A Real-Time Vehicle Detection System for ADAS in Autonomous Vehicles Using YOLOv11 Deep Neural Network on Embedded Edge Platforms

Mohammed Chaman
Laboratory of Electronic Systems Information Processing Mechanics and Energetics, Ibn Tofail University, Kenitra, Morocco
mohammed.chaman@uit.ac.ma (corresponding author)

Anas El Maliki
Laboratory of Electronic Systems Information Processing Mechanics and Energetics, Ibn Tofail University, Kenitra, Morocco
anas.elmaliki@uit.ac.ma

Noura Jariri
Laboratory of Electronic Systems Information Processing Mechanics and Energetics, Ibn Tofail University, Kenitra, Morocco
noura.jariri@uit.ac.ma

Azzedine El Mrabet
Laboratory of Advanced Systems Engineering, Ibn Tofail University, Kenitra, Morocco
azzedine.elmrabet@uit.ac.ma

Hamad Dahou
Laboratory of Electronic Systems Information Processing Mechanics and Energetics, Ibn Tofail University, Kenitra, Morocco
hamad.dahou@gmail.com

Hlou Laamari
Laboratory of Electronic Systems Information Processing Mechanics and Energetics, Ibn Tofail University, Kenitra, Morocco
hloul@yahoo.com

Abdelkader Hadjoudja
Laboratory of Electronic Systems Information Processing Mechanics and Energetics, Ibn Tofail University, Kenitra, Morocco
abdelkader.hadjoudja@uit.ac.ma

Received: 14 May 2025 | Revised: 30 June 2025 | Accepted: 6 July 2025

 DOI: https://doi.org/10.48084/etasr.12138

ABSTRACT

This paper presents a novel real-time vehicle detection system for autonomous vehicles, leveraging the advanced capabilities of the YOLOv11 deep neural network. The study addresses key challenges, including varying lighting conditions, occlusions, and the need to detect both traditional vehicle classes (cars, buses,

trucks, motorcycles) and emerging ones, like e-scooters and emergency vehicles (fire engines, ambulances, and police vehicles). A custom dataset of 38,500 annotated images, encompassing diverse real-world traffic scenarios, was utilized to train and evaluate the model. The optimized YOLOv11 model was deployed on embedded platforms, specifically the NVIDIA Jetson Nano and Raspberry Pi 5, achieving a balance between high detection accuracy and low resource consumption. The system achieved a precision of 98%, recall of 95%, and an F1-score of 96.5%, with a mAP@50 of 98.1%. The hardware-specific optimizations improved the inference speed and efficiency. The results demonstrate that YOLOv11, combined with lightweight embedded systems, offers a scalable, real-time solution for integration into Advanced Driver Assistance Systems (ADAS) and autonomous vehicles, ensuring safer and more reliable navigation.

Keywords-vehicle detection; ADAS; autonomous vehicles; deep neural network; YOLOv11; NVIDIA Jetson Nano; Raspberry Pi 5

I. INTRODUCTION

Autonomous vehicles represent a pivotal shift in modern transportation, promising fewer accidents, a smoother traffic flow, and enhanced passenger comfort. These gains depend on ADAS, which require real-time object perception and classification to support split-second decisions [1]. Deep learning, especially the You Only Look Once (YOLO) family, has become central to this task, offering high accuracy and rapid inference that can run on cost-sensitive edge devices, such as the NVIDIA Jetson Nano and Raspberry Pi 5 [2, 3]. Early Convolutional Neural Network (CNN) approaches demonstrated the value of the learned features for vehicle perception. Authors in [4] combined radial-basis-function networks with hand-crafted cues (shadows, taillights) to locate vehicles, while authors in [5] showed that the lightweight YOLOv3 and YOLOv5 architectures satisfy the latency constraints of pedestrian and vehicle detection. Yet, the real-world deployment remains challenging: illumination changes, weather, occlusions, and new road users, such as e-scooters or emergency vehicles, can degrade accuracy. Authors in [6] found YOLOv5 more accurate than YOLOv3 and YOLOv4, but highlighted its higher computational cost; Authors in [7], likewise, reported strong mAP scores across the YOLO variants, emphasizing the need to match the architecture to the embedded resources.

Edge platforms introduce further challenges. Authors in [8, 9] optimized YOLO on Jetson Nano for traffic-sign and pedestrian tasks, demonstrating that platform-specific tuning, such as TensorRT acceleration, can trim latency dramatically [10]. Authors in [11] added containerized deployment workflows, simplifying the updates in dynamic environments. Other studies refined YOLO for regional or resource-limited contexts: Authors in [12] adapted YOLOv7 with k-means++ anchor clustering for Moroccan roads. Authors in [13] reduced the compute load through frame selection, and authors in [14] pruned YOLOv7-tiny to boost the speed on Jetson modules. Beyond object detection, the YOLO variants have been extended to diverse safety applications. Authors in [15] fused audio and vision with YOLOv8 to spot ambulances near the traffic lights. Authors in [16] tailored YOLOv8 for driver-state monitoring, and authors in [17] compared YOLOv8 against YOLOv10, highlighting the trade-offs in precision and recall. These efforts underline YOLO's flexibility and its importance to ADAS.

YOLOv11 builds on this lineage with architectural upgrades aimed at greater accuracy and resilience, particularly for small or partially occluded objects. Authors in [18] verified YOLOv11's superiority in congested, high-speed scenarios, making it an excellent candidate for real-time vehicle perception. Its innovations include a C3k2 multibranch block for richer features, an SPPF layer for global-context pooling, and a pixel-level attention module that sharpens the focus on salient regions all while maintaining a lightweight footprint conducive to edge deployment.

The current work aims to design and deploy a real-time vehicle-detection pipeline based on YOLOv11, finely tuned for low-power edge devices, specifically the NVIDIA Jetson Nano and Raspberry Pi 5. The system is trained to recognize both conventional vehicles (cars, buses, trucks, motorcycles) and newer road users, such as e-scooters and emergency responders. High accuracy and minimal latency are achieved through device-specific optimizations, extensive data augmentation, and carefully selected hyperparameters. Demonstrating that advanced detection models can run effectively on lightweight hardware, the study paves the way for cost-efficient, reliable ADAS solutions, and thereby broadens access to safer autonomous driving technologies.

II. MATERIALS AND METHODS

A real-time vehicle detection system for autonomous driving is presented using YOLOv11. The proposed system is trained on a custom dataset of 38,500 annotated images encompassing both traditional and emerging vehicle classes [20]. Data preparation, model training, and deployment on the edge devices Jetson Nano and Raspberry Pi 5 are detailed and the evaluation metrics and experimental setup are outlined.

A. YOLOv11 Deep Learning Model

YOLOv11 marks a significant evolution in the Ultralytics YOLO series, offering a high-performing balance between the detection accuracy, inference speed, and computational efficiency - key requirements for real-time vehicle detection in ADAS. Designed to meet the demands of embedded edge platforms, YOLOv11 introduces architectural enhancements that improve the feature representation while minimizing the resource consumption [19, 20].

At its core, YOLOv11 replaces the traditional C2f module with the more advanced C3k2 block, a lightweight multibranch structure that includes residual connections for enhanced multiscale feature extraction. This modification enhances the model's adaptability to varying vehicle sizes and shapes without compromising the inference speed. The integration of the Spatial Pyramid Pooling Fast (SPPF) module allows the

network to capture both local and global semantic information through stacked MaxPooling operations, essential for detecting vehicles in complex traffic environments. While YOLOv11 achieves excellent results for certain small object classes, its general performance on small object detection remains challenging, particularly for objects that are less visually distinct or more occluded in traffic scenes. This limitation stems from the inherent trade-off in deep architectures, where spatial resolution diminishes in deeper layers, affecting the detection of small, low-contrast targets.

B. Data Preparation and Embedded Deployment

A structured workflow was followed in this study, as illustrated in Figure 1, comprising three main stages: dataset preparation, model training, and embedded deployment. A custom vehicle detection dataset of 38,500 manually annotated images was created and utilized in our previous work [20] to train and evaluate the YOLOv11 deep learning model. The images were captured across real-world traffic scenarios, including urban streets, highways, intersections, and parking lots under varied conditions, such as daylight, nighttime, fog, rain, and partial occlusions, ensuring robustness. The dataset includes seven vehicle classes: E-Scooter, emergency vehicle, bicycle, bus, car, motorcycle, and truck.

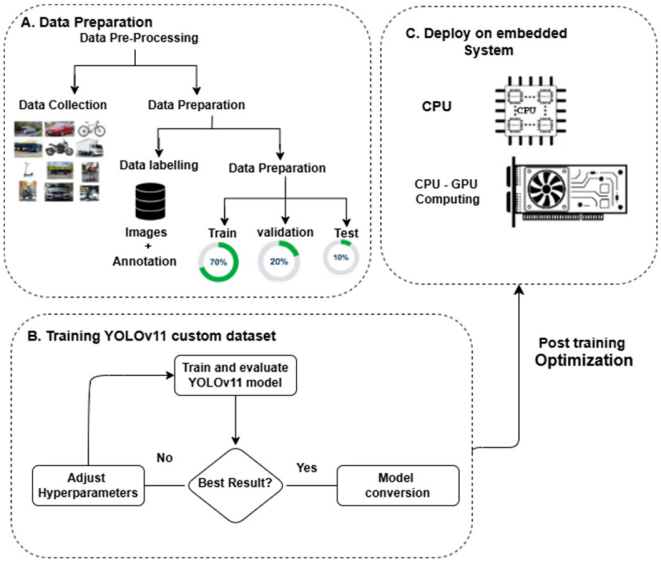

Fig. 1. Model training and deployment workflow: (a) dataset collection, (b) training, and (c) deployment on Jetson Nano and Raspberry Pi 5.

Images were collected using high-resolution cameras mounted on vehicles and roadside units. Additional samples were sourced from public datasets. All images were labeled using the Roboflow platform and exported in YOLO format. To improve generalization, data augmentation techniques - horizontal flips, rotations, noise injection, and exposure adjustments - were applied. The dataset was divided into 70% training, 20% validation, and 10% testing. Training was conducted on Google Colab using high-performance GPUs for faster convergence.

In [20], the trained model was compared with a YOLOv12 model. The focus was strictly on algorithmic and architectural improvements (e.g., R-ELAN and attention benefits), without any hardware implementation, embedded deployment, or system-level evaluation. Further, the same system was trained on a smaller dataset of 30,000 images and was presented in [22], again without any hardware experiments. This paper builds directly upon the YOLOv11 model and dataset also used in [20] but extends the research in new directions. Specifically, this manuscript focuses on embedded hardware deployment and ADAS-oriented evaluation, with the following key contributions:

- Implementation and optimization of YOLOv11 on NVIDIA Jetson Nano and Raspberry Pi 5.
- All experiments were re-generated under hardware-specific conditions.
- A comprehensive system-level evaluation, including detailed analysis of latency, FPS, and power consumption, was conducted.
- The feasibility for real-time ADAS integration is evaluated.

The trained YOLOv11 model was deployed on two embedded platforms: the NVIDIA Jetson Nano and Raspberry Pi 5, both suitable for ADAS applications. The Jetson Nano features a 1.43 GHz quad-core ARM Cortex-A57 CPU and a 128-core Maxwell GPU, running Ubuntu 20.04 with JetPack SDK v4.6 (CUDA, cuDNN, TensorRT) for accelerated inference. The Raspberry Pi 5, with a 2.4 GHz quad-core Cortex-A76 and 8 GB RAM, offered a more cost-efficient solution. Inference was performed using ONNX Runtime, and image preprocessing was handled via OpenCV.

Figure 2 depicts the experimental setup, showing the integration of cameras, HDMI USB adapters, and display hardware. The results confirmed that both platforms supported efficient, real-time vehicle detection. The Jetson Nano demonstrated superior inference speed and computational capability, while the Raspberry Pi 5 provided an energy-efficient solution ideal for lightweight, cost-sensitive ITS and ADAS deployments.

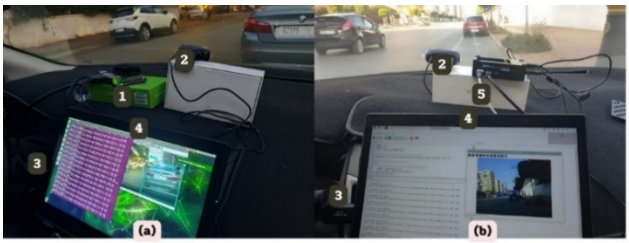

Fig. 2. Experimental setup for embedded deployment: (a) Jetson Nano, (b) Raspberry Pi 5. Components — 1: Jetson Nano, 2: camera, 3: HDMI to USB adapter, 4: Lenovo monitor, 5: Raspberry Pi 5.

C. Performance Metrics

To measure the performance of the vehicle detection algorithm, precision, recall, Average Precision (AP), and mean Average Precision (mAP) were employed as the primary

evaluation metrics. These metrics are crucial for evaluating the model's ability to accurately detect and classify objects across multiple categories while minimizing false detections. They are critical for assessing both the accuracy and robustness of a real-time detection system [21, 22]. Specifically, precision measures the proportion of the correctly predicted positive instances among all predicted positive instances, while recall measures the proportion of the correctly predicted positive instances among all actual positive instances. AP is defined as the area of a single class's precision-recall curve, while mAP computes the average of AP across all classes to provide an overall performance score.

These metrics are defined by the following equations, where TP stands for True Positives, FP stands for False Positives, and FN stands for False Negatives:

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \times 100\% \quad (1)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \times 100\% \quad (2)$$

$$\text{AP} = \int_0^1 \text{P(R)dR} \quad (3)$$

$$\text{mAP} = \frac{1}{\text{C}} \sum_{\text{j=1}}^{\text{c}} AP_j \quad (4)$$

$$\text{F 1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision+Recall}} \quad (5)$$

III. RESULTS AND DISCUSSION

The experimental results of the YOLOv11 model, as presented in Table I [20], highlight its strong performance during both training and validation. Trained in a PyTorch environment with NVIDIA GPUs over 100 epochs, the model used an SGD optimizer with a 0.01 learning rate, 0.9 momentum, and L2 regularization to prevent overfitting. A batch size of 8 balanced the efficiency and accuracy. As shown in Table II, YOLOv11 achieved 98% precision, 95% recall, 96.5% F1-score, 98.1% mAP@50, and 88.1% mAP@50-95, confirming its robustness and effectiveness for vehicle detection across diverse traffic conditions.

TABLE I. PERFORMANCE METRICS OF THE PROPOSED YOLOV11 MODEL

Model	Precision	Recall	F1-score	mAP@50	mAP@50-95
YOLOv11	98%	95%	96.5%	98.1%	88.1%

The YOLOv11 model achieved a peak F1 score of 0.965 at a confidence threshold of 0.671. This indicates a strong balance between precision and recall, with the model effectively minimizing the FP and FN across all seven vehicle classes, including more challenging ones, like e-scooters and emergency vehicles. The F1 score steadily rises with increasing confidence levels, peaking between 0.4 and 0.7 before tapering slightly, showcasing the model's stability under varying prediction thresholds. Vehicles, such as trucks, bicycles, and emergency vehicles, recorded particularly high F1 scores, reflecting the model's enhanced ability to detect complex and less frequent classes [20]. YOLOv11 achieved an overall mAP of 0.980 at IoU=0.5. The e-scooter class reached the highest accuracy (0.994), followed by trucks and motorcycles (0.991) [20]. The model maintained high precision across the recall levels, confirming its reliability in dynamic traffic scenarios. Figure 3 illustrates the detection outcomes under various real-world conditions, including urban streets, highways, and diverse lighting and weather. YOLOv11 consistently detected multiple vehicle types with high confidence, demonstrating strong robustness against occlusions, varied orientations, overlapping objects, and low visibility, making it well-suited for dynamic autonomous driving environments.

Even smaller or less frequently encountered vehicle classes, such as e-scooters, were accurately detected, reflecting the model's ability to generalize effectively across diverse and dynamic traffic conditions. The results further validate YOLOv11's ability to maintain reliable performance despite the environmental variability, making it highly suitable for real-time implementations in ADAS and intelligent transportation applications. By providing rapid, accurate detections, YOLOv11 enhances the overall safety and efficiency of autonomous driving systems, ensuring reliable object recognition and decision-making in critical scenarios.

Fig. 3. Results of performed by YOLOv11 networks.

The power consumption measurements, as summarized in Table II, show that the Raspberry Pi 5 operates with a lower energy draw ranging from 3.16 W (idle) to 6.8 W (execution), while the Jetson Nano requires an energy draw between 3.56 W (idle) and 10.2 W (execution). This highlights the efficiency versus performance trade-off inherent in embedded platforms.

TABLE II. COMPARATIVE POWER CONSUMPTION OF RASPBERRY PI 5 AND JETSON NANO

Platform	Mode	Voltage	Current	Power
Raspberry PI 5	Idle	5.4 V	0,58 A	3.16 W
	Execution	5.4 V	1.25 A	6.8 W
Jeston Nano	Idle	5.4 V	0.66 A	3.56 W
	Execution	5.4 V	1.9 A	10.2 W

Table III summarizes the comparative specifications to guide the system designers in selecting the optimal platform based on specific constraints.

TABLE III. HARDWARE AND PERFORMANCE COMPARISON OF EVALUATED PLATFORMS

Subsystem	Personal computer	Raspberry Pi 5	NVIDIA Jetson Nano
CPU	AMD Ryzen 9 7940HX	Broadcom BCM2712, quad-core (4x Arm Cortex-A76), 2.4GHz	Quad-core ARM Cortex-A57
GPU	NVIDIA GeForce RTX4070	VideoCore VII GPU, supporting OpenGL ES 3.1, Vulkan 1.2	128-Core Maxwell GPU with CUDA Core
Default storage	1 TB SSD high-speed PCIe interface (NVMe)	64 GB microSD	128 GB eMMC 5.1 Not Included (Dev-Kit)
System memory	32 GB DDR5	8GB RAM 64-bit	4 GB 64-bit LPDDR4
Camera interface	Webcam Full HD 1080p	2 × 4-lane MIPI camera/display transceivers	2-lane MIPI CSI-2 (1.5 Gbps per lane)
Typical power draw	115 W	6.8 W	10.2 W
Processing time	11.15ms	400ms	130 ms
Frame per s	89.7 FPS	2.5 FPS	7.7 FPS

The Raspberry Pi 5 demonstrates lower execution power consumption than the Jetson Nano (6.8 W versus 10.2 W), making it better suited for energy-efficient applications. However, this energy efficiency comes at the cost of performance: the Pi 5 delivers a frame rate of only 2.5 FPS and a processing time of 400 ms, limiting its suitability for high-speed, real-time detection tasks. In contrast, the Jetson Nano strikes a more favorable balance between performance and efficiency, offering a faster 130 ms processing time and 7.7 FPS, which is more appropriate for real-time object detection in autonomous systems. While the Jetson Nano is the better choice for latency-critical applications, the Raspberry Pi 5 remains a valuable option for low-power or cost-sensitive deployments.

Previous works have explored various YOLO-based optimizations to meet the constraints of embedded platforms. For example, YOLO-edge [23], built upon YOLOv5s, incorporated architectural enhancements, such as a slim neck and optimized spatial pyramid pooling, to lower the computational overhead. While it achieved impressive inference speeds (34 FPS on Jetson TX2 and 47 FPS on Orin NX), its accuracy improvements were modest (~92% mAP), and the targeted hardware was significantly more powerful than Raspberry Pi or Jetson Nano. In [24], YOLOv8 Detect and SSD-MobileNetV2 were employed for classifying Bangladeshi vehicles, attaining a 93% detection rate and 98% classification accuracy on Jetson Nano. However, integration challenges and performance limitations under variable conditions were reported. Authors in [25] implemented YOLOv3 for a smart traffic control system using Raspberry Pi and later Jetson Nano, but the older model yielded suboptimal inference speed and detection accuracy, particularly in complex traffic scenarios. Similarly, YOLOv9 was used in [26] for helmet detection on Raspberry Pi, showing high precision (0.894), recall (0.943), and mAP@50 (0.967). However, the system was constrained to binary classification and was not designed for scalable multi-class detection in diverse ADAS contexts.

On ther other hand, the proposed system based on YOLOv11 demonstrates superior adaptability to embedded environments. It integrates spatial attention mechanisms (e.g., C3k2 blocks and pixel attention), an optimized SPPF module, and lightweight, yet, robust architectural designs. These contribute to achieving high accuracy (98.1% mAP@50), excellent classification balance (96.5% F1-score), and operational real-time performance (7.7 FPS on Jetson Nano and 2.5 FPS on Raspberry Pi 5), all while maintaining efficient power consumption. This makes YOLOv11 particularly well-suited for multi-class vehicle detection in realistic ADAS scenarios. This study's findings affirm that YOLOv11 not only outperforms previous YOLO variants in both detection quality and execution efficiency, but also bridges the critical gap between the research-grade models and real-world deployment on low-power embedded platforms.

IV. CONCLUSION

This study proposes a robust real-time vehicle-detection system for autonomous driving, built on the YOLOv11 deep neural network architecture. The results of the current study demonstrate that YOLOv11, when carefully optimized, provides dependable, lightweight detection suitable for intelligent transportation systems. Future work will explore boosting the accuracy in edge-case scenarios, integrating additional sensor modalities, and refining the energy management, thereby creating a scalable, safe, and efficient autonomous-vehicle technology.

REFERENCES

[1] J. Maguire-Day, S. Al-Rubaye, A. Warrier, M. A. Sen, H. Whitworth, and M. Samie, "Emerging Decision-Making for Transportation Safety: Collaborative Agent Performance Analysis," *Vehicles*, vol. 7, no. 1, Mar. 2025, Art. no. 4, https://doi.org/10.3390/vehicles7010004.

[2] M. L. Ali and Z. Zhang, "The YOLO Framework: A Comprehensive Review of Evolution, Applications, and Benchmarks in Object Detection," *Computers*, vol. 13, no. 12, Dec. 2024, Art. no. 336, https://doi.org/10.3390/computers13120336.

[3] A. Setyanto *et al.*, "Knowledge Distillation in Object Detection for Resource-Constrained Edge Computing," *IEEE Access*, vol. 13, pp. 18200–18214, 2025, https://doi.org/10.1109/ACCESS.2025.3534020.

[4] X. Chen, H. Chen, and H. Xu, "Vehicle Detection Based on Multifeature Extraction and Recognition Adopting RBF Neural Network on ADAS System," *Complexity*, vol. 2020, no. 1, 2020, Art. no. 8842297, https://doi.org/10.1155/2020/8842297.

[5] C. Neelam Jaikishore et al., "Implementation of Deep Learning Algorithm on a Custom Dataset for Advanced Driver Assistance Systems Applications," *Applied Sciences*, vol. 12, no. 18, Jan. 2022, Art. no. 8927, https://doi.org/10.3390/app12188927.

[6] G. S. A. Mohammed, N. M. Diah, Z. Ibrahim, and N. Jamil, "Vehicle detection and classification using three variations of you only look once algorithm," *International Journal of Reconfigurable and Embedded Systems*, vol. 12, no. 3, pp. 442–452, Nov. 2023, https://doi.org/10.11591/ijres.v12.i3.pp442-452.

[7] N. R. Kolukula, R. P. Kalapala, S. S. R. Ivaturi, R. K. Tammineni, M. Annavarapu, and U. Pyla, "An efficient object detection by autonomous vehicle using deep learning," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 14, no. 4, pp. 4287–4295, Aug. 2024, https://doi.org/10.11591/ijece.v14i4.pp4287-4295.

[8] S. Wadhwa, P. Saini, R. Kumar, N. S. Kashyap, and T. Siag, "Real Time Object Detection of Aerial Images Using Deep Learning on Jetson

Nano," in *AIAA SCITECH 2025 Forum*, https://doi.org/10.2514/6.2025-1431.

[9] K. Sarvajcz, L. Ari, and J. Menyhart, "AI on the Road: NVIDIA Jetson Nano-Powered Computer Vision-Based System for Real-Time Pedestrian and Priority Sign Detection," *Applied Sciences*, vol. 14, no. 4, Jan. 2024, Art. no. 1440, https://doi.org/10.3390/app14041440.

[10] D.-J. Shin and J.-J. Kim, "A Deep Learning Framework Performance Evaluation to Use YOLO in Nvidia Jetson Platform," *Applied Sciences*, vol. 12, no. 8, Jan. 2022, Art. no. 3734, https://doi.org/10.3390/app12083734.

[11] W.-C. Shih *et al.*, "The Construction of a Stream Service Application with DeepStream and Simple Realtime Server Using Containerization for Edge Computing," *Sensors*, vol. 25, no. 1, Jan. 2025, Art. no. 259, https://doi.org/10.3390/s25010259.

[12] I. Taouqi, A. Klilou, K. Chaji, and A. Arsalane, "Traffic signs detection and prohibitor signs recognition in Morocco road scene," *International Journal of Electrical and Computer Engineering*, vol. 14, no. 6, pp. 6313–6321, Dec. 2024, https://doi.org/10.11591/ijece.v14i6.pp6313-6321.

[13] V. N. Ribeiro and N. S. T. Hirata, "Combining YOLO and Visual Rhythm for Vehicle Counting," in *Anais Estendidos da XXXVI Conference on Graphics, Patterns and Images (SIBRAPI Estendido 2023)*, Nov. 2023, pp. 164–167, https://doi.org/10.5753/sibgrapi.est.2023.27473.

[14] Z. Liang *et al.*, "Vehicle and Pedestrian Detection Based on Improved YOLOv7-Tiny," *Electronics*, vol. 13, no. 20, Jan. 2024, Art. no. 4010, https://doi.org/10.3390/electronics13204010.

[15] A. Mecocci and C. Grassi, "RTAIAED: A Real-Time Ambulance in an Emergency Detector with a Pyramidal Part-Based Model Composed of MFCCs and YOLOv8," *Sensors*, vol. 24, no. 7, Jan. 2024, Art. no. 2321, https://doi.org/10.3390/s24072321.

[16] M. Al-Mahbashi, G. Li, Y. Peng, M. Al-Soswa, and A. Debsi, "Real-Time Distracted Driving Detection Based on GM-YOLOv8 on Embedded Systems," *Journal of Transportation Engineering, Part A: Systems*, vol. 151, no. 3, Mar. 2025, Art. no. 04024126, https://doi.org/10.1061/JTEPBS.TEENG-8681.

[17] A. Sundaresan Geetha, M. A. R. Alif, M. Hussain, and P. Allen, "Comparative Analysis of YOLOv8 and YOLOv10 in Vehicle Detection: Performance Metrics and Model Efficacy," *Vehicles*, vol. 6, no. 3, pp. 1364–1382, Sept. 2024, https://doi.org/10.3390/vehicles6030065.

[18] M. A. R. Alif, "YOLOv11 for Vehicle Detection: Advancements, Performance, and Applications in Intelligent Transportation Systems." *arXiv*, Oct. 30, 2024, https://doi.org/10.48550/arXiv.2410.22898.

[19] H. M. Zayani *et al.*, "Cat Breed Classification with YOLOv11 and Optimized Training," *Engineering, Technology & Applied Science Research*, vol. 15, no. 2, pp. 21652–21657, Apr. 2025, https://doi.org/10.48084/etasr.10218.

[20] M. Chaman, A. El Maliki, H. El Yanboiy, H. Dahou, H. Laâmari, and A. Hadjoudja, "Comparative Analysis of Deep Neural Networks YOLOv11 and YOLOv12 for Real-Time Vehicle Detection in Autonomous Vehicles," International Journal of Transport Development and Integration, vol. 9, no. 1, pp. 39–48, Mar. 2025, https://doi.org/10.18280/ijtdi.090104.

[21] A. Tripathi, V. Gohokar, and R. Kute, "Comparative Analysis of YOLOv8 and YOLOv9 Models for Real-Time Plant Disease Detection in Hydroponics," *Engineering, Technology & Applied Science Research*, vol. 14, no. 5, pp. 17269–17275, Oct. 2024, https://doi.org/10.48084/etasr.8301.

[22] M. Chaman, A. El Maliki, N. Jariri, H. Dahou, H. Laâmari, and A. Hadjoudja, "Enhanced Deep Neural Network-Based Vehicle Detection System Using YOLOv11 for Autonomous Vehicles," in *2025 5th International Conference on Innovative Research in Applied Science, Engineering and Technology*, May 2025, pp. 1–6, https://doi.org/10.1109/IRASET64571.2025.11008084.

[23] X. Li, L. Chen, T. Huang, A. Yang, and W. Liu, "YOLO-edge: real-time vehicle detection for edge devices," *Cluster Computing*, vol. 28, no. 5, Apr. 2025, Art. no. 289, https://doi.org/10.1007/s10586-024-04963-w.

[24] Farid, P. Kumer Das, M. Islam, and E. Sina, "Bangladeshi Vehicle Classification and Detection Using Deep Convolutional Neural Networks With Transfer Learning," *IEEE Access*, vol. 13, pp. 26429–26455, 2025, https://doi.org/10.1109/ACCESS.2025.3539713.

[25] S. K. Lokeshwaran, N. S. Lohith, and K. Venkatasubramanian, "Intelligent Traffic Management System Using Real-Time Vehicle Detection and Dynamic Signal Control with Client-Server Architecture," in *2025 International Conference on Next Generation Communication & Information Processing (INCIP)*, Jan. 2025, pp. 582–586, https://doi.org/10.1109/INCIP64058.2025.11019986.

[26] N. Tahilramani, P. Ahir, S. Saxena, V. P. Talreja, and P. Charanarur, "Edge-based AI solution for enhancing urban safety: helmet compliance monitoring with YOLOv9 on Raspberry Pi," *Discover Internet of Things*, vol. 5, no. 1, Mar. 2025, Art. no. 25, https://doi.org/10.1007/s43926-025-00113-9.