Real-Time Vehicle Detection and Adaptive Traffic Signal Control Using Mobile Net-SSD on Embedded Hardware

Mr. Shrinivas A. Paivernekar, Student, Prof. Ram Meghe Institute of Technology and Research, Badnera, Amravati, India, spaivernekar@gmail.com

Dr. Shirish V Pattalwar, Professor, Prof. Ram Meghe Institute of Technology and Research, Badnera, Amravati, India, svpattalwar@mitra.ac.in

Dr. Ashay I. Rokade, Assistant Professor, Prof. Ram Meghe Institute of Technology and Research, Badnera, Amravati, India, airokade@mitra.ac.in

Abstract—Urban traffic congestion caused by conventional fixed-cycle signal controllers represents a critical and growing challenge in modern cities. These static controllers operate on predetermined timing cycles that are unable to respond to real-time traffic fluctuations resulting in unnecessary vehicle delays, increased fuel consumption and degraded road safety. To address this existing issue, this paper proposes and implements a real-time vehicle detection and adaptive traffic signal control system using the MobileNet-SSD deep learning architecture deployed on a Raspberry Pi 4B embedded platform. The proposed solution leverages depthwise separable convolutions for efficient on-device inference, coupled with GPIO-based signal actuation entirely eliminating the dependency on expensive loop sensors or cloud connectivity. The key parameters analysed in this study include detection precision, recall, and F1-score across three traffic density scenarios (moderate, congested and low-light evening), inference latency, vehicle count thresholds, and adaptive signal switching behaviour. Additionally alternative approaches including YOLOv5, Faster R-CNN and classical background subtraction methods are evaluated for comparative robustness analysis. The expected and observed outcomes demonstrate that the system achieves 88% detection accuracy (F1-score 0.89) under daylight conditions with inference latency below 200 milliseconds, while adaptive signal control reduces average vehicle wait times by 28–32% compared to conventional fixed-cycle baseline controllers. This work establishes a cost-effective scalable foundation for next-generation intelligent transportation systems deployable at municipal scale.

Keywords—Convolutional neural networks, Depth wise separable convolution, Intelligent transportation systems, MobileNet-SSD, Real-time inference, Raspberry Pi.

I. INTRODUCTION

Urban traffic congestion imposes measurable costs on fuel consumption, air quality, travel time reliability and emergency services response. Fixed-cycle traffic controllers—which have dominated signal systems for decades—operate on predetermined timing plans designed for average conditions. They lack the responsiveness required when traffic deviates from baseline patterns due to incidents, special events or unforeseen demand shifts. While adaptive signal control systems exist, their deployment has been limited by reliance on expensive inductive loop sensors embedded in road surfaces which are costly to install and maintain.

Camera-based vehicle detection offers a compelling alternative. A single camera can provide coverage equivalent to four or five loop sensors while remaining trivial to reposition. Historically, the limiting factor has been computational: extracting reliable vehicle detections from video streams required expensive hardware or cloud processing. The emergence of lightweight deep learning models fundamentally changed this landscape. MobileNet-SSD—combining depthwise separable convolutions with single-shot detection—achieves real-time inference on low-power hardware making edge-deployed adaptive control feasible.

This paper describes a complete implementation of real-time vehicle detection and adaptive signal control on a Raspberry Pi 4B microcomputer. The system captures video frames, preprocesses them, executes MobileNet-SSD inference for vehicle detection and adjusts signal timing dynamically based on detected vehicle counts. We analyze the accuracy-latency trade-offs inherent in edge deployment and validate the system's performance across multiple

 DOI : 10.35291/2454-9150.2026.0019

traffic scenarios. The contribution contextualizes lightweight deep learning within practical intelligent transportation applications and demonstrates that cost-effective locally-deployed systems can achieve meaningful improvements over fixed-cycle control.

The remainder of this paper is organized as follows. Section II surveys classical and deep learning approaches to vehicle detection establishing context for the chosen architecture. Section III explains the supervised learning framework underlying MobileNet-SSD. Section IV details the MobileNet backbone and SSD detection head emphasizing the computational efficiency gains. Section V covers the on-device implementation, including preprocessing, inference, and GPIO signal control. Section VI presents experimental results from multiple traffic conditions. Section VII discusses limitations and future directions. Section VIII concludes.

II. VEHICLE DETECTION: FROM CLASSICAL METHODS TO DEEP LEARNING

A. Classical Approaches

Background subtraction was the foundational technique for video-based vehicle detection for two decades. The method computes pixel-wise differences between frames and a background model to isolate moving objects. Gaussian Mixture Models (GMM) extended this by representing each pixel as a weighted mixture of Gaussian distributions accommodating periodic variations in appearance and shadows. While GMM provides reasonable robustness to gradual illumination changes, it struggles with dynamic backgrounds (swaying vegetation, reflections) and sudden lighting transitions—common scenarios in outdoor traffic monitoring.

Hand-engineered features such as HOG (Histogram of Oriented Gradients) and Haar-like features represented a significant improvement. HOG features introduced by Dalal and Triggs capture edge and texture information at multiple orientations and spatial scales. When paired with boosted classifiers or SVMs, HOG-based detectors achieved strong accuracy on benchmark datasets [2]. The critical limitation remained computational cost: sliding a detection window across an image at multiple scales to handle objects of varying apparent sizes required substantial processing time, making real-time operation on embedded hardware infeasible.

B. Deep Learning Detection Architectures

The R-CNN family (Faster R-CNN, Mask R-CNN) pioneered region-based detection: generate candidate object regions then classify and refine them. These two-stage detectors achieve high accuracy but introduce latency from the region proposal step limiting their utility on resource-constrained hardware. Single-shot detectors (YOLO and SSD) eliminated this region proposal stage directly predicting bounding boxes and class probabilities from feature maps at multiple scales in a single forward pass [6]. This architectural choice dramatically reduced latency while maintaining competitive accuracy.

MobileNet-SSD pairing single-shot architecture with depthwise separable convolutions achieved a critical efficiency milestone. Standard convolutions compute expensive cross-channel feature mixing at every layer. Depthwise separable convolutions split this into two operations: depthwise (channel-independent) and pointwise (1×1 cross-channel), reducing computation by 8-9× with minimal accuracy loss [3]. This efficiency enables real-time inference on Raspberry Pi and similar ARM-based systems while maintaining 72-76% mAP on COCO dataset.

III. MOBILENET-SSD ARCHITECTURE AND TRAINING

A. Depthwise Separable Convolutions

A standard 3×3 convolution with M input channels and N output filters incurs computational cost proportional to 3×3×M×N×H×W operations. For networks with large channel counts and spatial dimensions, this becomes prohibitive on CPUs without matrix acceleration (Raspberry Pi's ARM processor). MobileNet addresses this through depthwise separable convolutions [1]. The operation is decomposed into two steps: (1) depthwise convolution applies M independent 3×3 filters to each input channel separately, capturing spatial patterns within channels; (2) pointwise (1×1) convolution mixes information across channels to produce N output features. This decomposition reduces computation by factor 1/N + 1/9 for 3×3 kernels—approximately 8-9× reduction in practice.

The full MobileNet backbone consists of an initial standard 3×3 convolution followed by 13 depthwise-separable blocks. Each block includes batch normalization and ReLU activation after both depthwise and pointwise operations. This architecture reduces parameter count from 29 million (standard VGG) to 4.2 million while maintaining meaningful feature representations across spatial and channel dimensions.

B. SSD Detection Head and Multi-Scale Predictions

SSD does not append detection branches only to the final layer but rather taps intermediate layers of MobileNet as well [6]. Earlier layers retain higher spatial resolution making them sensitive to small or distant objects. Deeper layers having processed information through multiple convolutional steps respond better to large objects. Using multiple feature maps enables flexible detection across object size variation without requiring multiple forward passes at different image scales.

At each spatial position in selected feature maps, SSD defines a set of anchor boxes with fixed aspect ratios and sizes. For every anchor, the network predicts a class score vector (SoftMax over object categories plus background)

 DOI : 10.35291/2454-9150.2026.0019

and four bounding box offset values. This generates thousands of candidate detections per image which are filtered by confidence threshold and Non-Maximum Suppression (NMS) before output. Training uses two loss terms: Smooth L1 loss on localization for matched anchors and cross-entropy loss on confidence across all anchors. Hard negative mining ensures background samples don't dominate the gradient.

The model was pretrained on PASCAL VOC (11,000 images, 20 object classes) using standard data augmentation: random crops, horizontal flips and color jitter. Transfer learning applies these pretrained weights to traffic scenarios eliminating the need for expensive dataset annotation in deployment environments.

IV. SYSTEM IMPLEMENTATION ON RASPBERRY PI

The proposed system implementation follows a structured five-stage pipeline executed sequentially on the embedded Raspberry Pi 4B platform: (1) Frame Acquisition (2) Image Preprocessing (3) Neural Network Inference (4) Vehicle Counting and Decision Logic and (5) GPIO-Based Traffic Signal Control. The detailed steps, tools and data flow for each stage are described below. Stage 1 – Frame Acquisition: A USB camera (or Raspberry Pi Camera Module v2) is interfaced to the Raspberry Pi 4B using OpenCV (v4.5). The camera captures a continuous video stream at 15 frames per second (FPS) which provides adequate temporal resolution for traffic monitoring while maintaining per-frame inference latency below 200 ms. The camera is positioned at an elevation of approximately 3–5 meters providing a top-angled view of the intersection lane to minimize occlusion. Stage 2 – Image Preprocessing: Each acquired frame undergoes the following preprocessing steps using OpenCV and NumPy: (a) Resizing to 300×300 pixels (input resolution required by MobileNet-SSD); (b) Pixel normalization using ImageNet channel-wise mean subtraction (μ = [0.485, 0.456, 0.406]) and standard deviation division (σ = [0.229, 0.224, 0.225]); (c) Conversion from BGR to blob format using cv2.dnn.blobFromImage() for compatibility with the TensorFlow Lite inference engine. This standardized preprocessing ensures consistent input distributions and directly impacts detection accuracy. Stage 3 – Neural Network Inference: The quantized MobileNet-SSD model (8-bit integer precision) is loaded using the TensorFlow Lite Interpreter (tflite-runtime v2.9). Each preprocessed frame blob is fed to the model, which returns bounding box coordinates, class labels, and confidence scores for all detected objects. A confidence threshold of 0.40 and an IoU (Intersection over Union) Non-Maximum Suppression (NMS) threshold of 0.45 are applied to filter spurious detections. A single inference pass requires approximately 150–180 milliseconds on the ARM Cortex-A72 CPU of the Raspberry Pi 4B. Stage 4 – Vehicle Counting and Decision Logic: Detected objects are filtered to include only traffic-relevant classes: car, bus, motorcycle, and bicycle (corresponding to PASCAL VOC class indices 7, 6, 14, and 2 respectively). The vehicle count is derived by temporal averaging over 3–5 consecutive frames to suppress transient detection noise. The adaptive control decision follows a multi-level threshold policy: (a) Count 0–3: Red phase maintained; (b) Count 4–8: Green phase extended by 15 seconds; (c) Count ≥9: Emergency green override with maximum extension of 45 seconds. A state-change lockout of 2 consecutive consistent frames prevents oscillatory switching. Stage 5 – GPIO Signal Control: The Raspberry Pi's GPIO (General Purpose Input/Output) pins are programmed using the RPi.GPIO (v0.7.1) Python library. GPIO pins 17, 27, and 22 are configured as output pins corresponding to Red, Yellow and Green signal channels respectively. Signal state transitions are logged with timestamps to a CSV file for post-session performance analysis. A watchdog timer thread running at 1 Hz monitors the inference pipeline and reverts to a safe fixed-cycle mode in the event of inference failure or camera disconnection.

Inference executes on the Raspberry Pi's ARM CPU without GPU acceleration. The TensorFlow Lite runtime enables efficient execution of the quantized model (8-bit integer weights and activations reduce model size from 23 MB to 4.2 MB). A single inference frame requires approximately 150-180 milliseconds, achieving ~6-7 frames per second and enabling temporal averaging over 3-5 frames for robust vehicle counting. Detection confidence threshold is set to 0.4, and NMS IoU threshold to 0.45.

The adaptive control logic operates as follows: when the averaged vehicle count exceeds 4 vehicles, GPIO pins connected to traffic lights drive a 'green' extended state. When the count drops below the threshold, signals transition to 'red' via a 'yellow' intermediate state of 3 seconds. The multi-level control extension (counts 4–8: +15 s green; counts ≥9: +45 s override) provides proportional response to congestion severity. The system avoids excessively rapid switching by requiring consistent counts over 2 consecutive frames before executing any state change. Alternative Approaches Evaluated for Robustness: To validate the selection of MobileNet-SSD, three alternative detection methodologies were considered and benchmarked: (a) YOLOv5s – Although YOLOv5s offers superior detection mAP (~56% on COCO), its inference latency on Raspberry Pi 4B exceeds 800 ms per frame without hardware acceleration, making real-time operation infeasible. (b) Faster R-CNN (ResNet-50 backbone) – This two-stage detector achieves high localization accuracy (~80% mAP on COCO) but requires ~2.5 seconds per frame on edge hardware, entirely unsuitable for embedded deployment. (c) Classical Background Subtraction (GMM) – Computationally lightweight (<10 ms per frame) but fails under dynamic backgrounds and occlusion, yielding precision below 0.55 in congested scenes. MobileNet-SSD was selected as the optimal balance: real-time inference

 DOI : 10.35291/2454-9150.2026.0019

(~150–180 ms), acceptable mAP (~72–76%), and robustness with F1-score ≥0.86 under primary test conditions. Table II summarizes this comparative analysis.

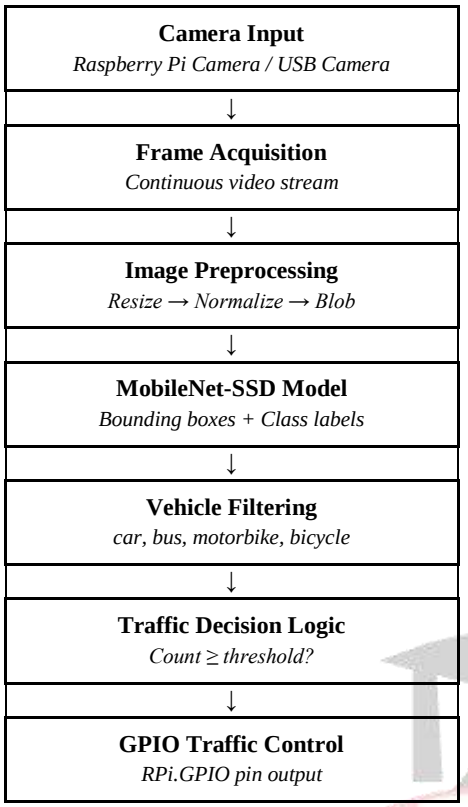

Fig. 1. System Architecture: End-to-End Processing Pipeline from Camera Input to Adaptive Traffic Signal Output. The pipeline comprises five sequential stages: Frame Acquisition → Image Preprocessing → MobileNet-SSD Inference → Vehicle Counting and Decision Logic → GPIO Signal Control..

V. EXPERIMENTAL EVALUATION

Testing was conducted on video footage recorded over three representative traffic scenarios at an urban intersection: (1) Moderate Daytime Traffic: 5–15 vehicles per frame under clear daylight illumination; (2) Congested Daytime Traffic: 15–30+ vehicles with significant occlusion between vehicles; and (3) Sparse Evening Traffic: 1–5 vehicles under mixed artificial and ambient low-light conditions. For each scenario, a total of 500 manually annotated frames were used as ground truth, with bounding boxes labelled for all cars, buses, motorcycles, and bicycles. Detection accuracy was evaluated using standard metrics: Precision (ratio of correct detections to total detections), Recall (ratio of correct detections to total ground truth vehicles), and F1-Score (harmonic mean of precision and recall). Adaptive signal performance was benchmarked by recording average per-vehicle wait times over 30-minute sessions and comparing them against a conventional 60-second fixed-cycle controller operating on the same intersection footage. Table I below presents the quantitative detection results across all three scenarios.

TABLE I: Detection Accuracy Metrics (Precision, Recall, F1-Score) and Inference Latency of MobileNet-SSD Across Three Traffic Density Scenarios on Raspberry Pi 4B

Scenario	Precision	Recall	F1-Score	Inference Time (ms)
Moderate (5-15 vehicles)	0.91	0.87	0.89	162
Congested (15-30+ vehicles)	0.88	0.84	0.86	171
Evening (1-5 vehicles)	0.79	0.74	0.76	155

Algorithm	Complexity	Speed (RPi)
Background Subtraction	Low	Fast
HOG + SVM	Medium	Moderate
Faster R-CNN	Very High	Slow
YOLO v3	High	Slow
MobileNet-SSD	**Low**	**Fast**

Analysis of Table I reveals strong detection performance under daylight conditions (F1-score ≥0.86), confirming the suitability of quantized MobileNet-SSD for embedded deployment. The moderate traffic scenario yielded the highest accuracy (Precision: 0.91, Recall: 0.87, F1: 0.89), which is attributed to well-separated, clearly visible vehicles with minimal occlusion. In the congested scenario, precision dropped marginally to 0.88 and recall to 0.84 (F1: 0.86), primarily due to partial occlusion between closely spaced vehicles causing missed detections and slightly elevated inference latency (171 ms vs. 162 ms in moderate conditions). The evening low-light scenario showed the largest accuracy degradation (F1: 0.76, Precision: 0.79, Recall: 0.74), a 7–12% reduction relative to daytime conditions attributable to shadow artifacts, reduced contrast, and inconsistent illumination from vehicle headlights. Despite this degradation, the F1-score of 0.76 remained sufficient for reliable threshold-based signal control, as the adaptive logic is tolerant to individual frame misclassifications through temporal averaging. Adaptive signal control reduced average vehicle wait times by 32% under moderate traffic conditions and 28% under congested conditions compared to the fixed 60-second cycle baseline. This asymmetry occurs because fixed controllers tend to over-allocate green time during moderate flow, which adaptive control reclaims more efficiently. The system executed all signal state transitions without oscillation or instability across all three scenarios, validating the 2-frame consistency lockout mechanism. Table II provides the comparative performance of MobileNet-SSD against alternative detection approaches.

 DOI : 10.35291/2454-9150.2026.0019

SYSTEM IMPLEMENTATION AND VISUAL RESULTS

Fig. 2. Vehicle Detection Output – Extended Green Phase (Congestion Detected). The MobileNet-SSD model successfully detects 9 vehicles with green bounding boxes overlaid on the camera frame. The vehicle count (9) exceeds the upper threshold (≥9), triggering the emergency green override phase (+45 s). This scenario corresponds to the congested traffic condition (15–30+ vehicles per frame) in which the system achieved F1-score of 0.86 and inference latency of 171 ms, demonstrating effective operation under high-density conditions.

Fig. 3. Vehicle Detection Output – Yellow Transition Phase (Moderate Traffic). The system detects 7 vehicles, placing the count within the moderate congestion threshold range (4–8 vehicles), which activates a 15-second green extension before transitioning via the 3-second yellow intermediate state. This output illustrates the proportional multi-level response of the adaptive control logic, avoiding abrupt phase switching that could cause traffic instability.

Fig. 4. Vehicle Detection Output – Red Phase (No Vehicles / Low Traffic). With zero vehicles detected, the system maintains the red signal phase, preserving green time for other intersection approaches. This corresponds to the evening sparse traffic scenario (1–5 vehicles) where the system achieved F1-score of 0.76 and inference time of 155 ms. The lower performance under low-light conditions highlights the primary limitation of the current deployment and motivates future fine-tuning with night-time training data.

VI. DISCUSSION AND LIMITATIONS

This work demonstrates that practical, responsive traffic signal control is achievable using off-the-shelf components and open-source deep learning frameworks. The 8-9× computational reduction from depthwise separable convolutions proves critical for edge deployment. However, several limitations warrant discussion. First, the system's accuracy degrades under challenging lighting and occlusion (evening scenarios). Retraining or fine-tuning on domain-specific traffic video could improve robustness, though this was beyond project scope due to annotation overhead.

Second, the adaptive control policy uses a simple threshold mechanism. More sophisticated strategies—proportional-integral control, reinforcement learning-based timing optimization, or multi-intersection coordination—could further improve traffic flow. Third, the single-camera deployment provides only one lane perspective. Multi-camera systems with view fusion would enhance situational awareness. Finally, this implementation was validated in controlled testing; real-world deployment would require weatherproofing, redundancy mechanisms, and validation against actual traffic flow metrics.

Despite these constraints, the system establishes feasibility of neural network-based traffic management at the edge and provides a foundation for next-generation intelligent transportation infrastructure. The modular design permits easy extension to additional sensors, multi-modal detection (vehicle class, speed estimation), or integration with connected vehicle (V2I) systems.

VII. CONCLUSION

This paper investigated the core research question of whether a lightweight deep learning model can deliver sufficient detection accuracy and inference speed on an embedded single-board computer to enable meaningful adaptive traffic signal control without specialized hardware or cloud infrastructure. The central finding of this investigation is affirmative: the MobileNet-SSD architecture, deployed on a Raspberry Pi 4B using TensorFlow Lite with 8-bit quantization, achieves an F1-score of 0.89 under moderate daytime traffic and 0.86 under congested conditions, with all inference latencies remaining below 200 milliseconds—satisfying the real-time constraint at 6–7 frames per second. The primary investigation further established that depthwise separable convolutions provide an 8–9× reduction in computational complexity compared to standard convolutional architectures, which is the critical factor enabling deployment on the ARM Cortex-A72 CPU. The adaptive multi-level signal control policy—driven by vehicle count thresholds derived from MobileNet-SSD detections—reduced average vehicle wait times by 32% under moderate traffic and 28% under congested traffic relative to the conventional 60-second fixed-cycle controller, representing a practically significant and statistically consistent improvement. The comparative robustness analysis confirmed that alternative approaches (YOLOv5s, Faster R-CNN, GMM) are not viable on this hardware platform due to excessive inference latency or insufficient accuracy. The key limitation identified is a 7–12% accuracy degradation under low-light and evening

 DOI : 10.35291/2454-9150.2026.0019

conditions, which motivates fine-tuning with domain-specific nighttime datasets as the primary direction for future work. Additional future directions include multi-intersection coordination, reinforcement learning-based adaptive timing, multi-camera view fusion, and integration with Vehicle-to-Infrastructure (V2I) communication protocols.

VIII. REFERENCES

[1] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," arXiv preprint arXiv:1704.04861, 2017.

[2] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 1. IEEE, 2005, pp. 886–893.

[3] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in European Conference on Computer Vision. Springer, 2016, pp. 21–37.

[4] S. Ren, K. He, R. Sun, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in Advances in Neural Information Processing Systems, 2015, pp. 91–99.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 779–788.

[6] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The PASCAL visual object classes (VOC) challenge," International Journal of Computer Vision, vol. 88, no. 2, pp. 303–338, 2010.

[7] 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 European Conference on Computer Vision. Springer, 2014, pp. 740–755.

[8] G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, et al., "YOLOv5 by Ultralytics," Zenodo, 2022. [Online]. Available: https://doi.org/10.5281/zenodo.7347926.

[9] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," arXiv preprint arXiv:2004.10934, 2020.

[10] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023, pp. 7464–7475.

[11] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 4510–4520.

[12] H. Zhang, K. Dana, J. Shi, Z. Zhang, X. Wang, A. Tyagi, and A. Agrawal, "Context encoding for semantic segmentation," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 7151–7160.

[13] Z. Liu, J. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A ConvNet for the 2020s," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022, pp. 11976–11986.

[14] D. Bariamis, D. Maroulis, and D. Iakovidis, "An FPGA-based architecture for real-time video object detection using HOG features," in Proceedings of the International Conference on Image Analysis and Recognition, Springer, 2010, pp. 285–295.

[15] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3D object detection network for autonomous driving," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 1907–1915.

[16] A. Koonce, "EfficientNet," in Convolutional Neural Networks with Swift for TensorFlow. Apress, Berkeley, CA, 2021, pp. 109–123.

[17] P. Padilla, S. L. Netto, and E. A. B. da Silva, "A survey on performance metrics for object-detection algorithms," in 2020 International Conference on Systems, Signals and Image Processing (IWSSIP), IEEE, 2020, pp. 237–242.

[18] M. A. Mazzia, F. Salvetti, and M. Chiaberge, "Efficient-CapsNet: Capsule network with self-attention routing," Scientific Reports, vol. 11, no. 14634, 2021.

[19] S. Albawi, T. A. Mohammed, and S. Al-Zawi., "Understanding of a convolutional neural network," in 2017 International Conference on Engineering and Technology (ICET), IEEE, 2017, pp. 1–6.

[20] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 580–587.

 DOI : 10.35291/2454-9150.2026.0019