p-ISSN : 2301-8402, e-ISSN : 2685-368X, available at : https://ejournal.unsrat.ac.id/index.php/elekdankom

Real-time Vehicle Detection using YOLOv10-nano Algorithm for Monitoring Traffic Analysis

Imanuel Kutika, Vicky Lahimade, Tomi Todingan, Steven Ray Sentinuwo, Muhamad Dwisnanto Putro,
Master Program of Informatics, Postgraduate Program, Sam Ratulangi University, Manado, North Sulawesi, Indonesia
e-mails: imanuelkutika026@student.unsrat.ac.id, vickylahimade026@student.unsrat.ac.id,
tomitodingan026@student.unsrat.ac.id, steven@unsrat.ac.id, dwisnantoputro@unsrat.ac.id.
Received: 3 May 2025; revised: 6 June 2025; accepted: 14 July 2025

***Abstract* — Vehicle detection becomes crucial in the automation of traffic analysis, enabling assignment for instance vehicle categorization, speed estimation, and traffic flow monitoring. While vision-based systems are essential for recognizing vehicle types and sizes, improving detection performance and efficiency remains a significant challenge. This work aiming to evaluate how effective the YOLOv10n for real-time vehicle detection in resource-limited context. YOLOv10n, the most efficient version of the YOLO iteration to date, offers remarkable advancements in feature extraction and computational efficiency. Using the Vehicle-COCO dataset, which reflects real-world surveillance traffic conditions, the proposed framework attained score 0.637 at mAP@50 of and 0.442 at mAP50:95, demonstrating its capability to accurately detect various vehicle types. Furthermore, the model runs at 23.08 frames per second on a standard CPU, underscoring its suitability for edge deployment on low-cost devices. The findings confirm that YOLOv10n combines high efficiency with competitive accuracy, making it a viable solution for intelligent transportation systems. This paper also outlines potential directions for future research, including accuracy improvement, parameter tuning, and further optimization of computational resources.**
***Key words* — Deep Learning, Vehicle Detection, YOLOV10n, Efficient Model**

I. INTRODUCTION

Vehicles are a means of mobility for people in their daily lives both in rural areas and especially in urban areas, with types of road vehicles, including cars, buses, and trucks that occupy urban roads [1]. This condition increases traffic congestion on certain highway sections experiencing rising congestion from yearly traffic growth [2]. In 2024, the global average time lost due to traffic congestion was approximately 88 hours per person per year [3]. This underscores the urgent need for effective vehicle detection systems to support real-time traffic monitoring and management. Traffic analysis is essential for understanding road usage patterns by quantifying the number of vehicles utilizing a roadway over a given time period [4]. This information is critical for identifying periods of peak congestion. Additionally, traffic analysis contributes to evaluating traffic efficiency by measuring key parameters such as vehicle speed, flow rate, and density across various vehicle categories [5].

Recent advancements in computer vision, particularly deep learning-enabled object detection, have gave rise to the development of powerful models that balance detection accuracy and computational efficiency [6]. The YOLO architectures have continued to evolve through object optimization and model enhancement strategies, leading to the development of YOLOv10 [7]. Recent works on YOLOv10-B-version have 46% latency decrease and 25% lower parameter count for the same performance from YOLOv9-C [8]. It shows that YOLOv10 has achieved performance and efficiency levels that surpass previous generation models. Based on that, YOLOv10 nano version becomes a highly efficient model that demonstrates object detection with low latency and decent accuracy.

Several works that implemented YOLOv10 architecture on object detection like using it on efficient fabric classification and detection [9]. YOLOv10-GCNet on real time detection of counting wheat spikes, monitoring crops growth along with predicting yield and managing fields [10]. YOLO-YSTs to developed a real-time field pest detection, using SPD-Conv and inverted residual block elevating precision of YOLOv10n [11]. Another works has used the YOLO architecture especially for vehicle detection purposes such as YOLOv3 [12] and YOLOv5 [13]. Another works on vehicle detection using Faster-RCNN [14] and CO-DETR that introducing multiple parallel auxiliary heads to boost training efficiency even with large computational cost [15]. However, the performance of the YOLOv10 nano (YOLOv10n) remains underexplored in the scope of vehicle detection.

This work objective is to measure and estimate the baseline output of the YOLOv10 nano architecture. It assesses how effective identification of vehicle-related objects. The core advancement of this research approach outlined below:

1. This work evaluates YOLOv10n performances without the integration of additional attention mechanisms or architectural enhancements, providing a foundational benchmark for future developments.
2. This work evaluates YOLOv10n in the scope of vehicle detection, which is vital for applications including vehicle tracking, speed estimation, and traffic flow analysis [16], [17].
3. This research reveals the feasibility of deploying YOLOv10n on standard CPUs, demonstrating its potential for real-time use on resource-constrained edge devices.

II. METHOD

The YOLOv10n architecture represents a lightweight deep learning framework specifically developed for object detection tasks, with a specific focus on vehicle detection in road environments. It is capable of identifying, distinguishing, and classifying various types of vehicles in real time. Structurally,

this framework is organized grouped into three essential modules. First, backbone functions as the architecture's starting point, extracting essential visual traits like edges, shapes, and patterns based on the data feed. These features are transformed into a structured feature representation that can be further processed. Second, neck acts as an intermediate layer that fuses multi-scale feature maps from Backbone, ensuring proper alignment and resolution compatibility between Backbone and Head. Finally, the Head handles object classification and for estimating the probability scores associated with each detected class. YOLOv10n utilizes lightweight convolutional layers optimized for efficient feature extraction, rendering it relevant for utilization in under-resource context while preserving high detection accuracy.

A. Backbone

The backbone extract feature from input image, starting with Conv layer that consists of convolution process, batch normalization and SILU activation. Batch normalization used for getting a faster and stable training. It also improve learning rate and in some cases elminiating the need for dropout [18]. SILU activation also known as Swish function, is a type of activation capable to address problems better if neurons stuck at zero. It can function as a gate means for negative input, SILU can behave similarly to a linear function but with smooth transition and not exactly zero [19].

Next layer is C2f block, its name originates from CSP bottleneck with two convolutions. The process starts from convolution 1x1, split by multiplying expansion factor. All result from each operation is concatenated and ends with convolution 1x1 to mix the feature for processing through the next layer. Some parts go through continues between the Conv and C2f Layers and some features bypass it bring computational efficiency. C2f support better gradient propagation during training, stabilize it in case on deeper networks. C2f layer also preserve important information while enabling deeper representation with lower computational cost. C2f block ended with convolution 1x1 to mix the features through to next layer.

The Spatial Pyramid Pooling Fast (SPPF) layer becomes fundamental capacity in broadening this model's spatial context, thereby enhancing its ability to detect objects of varying scales. It starts through convolution 1x1, continues through a sequence involving maximum pooling on two dimensional operations with window sizes between 5, 9, and 13 that aggregate contextual information across multiple spatial resolutions. The output feature maps from each 2D pooling window are combined with the result of the initial 1×1 convolution through concatenation. This approach aims to enhance the richness of feature representation and expand the receptive field, while keeping computational overhead low by utilizing three consecutive 2D max pooling operations instead of standard convolution layers.

Following the SPPF layer, the architecture incorporates a Partial Self-Attention (PSA) block to elevate the representation of global contextual features while maintaining computational efficiency. The PSA block integrates a Feed-Forward Network (FFN), a unidirectional network structure used to process multi-scale features. The FFN applies non-linear transformations and can serve as a final layer for classification or prediction tasks. Following the initial process, both branches proceed separately. This resulting feature maps are concatenated with those generated by the FFN branch. A final 1×1 convolution applies to mix the features, preparing the output for processing in the next block.

B. Neck

The neck module is essential for the object detection framework, aggregating as well as refining multi-scale features produced by the backbone. It typically incorporates a Path Aggregation Network (PAN), which enables the fusion of features throughout multiple spatial resolutions, thereby elevating its capability on object detection at diverse circumstances [8]. On that structure there are C2f Additionally, the neck includes an Upsample block that restores spatial resolution by upscaling feature maps, compensating for the detail loss incurred during downsampling. These components are integrated via a Concat block, which aligns feature dimensions and increases feature diversity, ultimately improving this architecture's capability to identify entities with varying scales and structural nuances.

C. Head

The head is responsible for object classification and localization, generating class probabilities and regressing bounding box coordinates for each detected object. YOLOv10 introduces a significant improvement over previous architectures by replacing the conventional Non-Maximum Suppression (NMS), which is often computationally expensive when dealing with a large number of detected boxes [8]. Instead, YOLOv10 adopts a novel approach called Consistent Dual Assignments. This mechanism integrates two key components: the One-to-Many head, which produce numerous estimations within the network to deliver dense supervisory signals during training, and the One-to-One head, that utilized a method of assigning labels by matching each other that produce one prediction for each ground-truth instance. This dual-head architecture improves the efficiency of the training process while also contributing to enhanced detection accuracy.

The head employs loss functions to quantify the gap within predictions and ground truth. Notably, the Complete IoU (CIoU) loss refines enclosing box regression consider of center distance, proportionality, and overlap [20]. This leads to more precise localization. YOLOv10 employs Distribution Focal Loss (DFL) that models bounding box regression as a distributional prediction problem, aligning the predicted distribution with a target derived from the ground truth. This approach strengthens the head's ability to generate precise bounding boxes and enables the model to produce a single optimal prediction for each object.

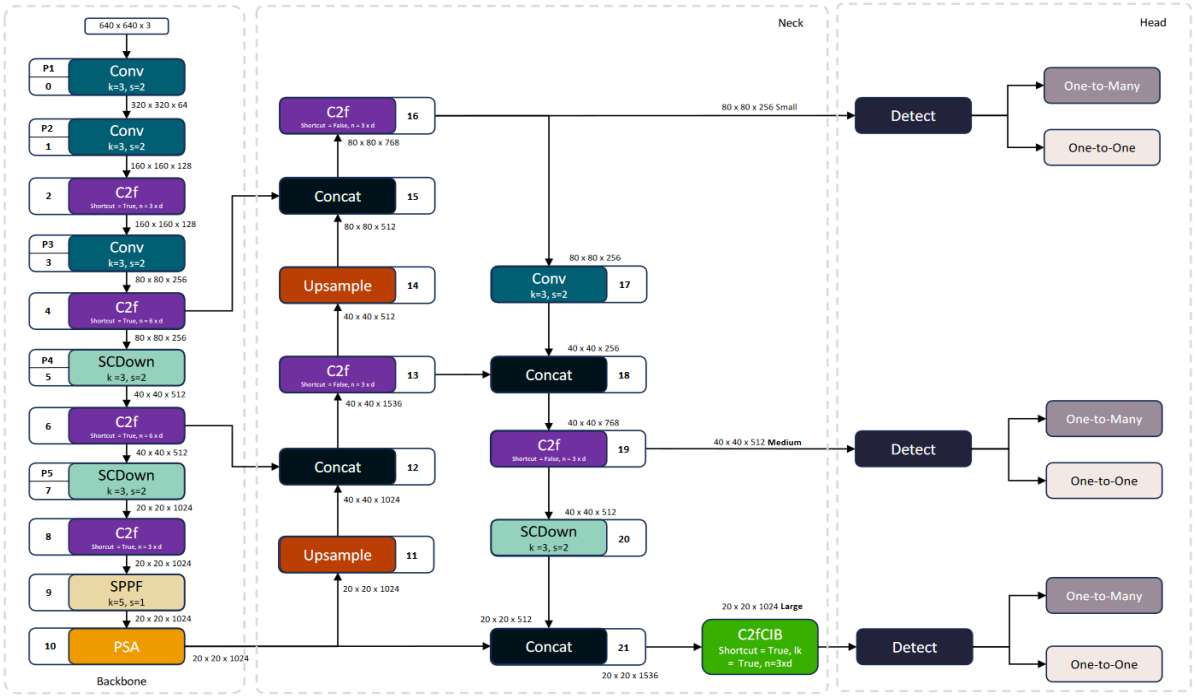

Figure 1. YOLOv10n Architecture for Vehicle Detection. It consists of Backbone, Neck and Head layer

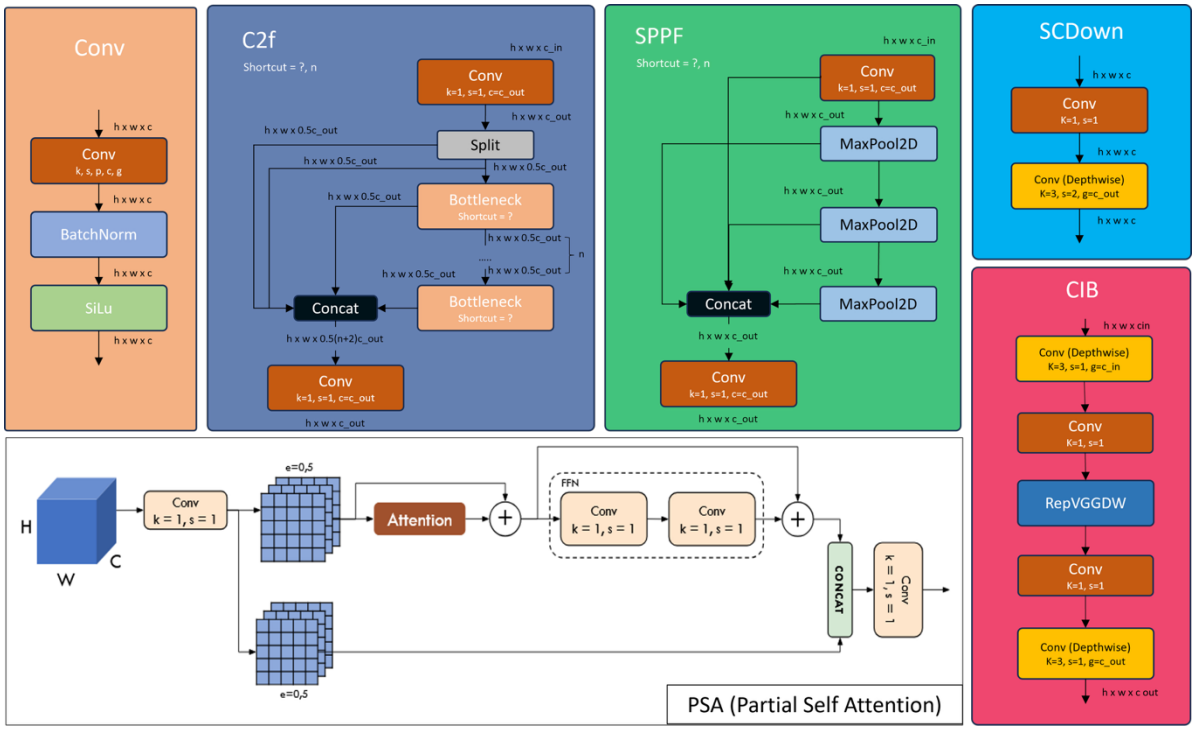

Figure 2. YOLOv10n blocks that used in the Backbone and Neck layer.

III. RESULT AND DISCUSSION

This segment outlines all evaluation data manifest the YOLOv10n baseline architecture in the context of vehicle detection, using the Vehicle-COCO dataset. The findings highlight the effectiveness of YOLOv10n in accurately detecting vehicles and differentiating between the various vehicle categories within the dataset. The model successfully detects several vehicle categories, including cars, motorcycles, buses, and trucks, under complex circumstances in particular: occlusion, low illumination environments, and varying weather factors. These results underscore this architecture's robustness and performance throughout various real-world scenarios, involving multiple vehicle categories.

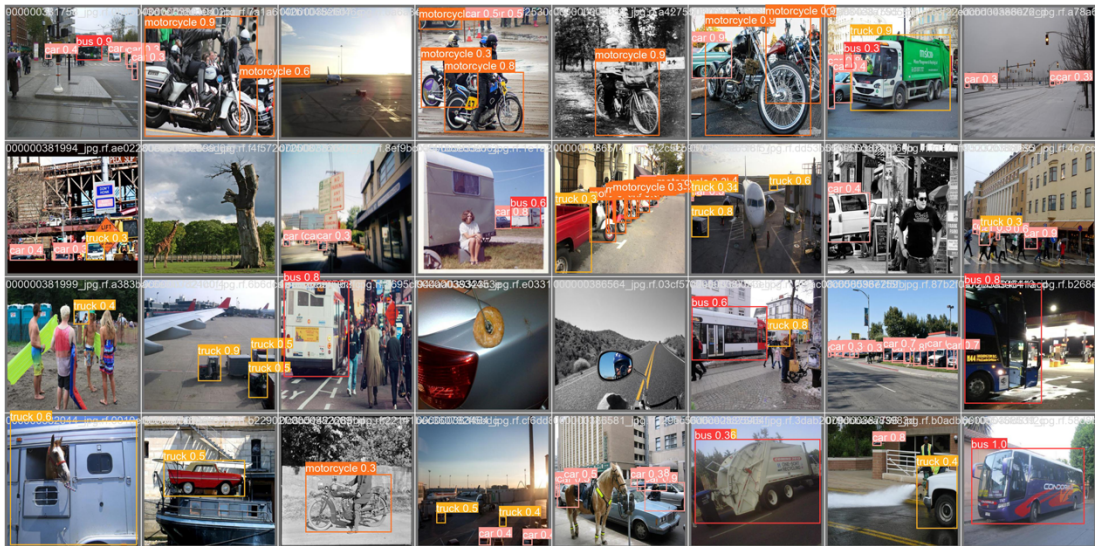

Figure 3. Vehicle detection result of YOLOV10n on Vehicle-COCO dataset

TABLE I
TRAINING SETUP CONFIGURATION

Metric	Value
Epoch	300
Batch size	16
Learning rate	0.01
Image size	640x640
Optimizer	Stochastic Gradient Descent (SGD)

TABLE II
INFERENCE CONFIGURATION

Components	Setup specification
Operating System	Ubuntu (Linux)
Framework	Pytorch 2.0
Python Version	3.9.20
CPU	AMD Ryzen 5 3500 6-core
GPU	Nvidia RTX 3060 12 GB

The proposed framework attained mAP@50 of 0.637 and also reached a mAP@50:95 of 0.442, which reflects its performance along a range of bounds. These results presenting a better comprehensive evaluation on model detection accuracy. In terms of computational efficiency, the model consists of 2.7 million parameters with requires 8.4 GFLOPs of computational load, making it efficient and ideal for deployment on resource-limited, low-cost edges. For real-time inference, the model achieves approximately 23 frames per second (FPS) using a CPU, featuring a delay around 0.05 seconds, demonstrating its ability to perform object detection at a practical speed in limited-edge environments.

A. Training Configuration

The following Table I shows the setup applied in YOLOv10n on vehicle detection experiment. The model trains on 300 epochs using a batch size of 16. Before processing, all image converted into 640x640 pixel to create a consistent input dimension and to ensure optimal performance. Stochastic Gradient Descent (SGD) employs as optimizer that known for effective large-scale training with purpose to find most favorable outcome.

Table II presents the training configuration, which was implemented on a system running the Ubuntu operating system (OS). Ubuntu, a Linux-based OS, is widely adopted in deep learning development due to its compatibility with frameworks such as PyTorch 2.0. The experiments utilized Python version 3.9.20, chosen for its stability and widespread support. Model training was conducted using an AMD Ryzen 5 3500 6-core CPU, with acceleration provided by an NVIDIA RTX 3060 GPU. The GPU, with its high parallel computing capabilities and 12 GB of memory, significantly reduced training time compared to CPU-only training. Its ample memory capacity also enabled efficient processing of complex models and large-scale datasets.

B. Evaluation of Dataset

The Vehicle-COCO dataset is a specialized division of the MS COCO 2017, developed specifically for vehicle object detection. It comprises approximately 18,998 images, with 13,300 images allocated for training and 3,788 for validation. Vehicle annotations are categorized into four classes: car, bus, truck, and motorcycle, encompassing a wide range of object sizes and environmental conditions.

Figure 3 illustrates the performance evaluation of YOLOv10n on the validation subset of the Vehicle-COCO dataset. Detected vehicle objects are highlighted with red bounding boxes. This visualization illustrates the capabilities towards accuracy when detect a range of vehicle types across varying sizes and scenes, reinforcing its adequacy to real-time vehicle recognition in diverse environments.

Despite its strong performance, the model exhibits limitations in specific scenarios, such as detecting overlapping vehicles or visually similar classes. For instance, large cars may occasionally be misclassified as trucks or buses due to subtle brand or structural variations, which can challenge the model's classification accuracy.

TABLE III
THE PERFORMANCE OF THE YOLOV10-NANO

Metric	Value
mAP 50	0.637
mAP 50:95	0.442

TABLE IV
THE EFFICIENCY ANALYSIS OF THE YOLOV10-NANO

Metric	Value
GFLOPS	8.4
Parameter	2.708.584
FPS (CPU)	23.08
Latency	0.05

Based on the results presented in Table III, YOLOv10n demonstrates competitive performance, achieving a mAP@50 of 0.637 and a mAP@50:90 of 0.442. These outcomes reveal about how the model maintains a balance amid precision along with model complexity. While operating under lightweight constraints, YOLOv10n delivers precision metrics comparable to larger models, reinforcing its suitability for deployment in real-time, resource-limited scenarios. This level of accuracy underscores how competence in identifying entities spanning various intersection-over-union (IoU) thresholds.

Table IV demonstrates that the proposed model remains lightweight, utilizing only 2.7 million parameters and requiring a low computational cost of 8.4 GFLOPs. This efficiency ensures it is well-adapted for deployment on edge hardware with limited resources. The model reaches 23.08 FPS in inference on a CPU, with a latency of just 0.05 seconds, indicating superior runtime performance compared to other lightweight models. These results highlight this network feasibility for actual scenarios, particularly in low-cost, resource-constrained environments where computational efficiency is critical. This exhibit that the model remains lightweight with only using 2.7 million parameters and require a low computational cost of 8.4 GFLOPs, making it suitable for edge deployment. It achieves 23.08 FPS on a CPU with a latency of 0.05 s, demonstrating higher efficiency than comparable lightweight models. These results confirm its viability for real-time applications in resource-limited environments.

IV. CONCLUSIONS

This work presents a baseline performance evaluation of YOLOv10n, a lightweight object detection model, in the context of vehicle detection using the Vehicle-COCO validation dataset. The evaluation was performed without incorporating additional architectural enhancements such as attention modules or advanced feature fusion techniques, providing a reference point for the model's raw capabilities.

YOLOv10n achieved a mAP@50 of 0.637 and operated at 23.08 FPS on a standard CPU, demonstrating its suitability for real-time traffic implementations such as traffic monitoring, violation detection, and automated ticketing. Its efficiency and low computational requirements make it notably viable for utilizations on resource-limited devices. However, performance limitations were noted in complex traffic scenes where overlapping vehicles or similar appearances led to occasional misclassification.

Upcoming researches might emphasize on elevating detection accuracy along the incorporation of multi-scale attention mechanisms and enhancements to the model's neck architecture. Optimizing the Path Aggregation Network (PAN) to improve feature integration throughout scales could enhance both localization and classification performance. Combining these improvements could induce to a robust and efficient model, proficient of operating real-time detection tasks in low-resource environments with greater precision.

ACKNOWLEDGMENTS

The authors emphasize their heartfelt thanks to the AIVISION research team for indispensable support along with collaboration throughout the duration of this study. The team's technical expertise, constructive discussions, and provision of computational resources were instrumental in the successful execution of the research. Their insightful feedback significantly contributed to the refinement of both the methodological framework and the final manuscript. The authors greatly appreciate their continued encouragement and contributions to this work.

REFERENCES

[1] A. Ceder, "Urban mobility and public transport: future perspectives and review," *Int. J. Urban Sci.*, vol. 25, no. 4, pp. 455–479, Oct. 2021, doi: 10.1080/12265934.2020.1799846.

[2] X. Chen *et al.*, "Quantifying on-road vehicle emissions during traffic congestion using updated emission factors of light-duty gasoline vehicles and real-world traffic monitoring big data," *Sci. Total Environ.*, vol. 847, p. 157581, Nov. 2022, doi: 10.1016/j.scitotenv.2022.157581.

[3] S. Xu, C. Sun, and N. Liu, "Road congestion and air pollution -Analysis of spatial and temporal congestion effects," *Sci. Total Environ.*, vol. 945, p. 173896, Oct. 2024, doi: 10.1016/j.scitotenv.2024.173896.

[4] R. Zhou, H. Chen, H. Chen, E. Liu, and S. Jiang, "Research on Traffic Situation Analysis for Urban Road Network Through Spatiotemporal Data Mining: A Case Study of Xi'an, China," *IEEE Access*, vol. 9, pp. 75553–75567, 2021, doi: 10.1109/ACCESS.2021.3082188.

[5] J. Zang, P. Jiao, S. Liu, X. Zhang, G. Song, and L. Yu, "Identifying Traffic Congestion Patterns of Urban Road Network Based on Traffic Performance Index," *Sustainability*, vol. 15, no. 2, Art. no. 2, Jan. 2023, doi: 10.3390/su15020948.

[6] T. Diwan, G. Anirudh, and J. V. Tembhurne, "Object detection using YOLO: challenges, architectural successors, datasets and applications," *Multimed. Tools Appl.*, vol. 82, no. 6, pp. 9243–9275, Mar. 2023, doi: 10.1007/s11042-022-13644-y.

[7] M. A. R. Alif and M. Hussain, "YOLOv1 to YOLOv10: A comprehensive review of YOLO variants and their application in the agricultural domain," Jun. 14, 2024, *arXiv*: arXiv:2406.10139. doi: 10.48550/arXiv.2406.10139.

[8] A. Wang *et al.*, "YOLOv10: Real-Time End-to-End Object Detection," *Adv. Neural Inf. Process. Syst.*, vol. 37, pp. 107984–108011, Dec. 2024.

[9] M. Mao, A. Lee, and M. Hong, "Efficient Fabric Classification and Object Detection Using YOLOv10," *Electronics*, vol. 13, no. 19, p. 3840, Sep. 2024, doi: 10.3390/electronics13193840.

[10] S. Guan *et al.*, "Real-Time Detection and Counting of Wheat Spikes Based on Improved YOLOv10," *Agronomy*, vol. 14, no. 9, Art. no. 9, Sep. 2024, doi: 10.3390/agronomy14091936.

[11] Y. Huang *et al.*, "YOLO-YSTs: An Improved YOLOv10n-Based Method for Real-Time Field Pest Detection," *Agronomy*, vol. 15, no. 3, Art. no. 3, Mar. 2025, doi: 10.3390/agronomy15030575.
[12] Y. Miao, F. Liu, T. Hou, L. Liu, and Y. Liu, "A Nighttime Vehicle Detection Method Based on YOLO v3," in *2020 Chinese Automation Congress (CAC)*, Nov. 2020, pp. 6617–6621. doi: 10.1109/CAC51589.2020.9326819.
[13] S. Cepni, M. E. Atik, and Z. Duran, "Vehicle Detection Using Different Deep Learning Algorithms from Image Sequence," *Balt. J. Mod. Comput.*, vol. 8, no. 2, 2020, doi: 10.22364/bjmc.2020.8.2.10.
[14] A. Chaudhuri, "Smart traffic management of vehicles using faster R-CNN based deep learning method," *Sci. Rep.*, vol. 14, no. 1, p. 10357, May 2024, doi: 10.1038/s41598-024-60596-4.
[15] I. A. Fahad, A. I. H. Arean, N. S. Ahmed, and M. Hasan, "Automatic Vehicle Detection using DETR: A Transformer-Based Approach for Navigating Treacherous Roads," Feb. 25, 2025, *arXiv*: arXiv:2502.17843. doi: 10.48550/arXiv.2502.17843.
[16] J. Azimjonov and A. Özmen, "A real-time vehicle detection and a novel vehicle tracking systems for estimating and monitoring traffic flow on highways," *Adv. Eng. Inform.*, vol. 50, p. 101393, Oct. 2021, doi: 10.1016/j.aei.2021.101393.
[17] C.-J. Lin, S.-Y. Jeng, and H.-W. Lioa, "A Real-Time Vehicle Counting, Speed Estimation, and Classification System Based on Virtual Detection Zone and YOLO," *Math. Probl. Eng.*, vol. 2021, no. 1, p. 1577614, 2021, doi: 10.1155/2021/1577614.
[18] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proceedings of the 32nd International Conference on Machine Learning*, PMLR, Jun. 2015, pp. 448–456. Accessed: Jun. 06, 2025. [Online]. Available: https://proceedings.mlr.press/v37/ioffe15.html
[19] A. Fatima and A. Pethe, "NVM Device-Based Deep Inference Architecture Using Self-gated Activation Functions (Swish)," in *Machine Vision and Augmented Intelligence—Theory and Applications*, M. K. Bajpai, K. Kumar Singh, and G. Giakos, Eds., Singapore: Springer, 2021, pp. 33–44. doi: 10.1007/978-981-16-5078-9_4.
[20] S. Du, B. Zhang, P. Zhang, and P. Xiang, "An Improved Bounding Box Regression Loss Function Based on CIOU Loss for Multi-scale Object Detection," in *2021 IEEE 2nd International Conference on Pattern Recognition and Machine Learning (PRML)*, Jul. 2021, pp. 92–98. doi: 10.1109/PRML52754.2021.9520717.

Imanuel Kutika was born in Manado, Indonesia, on June 30, 1992. He is currently pursuing his postgraduate studies in the Informatics Program at Sam Ratulangi University. He obtained undergraduate degree in Informatics Engineering from the Département of Engineering, De La Salle University, Manado. During his academic journey, he has been actively involved in various research initiatives and system development projects, particularly in the areas of mobile and web application, microcontroller embedded system and Internet of Things (IoT). Motivated by a strong interest in technology and intelligent systems, he continued his studies at the graduate level with a focus on artificial intelligence and computer vision. His current research, titled *"Real-Time Vehicle Detection Using the YOLOv10-Nano Algorithm for Monitoring Traffic Analysis,"* explores the deployment of lightweight deep learning models in live vehicle detection using hardware platforms with low resources. His broader research interests lie in the practical implementation of AI algorithms, especially in intelligent transportation and smart systems.

The author committed to promoting accessible technological innovation and aims to contribute to the advancement of digital technologies in Indonesia. He also aspires to inspire future engineers to create responsive, forward-thinking technological solutions. He can be contacted via email at imanuelkutika026@student.unsrat.ac.id for academic collaborations, research discussions, or technology-related inquiries.