements, challenges, and future directions," Hygiene and Environmental Health Advances, vol. 12, p. 100114, 2024. doi: 10.1016/j.heha.2024.100114.

[5] B. Bhavya Sree, V. Yashwanth Bharadwaj, and N. Neelima, "An inter-comparative survey on state-of-the-art detectors—R-CNN, YOLO, and SSD," in Intelligent Manufacturing and Energy Sustainability, A. Reddy, D. Marla, M. N. Favorskaya, and S. C. Satapathy, Eds., vol. 213, Smart Innovation, Systems and Technologies. Singapore: Springer, 2021, pp. 555–567. doi: 10.1007/978-981-33-4443-3_46.

[6] P. Dandekar, S. S. Aote, and A. Raipurkar, "Low-resolution face recognition: Review, challenges and research directions," Computers and Electrical Engineering, vol. 120, p. 109846, 2024. doi: 10.1016/j.compeleceng.2024.109846.

[7] D. Heller, M. Rizk, R. Douguet, A. Baghdadi, and J.-P. Diguet, "Marine objects detection using deep learning on embedded edge devices," in Proc. 2022 IEEE International Workshop on Rapid System Prototyping (RSP), 2022, pp. 1–7. doi: 10.1109/RSP57251.2022.10039025.

[8] R. Sunkara and T. Luo, "No more strided convolutions or pooling: A new CNN building block for low-resolution images and small objects," in Lecture Notes in Computer Science, vol. 13715. Springer, 2023. doi: 10.1007/978-3-031-26409-2_27.

[9] J. Diez-Tomillo, I. Martinez-Alpiste, G. Golcarenarenji, A. Muñoz, D. Moreno, and R. Alvarez, "Efficient CNN-based low-resolution facial detection from UAVs," Neural Computing and Applications, vol. 36, pp. 5847–5860, 2024. doi: 10.1007/s00521-023-09401-3.

[10] C. Xie, D. J. Pagliari, and A. Calimera, "Energy-efficient and privacy-aware social distance monitoring with low-resolution infrared sensors and adaptive inference," in Proc. 2022 17th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME), 2022, pp. 181–184. doi: 10.1109/PRIME55000.2022.9816801.

[11] P. Mittal, "A comprehensive survey of deep learning-based lightweight object detection models for edge devices," Artificial Intelligence Review, vol. 57, p. 242, 2024. doi: 10.1007/s10462-024-10877-1.

[12] C. Avasalcai and S. Dustdar, "Edge computing: Use cases and research challenges," in Edge Computing: Use Cases and Research Challenges, B. Vogel-Heuser and M. Wimmer, Eds., Springer, 2023, pp. 125–142. doi: 10.1007/978-3-662-65004-2_5.

[13] L. Xu, Y. Zhao, Y. Zhai, W. Zhang, and N. Liu, "Small object detection in UAV images based on YOLOv8n," International Journal of Computational Intelligence Systems, vol. 17, p. 223, 2024. doi: 10.1007/s44196-024-00632-3.

[14] M. Desai, H. Mewada, I. M. Pires, and S. Roy, "Evaluating the performance of the YOLO object detection framework on COCO dataset and real-world scenarios," Procedia Computer Science, vol. 251, pp. 157–163, 2024. doi: 10.1016/j.procs.2024.11.096.

[15] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in Computer Vision – ECCV 2014, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., Lecture Notes in Computer Science, vol. 8693, Cham: Springer, 2014, pp. 740–755. doi: 10.1007/978-3-319-10602-1_48.

[16] Y. Lin, N. Ding, Z. Liu, and M. Sun, "Pre-trained models for representation learning," in Pre-trained Models for Representation Learning, Z. Liu, Y. Lin, and M. Sun, Eds. Cham: Springer, 2023, pp. 127–167. doi: 10.1007/978-981-99-1600-9_5.

[17] A. Vijayakumar and S. Vairavasundaram, "YOLO-based object detection models: A review and its applications," Multimedia Tools and Applications, vol. 83, pp. 83535–83574, 2024. doi: 10.1007/s11042-024-18872-y.

[18] Y. Zhou, H. Qian, and P. Ding, "Lite-YOLOv3: A real-time object detector based on multi-scale slice depthwise convolution and lightweight attention mechanism," Journal of Real-Time Image Processing, vol. 20, p. 123, 2023. doi: 10.1007/s11554-023-01379-4.

[19] Y. Zhou, H. Qian, and P. Ding, "Lite-YOLOv3: A real-time object detector based on multi-scale slice depthwise convolution and lightweight attention mechanism," Journal of Real-Time Image Processing*, vol. 20, p. 123, 2023. doi: 10.1007/s11554-023-01379-4.

[20] M. A. B. Abbass and Y. Ban, "MobileNet-based architecture for distracted human driver detection of autonomous cars," Electronics, vol. 13, no. 2, 2024. doi: 10.3390/electronics13020365.

[21] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and efficient object detection," in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 2020, pp. 10778–10787. doi: 10.1109/CVPR42600.2020.01079.

[22] L. Peng, H. Wang, and J. Li, "Uncertainty evaluation of object detection algorithms for autonomous vehicles," Automotive Innovation, vol. 4, no. 3, pp. 241–252, 2021. doi: 10.1007/s42154-021-00154-0.

[23] R. Estrada, I. Valeriano, and X. Aizaga, "CPU usage prediction model: A simplified VM clustering approach," in Complex, Intelligent and Software Intensive Systems, L. Barolli, Ed. Cham: Springer, 2023, pp. 210–221. doi: 10.1007/978-3-031-35734-3_21.

[24] W. Chen, J. Luo, F. Zhang, Y. Liu, C. Li, and J. Chen, "A review of object detection: Datasets, performance evaluation, architecture, applications and current trends," Multimedia Tools and Applications, vol. 83, pp. 65603–65661, 2024. doi: 10.1007/s11042-023-17949-4.

[25] D. Xu, H. Tao, J. Yu, and C. Xiao, "Real-time multi-camera video stitching based on improved optimal stitch line and multi-resolution fusion," in Image and Graphics (ICIG 2017), Y. Zhao, X. Kong, and D. Taubman, Eds., Lecture Notes in Computer Science, vol. 10668. Cham: Springer, 2017. doi: 10.1007/978-3-319-71598-8_12.

[26] H. She, Y. Luo, Z. Xiang, W. Liang, and Y. Xie, "Accurate latency prediction of deep learning model inference under dynamic runtime resource," in Neural Information Processing, B. Luo, L. Cheng, Z.-G. Wu, H. Li, and C. Li, Eds. Cham: Springer, 2024, pp. 495–510. doi: 10.1007/978-981-99-8126-7_39.