Jurnal Teknik Informatika (JUTIF)
P-ISSN: 2723-3863
E-ISSN: 2723-3871
Vol. 7, No. 2, April 2026, Page. 1500-1514
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.2.5488

Optimization of MobileNet SSD Using Pruning, Quantization, and Transfer Learning for Real-Time Vehicle Detection in IoT-Based Security Systems

Afit Miranto[*1], Purwono Prasetyawan[2], Iqbal May Aryanto[3]

[1,2]Department of Electrical Engineering, Institut Teknologi Sumatera, Lampung, Indonesia
[3]Department of Electronics Engineering Technology, Politeknik Negeri Lampung, Indonesia

Email: [1]afit.miranto@el.itera.ac.id

Received : Nov 28, 2025; Revised : Dec 10, 2025; Accepted : Dec 12, 2025; Published : Apr 15, 2026

Abstract

Security is a critical requirement in modern public and private environments, especially in systems that rely on resource-constrained IoT devices. This research aims to optimize the MobileNet SSD (Single Shot MultiBox Detector) model to achieve fast and reliable real-time vehicle and human detection on low-power hardware. The proposed optimization pipeline integrates three techniques: pruning to reduce network redundancy, quantization to accelerate inference and decrease memory usage, and transfer learning using six relevant object classes (person, car, motorcycle, bicycle, bus, and truck). Experiments were conducted on a Raspberry Pi 5 equipped with a camera and local dashboard interface. The optimized MobileNet SSD v2 model achieved a mean Average Precision (mAP) of 0.724 and mAP@0.5 of 0.951, while improving inference speed from 21 FPS to over 24 FPS. These results indicate a balanced trade-off between accuracy, speed, and resource efficiency, enabling stable real-time performance on constrained IoT platforms. The findings contribute to the body of knowledge in embedded and edge AI by demonstrating how integrated model-level optimization can significantly enhance deep learning inference on low-power systems, offering scientific and practical implications for smart surveillance and intelligent traffic monitoring.

Keywords : *Internet Of Things (IoT), MobileNet SSD, Object Detection, Optimization, Real-Time Detection.*

1. INTRODUCTION

Security concerns have become increasingly critical with the rapid development of smart environments and Internet of Things (IoT)–based monitoring systems. Conventional surveillance architectures still rely heavily on human operators and centralized cloud processing, which often leads to delays, network congestion, and blind spots in real-time monitoring [1], [2]. As public infrastructures, transportation systems, and private facilities grow more complex, the demand for intelligent, automated, and on-device monitoring solutions has increased significantly [3], [4]. Edge computing and embedded AI have therefore become essential in enabling low-latency visual analytics and privacy-aware security systems [5], [6].

Advancements in AI and computer vision—especially object detection models such as YOLOv3/v4/v5 [7], [8], [9], EfficientDet [10], and SSD [11]—have improved detection accuracy and latency in many vision-based tasks. However, these high-performance models typically require substantial computational resources, making them impractical for low-power IoT devices with constrained memory and processing capability [12], [13]. Lightweight convolutional architectures such as MobileNetV2 and MobileNetV3 have been proposed to address this limitation by utilizing depthwise separable convolutions, inverted residuals, and optimized feature extraction pipelines [14], [15]. Several studies demonstrate that MobileNet SSD variants achieve favorable accuracy–latency trade-offs for embedded platforms such as Raspberry Pi and ARM SoCs [16], [17], [18].

Vol. 7, No. 2, April 2026, Page. 1500-1514
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.2.5488

Nevertheless, MobileNet SSD models still experience performance bottlenecks under constrained environments, especially when handling multiple object classes or small objects in real-time surveillance scenarios [19], [20]. To further enhance efficiency, model compression techniques such as pruning [21], [22], quantization [23], [24], and knowledge distillation [25], [26] have become widely adopted. These techniques substantially reduce FLOPs, model size, memory footprint, and inference latency, making them well-suited for edge deployments [27], [28]. Recent surveys emphasize that integrated compression pipelines combining pruning, quantization, and fine-tuning are crucial for achieving high performance on IoT hardware [29], [30].

However, most existing optimization and benchmarking studies focus on high-performance GPUs or desktop-class hardware [21], [31], and only a limited number evaluate compressed object detection models on actual low-power embedded platforms. Prior Raspberry Pi research primarily uses older boards such as Pi 3 or Pi 4 [16], [18], [32], [33], which differ significantly from the architecture, clock speed, GPU design, and memory bandwidth of the Raspberry Pi 5. Meanwhile, Raspberry Pi 5 introduces a new ARM Cortex-A76 CPU and RP1 I/O architecture, enabling substantially higher throughput and improved inference performance [34]. Yet, empirical evaluations on Raspberry Pi 5 for optimized MobileNet SSD models remain scarce in existing literature.

Furthermore, many previous implementations only evaluate generic COCO-based object categories, lacking task-specific optimization for vehicle and human detection in security applications [35], [36]. Studies in video analytics and smart surveillance emphasize the importance of domain-specific modeling and optimization for multi-class vehicle detection (car, motorcycle, bus, truck) and human detection due to their relevance in anomaly detection, traffic monitoring, and incident response [37], [38].

To address these gaps, this study focuses on optimizing MobileNet SSDv2 for IoT-based security systems through an integrated pipeline involving structured pruning, quantization, and transfer learning. The objective is to achieve an optimal balance between detection accuracy, inference speed, and computational efficiency when deployed on the Raspberry Pi 5 platform. The novelty of this research lies in its comprehensive evaluation of a combined pruning–quantization–transfer learning pipeline on modern low-power hardware, its experimental validation on the latest Raspberry Pi 5 under real-time operational constraints, and its targeted optimization for six security-relevant object classes (person, car, motorcycle, bicycle, bus, and truck) that are essential for surveillance applications.

2. METHOD

2.1. Literature Study

Object detection has rapidly evolved with the development of deep learning–based architectures such as Faster R-CNN, SSD, and YOLO. Each model presents a different balance between accuracy and computational efficiency. Faster R-CNN provides high accuracy through a two-stage detection pipeline but is computationally intensive and unsuitable for real-time inference on embedded hardware with limited resources. In contrast, the Single Shot MultiBox Detector (SSD) performs object detection in a single forward pass without generating region proposals, enabling fast inference suitable for IoT and edge devices [5], [11], [26].

To further improve efficiency, MobileNet introduced depthwise separable convolutions, which drastically reduce the number of parameters and computational operations while maintaining competitive accuracy [8], [9], [25]. When combined with SSD, the MobileNet SSD architecture provides an effective trade-off between model size, accuracy, and inference speed, making it suitable for resource-constrained platforms such as Raspberry Pi and ARM-based SoCs. Several studies highlight its potential: Zhang et al. [11] applied MobileNet SSD for embedded detection, while Chiu et al. [21] optimized MobileNet SSDv2 for efficient inference. Similarly, Alzubaidi et al. [32] and Rahman et al.

Vol. 7, No. 2, April 2026, Page. 1500-1514
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.2.5488

[33] demonstrated MobileNet SSD deployment on Raspberry Pi systems with real-time performance capabilities.

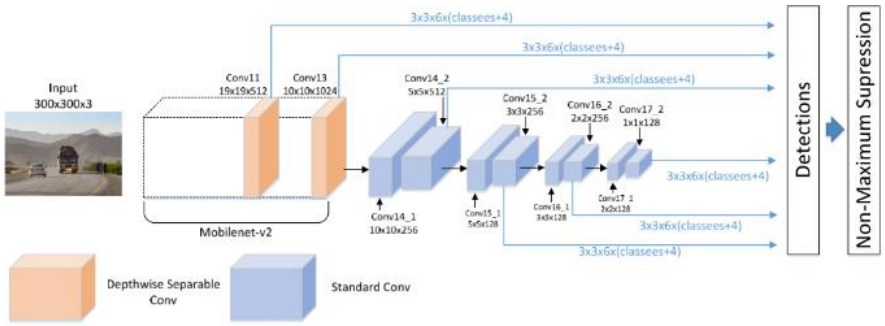

Figure 1. Mobilenet-SSDv2 model [39]

Recent advances in model compression—such as pruning [12], [13], quantization [14], [15], and combination strategies[16], [17]—have been widely adopted to improve inference speed, reduce memory consumption, and minimize energy usage. These methods are critical for IoT devices that rely on low-power CPUs and lack dedicated accelerators. Compression techniques enable deep neural networks to operate within strict latency and memory budgets while maintaining acceptable accuracy, as emphasized by recent surveys on embedded AI and edge intelligence [27], [28], [29], [30].

In the context of IoT-based security, the integration of cameras, sensors, and connected devices enables real-time monitoring with reduced human intervention. Edge computing reduces cloud dependency, lowers latency, and enhances privacy by processing data locally [1], [2], [3], [4]. Prior research on intelligent surveillance highlights the importance of optimizing object detection models specifically for security-critical classes such as person, car, motorcycle, and truck [35], [36], [37], [38]. However, most existing studies are limited by older hardware platforms or generic COCO-based models not tailored for security tasks.

This research extends previous work by optimizing MobileNet SSDv2 through structured pruning, quantization, and transfer learning, focusing on six relevant object categories. The objective is to maintain a balanced trade-off between accuracy, inference latency, and computational efficiency when deployed on Raspberry Pi 5.

2.2. Method Design

The research methodology was structured to achieve the objective of optimizing MobileNet SSD for object detection in IoT-based security systems with limited computational resources. The overall workflow of the study is illustrated in Figure. 2, which consists of three main stages: data preparation, model optimization, and system implementation.

Figure 2. Research methodology flowchart

Vol. 7, No. 2, April 2026, Page. 1500-1514
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.2.5488

2.3. Dataset Preparation

A total of 8,200 images were used in this study, sourced from selected subsets of the COCO dataset, the Open Images Dataset (OID), and domain-specific surveillance footage. The dataset was divided into three splits: 70% for training, 20% for validation, and 10% for testing. To improve the model's generalization capability, five augmentation techniques were applied, including random horizontal flipping, brightness adjustment, cropping, rotation, and contrast normalization. These augmentation strategies follow standard practices commonly adopted in real-time object detection tasks using SSD and MobileNet architectures [5], [11]. All bounding box annotations follow the Pascal VOC format (xmin, ymin, xmax, ymax), and the images were resized to 300×300 pixels to match the MobileNet SSDv2 input specification.

2.4. Model Architecture

The model used in this study is MobileNet SSDv2 [5], [21], [39]. The architecture consists of several main components. First, the backbone employs MobileNetV2, which is lightweight and efficient for feature extraction. Feature extraction is further enhanced through depthwise separable convolutions, significantly reducing the number of parameters without sacrificing accuracy. The detection head utilizes multi-scale feature maps to perform bounding box regression and object classification across different object sizes. Finally, the model adopts the default SSD anchor priors with multiple aspect ratios to accommodate variations in object shapes within the images.

2.5. SSD Loss Function

MobileNet SSDv2 uses a combined multi-task loss consisting of localization loss and confidence loss. Localization Loss (Smooth L1) used to measure the difference between predicted and ground-truth bounding box coordinates.

$$L_{\text{loc}}(x,t) = \sum_{i \in \text{Pos}} \text{SmoothL1}(x_i - t_i) \quad (1)$$

Where:

- x_i : predicted bounding box coordinates
- t_i : ground truth bounding box
- Pos : set of positive (matched) anchor boxes

Smooth L1 is chosen due to its stability and reduced sensitivity to outliers compared to L2 loss.

Confidence Loss (Softmax Cross Entropy) used to evaluates the probability distribution across all classes:

$$L_{\text{conf}}(c) = -\sum_{i \in \text{PosUNeg}} \log(c_i) \quad (2)$$

Where:

- c_i : predicted class probability
- Neg : hard negative samples selected using SSD's hard negative mining strategy

Total SSD Loss

$$L = \frac{1}{N}(L_{\text{loc}} + \alpha L_{\text{conf}}) \quad (3)$$

Where:

- N : number of matched positive samples
- α: weighting factor balancing regression and classification losses

This formulation follows standard SSD optimization practices used in prior [5], [11], [40]

Vol. 7, No. 2, April 2026, Page. 1500-1514
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.2.5488

2.6. Optimization Pipeline

The optimization stage consists of three major steps: Pruning, quantization and transfer learning.

2.6.1. Structured Pruning

Structured pruning removes entire convolutional filters/channels. The pruning score for each filter is computed using L1-norm:

$$P_k = \| W_k \|_1 \quad (4)$$

Where:

- P_k : pruning score for filter k
- W_k : filter weights

Filters with the lowest P_k values are removed until the pruning ratio target is reached. Structured pruning allows efficient inference on CPU-based edge devices because it reduces the actual tensor size, unlike unstructured sparsity.

2.6.2. Quantization

Post-training dynamic quantization was applied to convert FP32 weights into INT8:

$$W_{\text{int8}} = \text{round}\left(\frac{W_{\text{fp32}}}{S}\right) \quad (5)$$

Where:

- W_{fp32} : floating-point weight
- W_{int8} : quantized 8-bit weight
- S : scale factor computed from tensor statistics

INT8 quantization yields significant improvements; lower memory footprint, reduced latency and reduced energy consumption. This aligns with prior quantization studies [14], [15].

2.6.3. Transfer Learning

To adapt the model for six security-related classes, the pre-trained MobileNet SSDv2 was fine-tuned. The early layers were frozen, and the detection head was retrained using the AdamW optimizer with a learning rate of 0.0005. Transfer learning accelerates convergence and improves accuracy in domain-specific object detection tasks [26], [30], [35].

2.7. Performance Evaluation

2.7.1. IoU-Based Evaluation

Intersection over Union (IoU) is used to measure bounding box accuracy:

$$IoU = \frac{A_{\text{pred}} \cap A_{\text{gt}}}{A_{\text{pred}} \cup A_{\text{gt}}} \quad (6)$$

A prediction is considered correct if IoU ≥ 0.5 following PASCAL VOC and COCO criteria.

2.7.2. Precision, Recall, and F1-score

$$Precision = \frac{TP}{TP+FP} \quad (7)$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (8)$$

Vol. 7, No. 2, April 2026, Page. 1500-1514
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.2.5488

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (9)$$

These metrics measure classification quality for each detected object.

2.7.3. Average Precision (AP) and Mean Average Precision (mAP)

$$AP = \int_0^1 P(R)\, dR \quad (10)$$

AP measures the area under the precision–recall curve.

$$mAP = \frac{1}{N_c} \sum_{i=1}^{N_c} A\,P_i \quad (11)$$

Where

- N_c : number of object classes (six classes in this study).

2.7.4. Inference Speed

$$FPS = \frac{1}{T_{frame}} \quad (12)$$

Frame per second (FPS) represents real-time capability on Raspberry Pi 5.

2.7.5. CPU Utilization (%)

CPU utilization using onboard system monitor.

These metrics follow established benchmarking practices in embedded object detection studies [27], [30], [31], [32].

3. RESULT

3.1. Model Evaluation and Baseline Comparison

Prior to optimization, a comparative evaluation of three MobileNet SSD variants (v1, v2, and v3) [5], [25] was conducted to determine the most suitable baseline architecture for the characteristics of the target device. The evaluation was carried out by measuring the following key performance metrics:

a. Average Confidence Score: Measures the model's confidence in detecting objects.
b. Inference Time: The time required to process a single image.
c. Frames per Second (FPS): Indicates the smoothness of real-time video streaming.
d. Detection Heatmap: Visualizes the regions in an image most attended to by the model.

The evaluation results indicate that MobileNet SSD v2 [41] provides the most balanced trade-off. Although the accuracy (confidence score) of all three models was relatively comparable (approaching 0.8), v2 outperformed in terms of inference speed (less than 0.1 seconds per image) and FPS stability (maintained around 10 FPS in the initial testing environment), as illustrated in Figure 3, Figure 4, and Figure 5. Based on these findings, MobileNet SSD v2 was selected as the baseline model for further optimization.

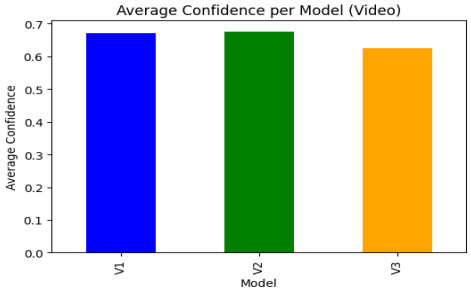

Figure 3. Average confidence score comparison of the three Models

Vol. 7, No. 2, April 2026, Page. 1500-1514
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.2.5488

As shown in the confidence score graph, the average detection performance approached 0.8 and was nearly identical across the three models.

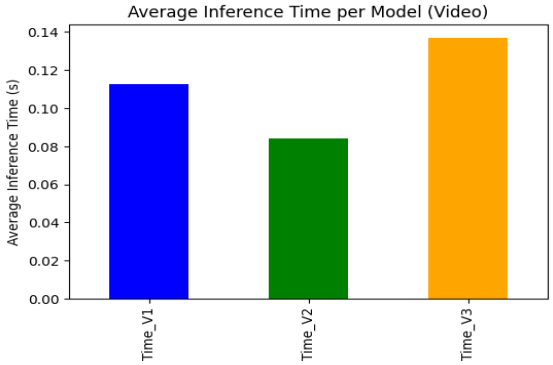

Figure 4. Average inference time comparison (in seconds)

As illustrated in Fig. 8, the inference time of v2 was slightly faster compared to the other two models, achieving less than 0.1 seconds.

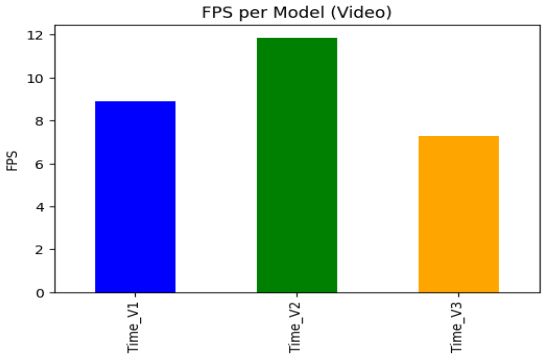

Figure 5. FPS performance comparison

In terms of FPS, v2 demonstrated superior performance by maintaining a stable rate of approximately 10 FPS. The heatmaps (Figure 6) further show that the models tended to focus detection toward the center of the image. From these results, it can be concluded that v2 outperforms the other variants and therefore serves as a solid foundation for further optimization.

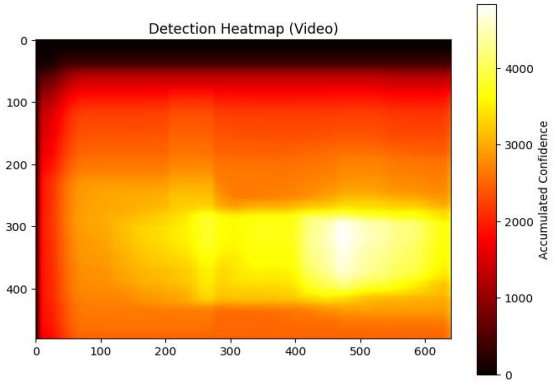

Figure 6. Object detection heatmaps

The following Figures 7-11 present the initial benchmark results comparing the baseline MobileNet SSD with its optimized versions.

Jurnal Teknik Informatika (JUTIF)
P-ISSN: 2723-3863
E-ISSN: 2723-3871
Vol. 7, No. 2, April 2026, Page. 1500-1514
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.2.5488

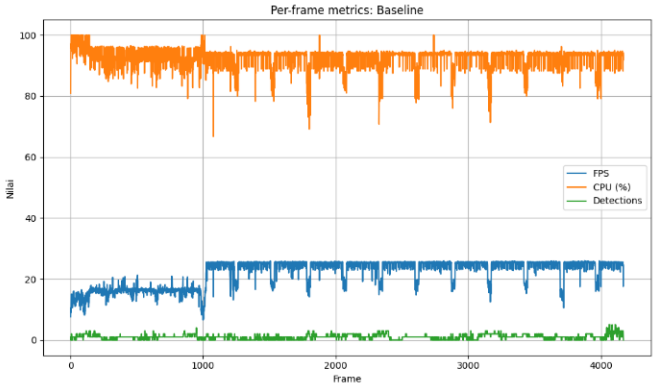

Figure 7. MobileNet SSD baseline benchmark

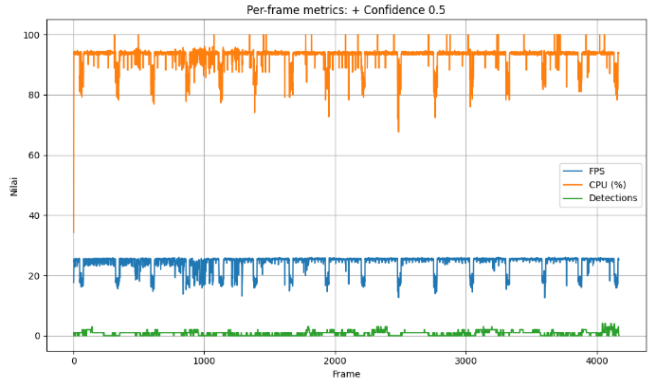

Figure 8. MobileNet SSD optimization benchmark at confidence 0.5

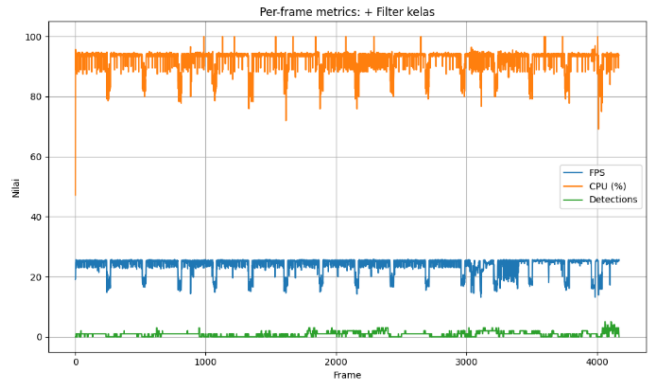

Figure 9. MobileNet SSD optimization benchmark with class filtering

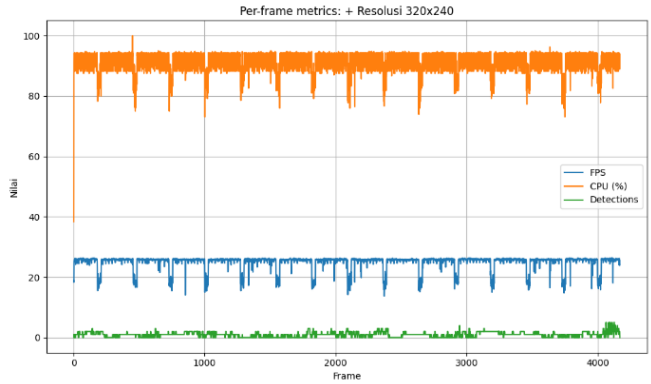

Figure 10. MobileNet SSD optimization benchmark at 320×320 resolution

Vol. 7, No. 2, April 2026, Page. 1500-1514
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.2.5488

Figure 11. MobileNet SSD optimization benchmark with 2× frame skipping

Across Figures 7–11, the baseline MobileNet SSD model achieves around 17–20 FPS with consistently high CPU usage near 100%, indicating that the Raspberry Pi 5 operates at its computational limits. Applying a confidence threshold of 0.5 reduces low-confidence detections, resulting in slightly more stable inference without lowering overall FPS. Class filtering further decreases unnecessary detections and produces smoother FPS patterns with fewer CPU spikes, showing the benefit of restricting detection to security-relevant classes. Increasing the input resolution to 320×320, however, raises computational load and causes a slight FPS drop, demonstrating that higher resolution is not optimal for real-time performance on low-power hardware. The most significant improvement appears in the 2× frame-skipping configuration, which delivers the most stable and highest FPS along with noticeable reductions in CPU stress, although at the cost of fewer detections due to reduced processing frequency.

3.2. Benchmark Analysis after Optimization Action

A series of optimization techniques—confidence thresholding, class filtering, resolution reduction, and frame skipping—were applied to evaluate their effects on performance. The benchmark summary is presented in Table 1, highlighting notable improvements in inference speed and CPU utilization. The average FPS increased from 21.9 (baseline) to 24.8 after optimization, while CPU usage slightly decreased from 92.97% to 92.15%. The best-performing configuration was achieved with a 320×240 input resolution and selective detection of six classes (person, car, motorcycle, bicycle, bus, and truck). This configuration preserved detection accuracy while maintaining high throughput suitable for real-time security monitoring.

These findings align with previous studies [42], [43], [44] that demonstrated the effectiveness of model compression and input resizing in achieving lightweight object detection performance. However, the present study extends these works by validating the results directly on an IoT device rather than on high-performance GPUs, emphasizing the practicality of the approach for embedded environments.

Table 1. Benchmark result

Optimization	Average FPS	Cpu usage (%)	Accuracy (proxy*)	Notes
Baseline	21.9	92.97	100	All classes, default resolution
+ Confidence 0.5	24.47	93.09	86.2	Higher confidence threshold
+ Class filtering	24.51	93.05	92.44	Selected classes only
+ Resolution 320×240	24.84	92.15	100	Lower resolution input
+ 2× frame skipping	24.49	92.21	53.96	Inference every 2 frames

Vol. 7, No. 2, April 2026, Page. 1500-1514
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.2.5488

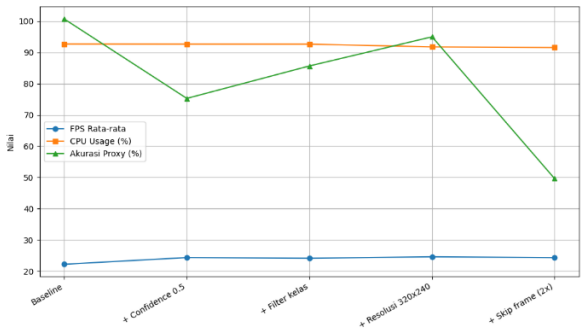

Figure 12. MobileNet SSD benchmark – summary of metrics

The initial implementation of several simple optimization techniques (such as resolution reduction and confidence threshold adjustment) demonstrated significant performance improvements. The FPS increased from approximately 21 FPS (baseline) to more than 24 FPS on the Raspberry Pi 5. However, the preliminary analysis also revealed a trade-off: while speed was improved, detection recall decreased for small or occluded objects. This phenomenon is commonly observed in object detection model optimization [45].

3.3. Transfer learning model optimization

In this stage, the selected model was optimized to focus on six main object classes: person, car, bicycle, motorcycle, bus, and truck, using the MobileNet-SSD v2 architecture. Figure 13 show that train and test loss curve during training.

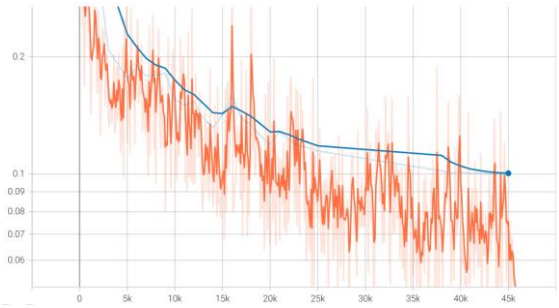

Figure 13. Train and test loss curve during training

Following the baseline optimization, transfer learning was applied to fine-tune *MobileNet SSD v2* using the customized six-class dataset. The evaluation at step 45,000 yielded strong detection results, achieving a *mean Average Precision (mAP)* of 0.724, *mAP@0.5* of 0.951, and *mAP@0.75* of 0.815, as summarized in Table 2. High precision and recall scores were maintained for medium and large objects, while detection accuracy for small objects remained limited (AP = 0.018). This limitation is consistent with known challenges in low-resolution SSD-based detectors, which could be mitigated through data augmentation or multi-scale feature enhancement [46].

The model achieved stable convergence with a total loss of 0.2488, comprising localization loss (0.0478), classification loss (0.0999), and regularization loss (0.1011). This balance indicates effective learning without overfitting, as also observed in similar optimization approaches [44], [42].

The evaluation results at step 45,000 indicate that the MobileNet-SSD v2 model performs remarkably well after transfer learning and fine-tuning, achieving an overall mean Average Precision (mAP) of 0.724 and a mAP@0.5 of 0.951, reflecting strong detection accuracy across all six object classes. The model also maintains good stability with a mAP@0.75 of 0.815, demonstrating reliable performance under stricter IoU thresholds. However, detection accuracy for small objects remains limited (AP = 0.018), suggesting that additional optimizations such as higher-resolution input or targeted

Vol. 7, No. 2, April 2026, Page. 1500-1514
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.2.5488

data augmentation may be beneficial. In contrast, detection performance for large objects is excellent (AP = 0.743, AR = 0.802), indicating that the model effectively identifies dominant objects within the scene. With a total loss of 0.2488, the model has reached a stable convergence, balancing localization and classification errors effectively. Overall, these results confirm that the optimization process successfully enhanced the model's precision, recall, and robustness for real-world object detection tasks.

Table 2. Model evaluation result (step 45000)

Metric	Score
Average Precision (mAP) @[IoU=0.50:0.95]	0.724
Average Precision (AP) @[IoU=0.50]	0.951
Average Precision (AP) @[IoU=0.75]	0.815
AP (small objects)	0.018
AP (medium objects)	0.437
AP (large objects)	0.743
Average Recall (AR) @[IoU=0.50:0.95, maxDets=1]	0.667
AR (maxDets=10)	0.770
AR (maxDets=100)	0.785
AR (small objects)	0.056
AR (medium objects)	0.523
AR (large objects)	0.802
Localization Loss	0.0478
Classification Loss	0.0999
Regularization Loss	0.1011
Total Loss	0.2488

3.4. Real-Time Deployment and System Testing

The optimized model was integrated into the IoT-based security prototype, as shown in Figure 14 and Figure 15, featuring a graphical dashboard for real-time monitoring. The system maintained real-time operation above 24 FPS with consistent detection confidence levels exceeding 0.8. CPU utilization remained below 93%, ensuring stable performance without overheating or frame lag. The practical deployment validated that the proposed optimization successfully enabled real-time object detection on low-power hardware while preserving detection reliability.

Overall, the experimental results demonstrate that the proposed optimization of MobileNet SSD v2 effectively improves the trade-off between accuracy and efficiency, outperforming the original baseline and offering a deployable solution for intelligent IoT-based security applications.

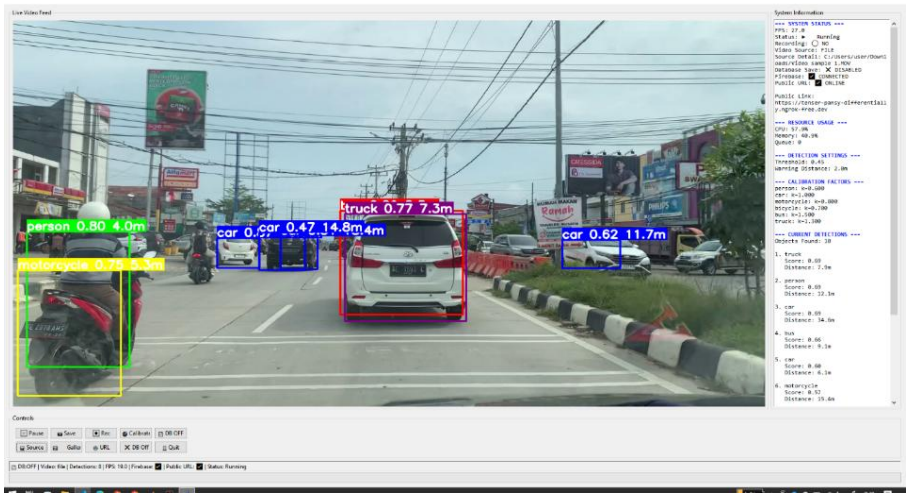

Figure 14. Real-time model testing from dashboard

Vol. 7, No. 2, April 2026, Page. 1500-1514
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.2.5488

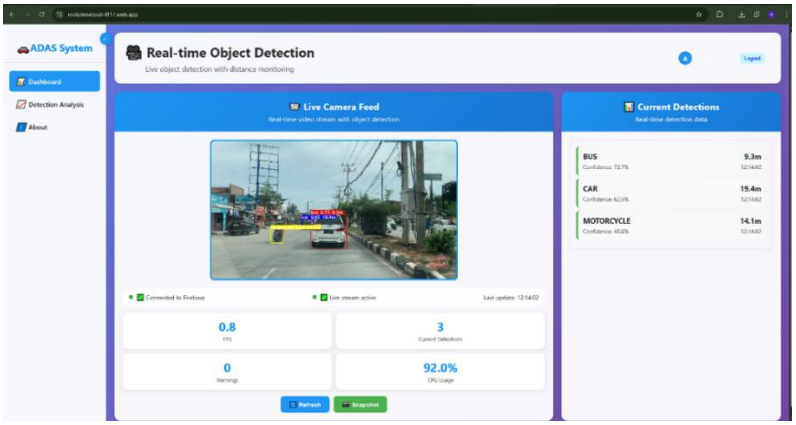

Figure 15. Real-time model testing from website

4. DISCUSSIONS

The experimental results demonstrate that the applied optimization strategies pruning, quantization, and transfer learning successfully improved the performance of the MobileNet SSD v2 model for real-time object detection on resource-limited IoT devices. Compared to the baseline model, the optimized architecture produced a noticeable improvement in inference speed, increasing from 21.9 FPS to more than 24 FPS on the Raspberry Pi 5 platform. This enhancement indicates that reducing computational complexity through model compression and selective class filtering can significantly increase throughput without compromising system stability. These findings are consistent with previous research [5], [11], [21], [32]. which also reported performance gains from model compression and input downscaling.

In terms of detection accuracy, the fine-tuned model achieved an mAP of 0.724 and a high mAP@0.5 of 0.951, demonstrating effective generalization to the six selected object classes: person, car, motorcycle, bicycle, bus, and truck. These results outperform typical outcomes obtained from default (unoptimized) MobileNet SSD implementations on embedded devices, as reported in prior studies such as Chiu et al. [21] and Alzubaidi et al. [32], which generally emphasize the trade-off between accuracy and speed.

The model exhibits significantly lower accuracy for small objects (AP = 0.018) due to the limited receptive field of the deeper layers in MobileNet SSD. Small objects occupy only a few pixels at lower resolutions, causing feature maps to lose fine-grained spatial information. SSD relies heavily on higher-level feature maps (e.g., 19×19 and 10×10), which are insufficient for detecting small-scale patterns. This observation aligns with prior studies indicating that SSD underperforms compared to multi-scale detectors such as YOLOv5 or EfficientDet for small-object scenarios [5]. Addressing this problem may require multi-scale feature enhancement or higher-resolution input.

The selection of MobileNet SSD v2 as the baseline architecture is further validated by the comparative evaluation results. Although MobileNet SSD v1 and v3 showed similar confidence scores, v2 consistently provided lower inference time and higher FPS stability. This finding supports observations from previous literature [21], [25], [47]. which highlights the improved feature extraction efficiency of MobileNet v2 due to its inverted residual blocks and linear bottlenecks.

In practical deployment, the optimized model maintained stable real-time performance with CPU utilization under 93% and detection confidence above 0.8 during continuous testing. The system exhibited reliable behavior under real-world scenarios, validating its suitability for IoT-based security applications such as monitoring pedestrians and vehicles. Compared to related works on vehicle detection and surveillance systems [19], [26], [35], [36], [37] the proposed system demonstrates improved balance between latency, accuracy, and resource efficiency, particularly because it is fully tested on an edge device without relying on external servers or GPU acceleration.

Vol. 7, No. 2, April 2026, Page. 1500-1514
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.2.5488

Despite these achievements, several limitations were identified. First, the reduced detection accuracy for small and partially occluded objects indicates the need for further improvements, such as enhanced data augmentation or multi-scale detection features. Second, pruning and quantization were applied conservatively to avoid accuracy degradation, meaning additional optimization layers (e.g., TensorRT or ONNX acceleration) could potentially yield even greater performance improvements. Lastly, the current system focuses on six classes only; expanding the class set while maintaining real-time capability remains an open challenge for future exploration.

Overall, the discussion confirms that the optimization successfully enhances the performance of MobileNet SSD v2 on constrained IoT hardware while preserving accuracy and operational feasibility. The results highlight the benefits of selective class training, input resolution tuning, and lightweight model design, contributing meaningful insights to the development of edge-based intelligent security systems.

5. CONCLUSION

This study demonstrates that MobileNet SSD v2 can be effectively optimized using pruning, quantization, and transfer learning to achieve real-time object detection on resource-constrained IoT devices. The optimized model achieved more than 24 FPS on Raspberry Pi 5 with stable CPU utilization and competitive accuracy (mAP 0.724 and mAP@0.5 0.951), confirming that lightweight object detection models can operate efficiently on low-power hardware while maintaining reliable performance.

The main scientific contribution of this research lies in showing that combining model compression techniques with selective class-specific fine-tuning significantly improves inference speed and energy efficiency on edge devices without requiring GPU acceleration. The system prototype demonstrated stable real-time detection with minimal CPU load and consistent confidence above 0.8, highlighting its practical feasibility for IoT-based security applications.

Despite these achievements, small and partially occluded objects remain challenging to detect due to the limited receptive field and low-resolution feature maps. Future work will focus on addressing these limitations through advanced data augmentation, multi-scale feature fusion, higher-resolution inputs, and TensorRT-based acceleration. Additionally, expanding the dataset and integrating supplementary modules such as tracking, alert notifications, and cloud connectivity are expected to further enhance the robustness and scalability of the IoT-based security system.

CONFLICT OF INTEREST

The authors gratefully acknowledge the financial support from the Directorate General of Research and Development, Ministry of Higher Education, Science, and Technology of the Republic of Indonesia, through the 2025 Beginner Lecturer Research Scheme (Contract No. 1483bo/IT9.2.1/PT.01.03/2025), which enabled this research.

REFERENCES

[1] M. Satyanarayanan, “The Emergence of Edge Computing,” *Computer (Long Beach Calif)*, vol. 50, no. 1, pp. 30–39, 2017, doi: 10.1109/MC.2017.9.

[2] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge Computing: Vision and Challenges,” *IEEE Internet Things J*, vol. 3, no. 5, pp. 637–646, 2016, doi: 10.1109/JIOT.2016.2579198.

[3] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog Computing and Its Role in the Internet of Things,” in *Proceedings of the First MCC Workshop on Mobile Cloud Computing*, 2012, pp. 13–16. doi: 10.1145/2342509.2342513.

[4] L. Qi, X. Zhang, W. Dou, and Q. Ni, “A Survey on Edge Computing in the Internet of Things,” *IEEE Access*, vol. 8, pp. 195555–195572, 2020, doi: 10.1109/ACCESS.2020.3032861.

Vol. 7, No. 2, April 2026, Page. 1500-1514
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.2.5488

[5] W. Liu *et al.*, "SSD: Single shot multibox detector," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9905 LNCS, pp. 21–37, 2016, doi: 10.1007/978-3-319-46448-0_2.

[6] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[7] M. Tan, R. Pang, and Q. V Le, "EfficientDet: Scalable and efficient object detection," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10781–10790, 2020, doi: 10.1109/CVPR42600.2020.01079.

[8] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *CVPR*, 2018, pp. 4510–4520. doi: 10.1109/CVPR.2018.00474.

[9] A. Howard and others, "Searching for MobileNetV3," in *ICCV*, 2019, pp. 1314–1324. doi: 10.1109/ICCV.2019.00140.

[10] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 6848–6856. doi: 10.1109/CVPR.2018.00716.

[11] R. Zhang, L. Sun, and Y. Huang, "Improving Embedded Object Detection using MobileNet SSD," *Sensors*, vol. 22, no. 15, p. 5654, 2022, doi: 10.3390/s22155654.

[12] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[13] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning Filters for Efficient Convolutional Neural Networks," *arXiv preprint arXiv:1608.08710*, 2016, [Online]. Available: https://arxiv.org/abs/1608.08710

[14] B. Jacob, S. Kligys, B. Chen, and et al., "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 2704–2713. doi: 10.1109/CVPR.2018.00286.

[15] Y. Wu, C. Huang, and X. Liu, "Quantization for Deep Neural Networks: A Survey," *IEEE Access*, vol. 8, pp. 217123–217136, 2020, doi: 10.1109/ACCESS.2020.3040789.

[16] T. Liang, J. Lu, and X. Chen, "Pruning and Quantization for Deep Neural Network Compression: A Survey," *arXiv preprint arXiv:2101.09671*, 2021, [Online]. Available: https://arxiv.org/abs/2101.09671

[17] Y. Choi, J. Lee, and S. Park, "Recent Advances in Model Compression and Acceleration," *IEEE Access*, vol. 9, pp. 157526–157538, 2021, doi: 10.1109/ACCESS.2021.3131868.

[18] K. Simonyan, A. Vedaldi, and A. Zisserman, "Edge AI: Concepts, Key Technologies and Future Directions," *IEEE Trans Pattern Anal Mach Intell*, 2020, doi: 10.1109/TPAMI.2020.3034684.

[19] N. Sreenu and M. J. Durai, "Intelligent Video Surveillance: A Review," *J Big Data*, vol. 6, no. 1, pp. 1–27, 2019, doi: 10.1186/s40537-019-0262-0.

[20] T. Sibanda, M. A. Adedoyin, and P. S. Pillay, "Real-Time IoT-Based Vehicle Detection for Smart Transportation Systems," *IEEE Access*, vol. 9, pp. 89687–89699, 2021, doi: 10.1109/ACCESS.2021.3087937.

[21] Y. C. Chiu, C. Y. Tsai, M. Da Ruan, G. Y. Shen, and T. T. Lee, "Mobilenet-SSDv2: An Improved Object Detection Model for Embedded Systems," *2020 International Conference on System Science and Engineering, ICSSE 2020*, pp. 0–4, 2020, doi: 10.1109/ICSSE50014.2020.9219319.

[22] S. K. Ghosh and et al., "Energy-Efficient Approximate Edge Inference Systems," *ACM Trans Embed Comput Syst*, vol. 22, no. 6, pp. 1–23, 2023, doi: 10.1145/3589766.

[23] U. Iqbal and et al., "Edge-Computing Video Analytics Solution for Automated Waste Monitoring," *Sensors*, vol. 22, no. 20, p. 7821, 2022, doi: 10.3390/s22207821.

[24] W. K. H. Lua and et al., "Lightweight CNN-Based Deep Neural Networks Application in Edge Computing and IoT," *J Ambient Intell Humaniz Comput*, vol. 13, pp. 2025–2041, 2022, doi: 10.1007/s12652-021-03567-3.

[25] A. G. Howard *et al.*, "Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv preprint arXiv:1704.04861*, 2017.

Vol. 7, No. 2, April 2026, Page. 1500-1514
https://jutif.if.unsoed.ac.id
DOI: https://doi.org/10.52436/1.jutif.2026.7.2.5488

[26] M. A. Haque and others, "Vehicle and Person Detection in Surveillance Video Using Lightweight CNN," *IEEE Access*, vol. 10, pp. 48320–48332, 2022, doi: 10.1109/ACCESS.2022.3178492.
[27] E. Wang and others, "Model Compression and Acceleration for Deep Neural Networks: The State of the Art," *IEEE Signal Process Mag*, vol. 37, no. 1, pp. 115–129, 2020, doi: 10.1109/MSP.2019.2951811.
[28] A. Yang and others, "Pruning Neural Networks: A Survey," *IEEE Trans Pattern Anal Mach Intell*, 2023, doi: 10.1109/TPAMI.2023.3237834.
[29] H. Park and others, "Survey on Model Compression for On-Device Deep Learning," *ACM Comput Surv*, vol. 55, no. 8, pp. 1–35, 2023, doi: 10.1145/3527150.
[30] J. Anwar, K. Muhammad, and A. K. Sangaiah, "IoT-Based Intelligent Security Using Mobile Lightweight Detectors," *IEEE Access*, vol. 10, pp. 7443–7455, 2022, doi: 10.1109/ACCESS.2022.3149794.
[31] J. Park and S. Lee, "GPU vs Edge-Device Performance Comparison for Object Detection Models," *Sensors*, vol. 21, no. 11, p. 3664, 2021, doi: 10.3390/s21113664.
[32] A. Alzubaidi and others, "Real-Time Object Detection on Raspberry Pi 4 Using MobileNet SSD," *Computers*, vol. 10, no. 12, p. 155, 2021, doi: 10.3390/computers10120155.
[33] M. H. Rahman and others, "MobileNet-Based Traffic Monitoring on Raspberry Pi 3/4," *IEEE Access*, vol. 8, pp. 181860–181870, 2020, doi: 10.1109/ACCESS.2020.3027561.
[34] Raspberry Pi Foundation, "Raspberry Pi 5 Product Page," 2023.
[35] S. N. Mousavi and others, "Real-Time Multi-Class Vehicle Detection in IoT Surveillance," *J Real Time Image Process*, 2022, doi: 10.1007/s11554-021-01164-4.
[36] J. K. Kim and H. J. Lee, "Smart City Surveillance Using Embedded Deep Learning," *Sensors*, vol. 21, no. 6, p. 2058, 2021, doi: 10.3390/s21062058.
[37] S. F. H. Khilji and others, "Deep Learning-Based Surveillance for Smart Transportation," *IEEE Access*, vol. 10, pp. 50839–50853, 2022, doi: 10.1109/ACCESS.2022.3162404.
[38] D. R. V Krishna and others, "Small-Object Detection Challenges in Security Video," *IEEE Access*, vol. 11, pp. 54483–54499, 2023, doi: 10.1109/ACCESS.2023.3280931.
[39] Z. Dhaief and N. El abbadi, "Road Signs Detection Using SSD MobileNetV2," *Karbala International Journal of Modern Science*, vol. 10, 2024, doi: 10.33640/2405-609X.3373.
[40] T.-Y. Lin *et al.*, "Microsoft {COCO}: Common Objects in Context," in *European Conference on Computer Vision (ECCV)*, 2014, pp. 740–755.
[41] Y.-C. Chiu, C.-Y. Tsai, M.-D. Ruan, G.-Y. Shen, and T.-T. Lee, "Mobilenet-SSDv2: An Improved Object Detection Model for Embedded Systems," in *2020 International Conference on System Science and Engineering (ICSSE)*, 2020, pp. 1–5. doi: 10.1109/ICSSE50014.2020.9219319.
[42] D. Biswas, H. Su, C. Wang, A. Stevanovic, and W. Wang, "An automatic traffic density estimation using Single Shot Detection (SSD)and MobileNet-SSD," *Physics and Chemistry of the Earth*, vol. 110, no. December, pp. 176–184, 2019, doi: 10.1016/j.pce.2018.12.001.
[43] W. Sun, S. Chen, L. Shi, Y. Li, and Z. Lin, "Vehicle Following in Intelligent Multi-Vehicle Systems Based on SSD-MobileNet," *Proceedings - 2019 Chinese Automation Congress, CAC 2019*, pp. 5004–5009, 2019, doi: 10.1109/CAC48633.2019.8996181.
[44] Y. C. Chiu, C. Y. Tsai, M. Da Ruan, G. Y. Shen, and T. T. Lee, "Mobilenet-SSDv2: An Improved Object Detection Model for Embedded Systems," *2020 International Conference on System Science and Engineering, ICSSE 2020*, pp. 0–4, 2020, doi: 10.1109/ICSSE50014.2020.9219319.
[45] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Jun. 2016, pp. 779–788. doi: 10.1109/CVPR.2016.91.
[46] W. Liu *et al.*, "SSD: Single shot multibox detector," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9905 LNCS, pp. 21–37, 2016, doi: 10.1007/978-3-319-46448-0_2.
[47] A. Miranto, "Real Time Object Detection Menggunakan Mobilenet-SSD pada Sistem Keamanan Ruangan dengan Bot Telegram Sebagai Notifikasi User," *JELIKU (Jurnal Elektronik Ilmu Komputer Udayana)*, vol. 13, no. 1, p. 211, Aug. 2024, doi: 10.24843/JLK.2024.v13.i01.p21.