Institut Riset dan Publikasi Indonesia (IRPI)
MALCOM: Indonesian Journal of Machine Learning and Computer Science
Journal Homepage: https://journal.irpi.or.id/index.php/malcom
Vol. 6 Iss. 2 April 2026, pp: 962-972
ISSN(P): 2797-2313 | ISSN(E): 2775-8575

Development and Optimization of YOLOv12 for Autonomous Vehicle Navigation Systems

Dian Ramadhani[1*], Muhammad Muttakin[2], Hidayat Hatta Irsyad[3], Edi Susilo[4], Rahmat Rizal Andi[5]

[1,2,3,4,5]Department of Informatics Engineering, Universitas Riau, Indonesia

E-Mail: [1]dianramadhani@lecturer.unri.ac.id, [2]muhammad.muttakin5094@student.unri.ac.id, [3]hidayat.hatta1397@student.unri.ac.id, edi.susilo@lecturer.unri.ac.id, rahmat.rizal@lecturer.unri.ac.id

Received Feb 11th 2026; Revised Apr 08th 2026; Accepted Apr 28th 2026; Available Online Apr 30th 2026
Corresponding Author: Dian Ramadhani

Abstract

This study develops and enhances a YOLOv12-based object detection model for autonomous vehicle perception on Indonesian highways, addressing limitations of earlier research that lacked realistic traffic scenarios and field validation. The Roboflow dataset contains 29 object classes, including vehicles, pedestrians, and traffic signs, with existing annotations. Preprocessing included data quality assessment, image resizing, dataset split validation, annotation format conversion, and data augmentation to improve training performance. Eight training configurations were evaluated by varying learning rate, batch size, and optimizer. Initial comparisons showed YOLOv12 significantly outperformed SSD, achieving mAP50 of 0.978 and mAP50–95 of 0.831, compared to SSD's 0.816 and 0.639. SGD consistently provided more stable and accurate performance than Adam. The best model used SGD with a learning rate of 0.001 and batch size of 16, achieving precision of 0.952, recall of 0.955, mAP50 of 0.974, and mAP50–95 of 0.834. Field testing confirmed strong detection of pedestrians and traffic signs, although challenges remained with small and overlapping objects. Future work should improve small-object detection, expand dataset diversity, and explore advanced architectures or hybrid optimization strategies. These findings support YOLOv12 as a reliable foundation for safer, more efficient self-driving perception systems tailored to Indonesia's complex road environments in real conditions.

Keywords: Autonomous Vehicles, Computer Vision, Intelligent Transportation Systems, Object Detection, YOLOv12

1. INTRODUCTION

The development of autonomous vehicles in Indonesia has shown a growing trend in recent years, both in academia and public policy. Studies show that autonomous vehicles are seen as having a positive impact in supporting the transformation of the transportation system, particularly through increased mobility efficiency, reduced congestion and emissions, and improved traffic safety [1].

Self-driving cars can be understood by examining two main aspects: their technical design and their use in real life. The functional architecture of autonomous vehicles includes perception, planning, and control, with the perception stage identifying environmental conditions using sensors such as cameras, radar, and LIDAR. The results of this detection form the basis for vehicle decision-making and control [2]. This perception technology also forms the basis for the development of Advanced Driver Assistance Systems (ADAS), which are autonomous vehicle assistance systems. ADAS features, such as lane-keeping assist and pedestrian detection, rely heavily on accurate object detection. The system's ability to stay accurate, even in challenging weather and lighting, is achieved by combining sensor data with deep learning methods [3].

The implementation of autonomous vehicles in countries with right-hand driving systems such as the United States, Germany, and China differs from Indonesia, which uses left-hand driving, thus posing challenges in the placement of road signs and features. As a result, artificial intelligence (AI) models trained using datasets from these countries are often not directly compatible with traffic conditions in Indonesia. This is reinforced by Akhauri's findings, which show that although many autonomous vehicle training datasets are freely available, models trained in one visual domain may not perform optimally in other domains that differ spatially or temporally. This research highlights that variations in driving orientation (right- or left-handed), traffic sign design, and vehicle behavior can impede the model's ability to generalize during cross-domain transfer learning [4].

Furthermore, beyond design and positioning discrepancies, the physical state of traffic signs in Indonesia has been documented to deteriorate, with color fading resulting from environmental exposure and inadequate upkeep [5][6]. Consequently, the fading of these signs diminishes visual contrast and shape definition, which in turn complicates the accurate recognition of signs by computer vision-based object detection models [7]. In addition, most object detection models implemented in Indonesia still rely on foreign models trained using international datasets. These models often show reduced accuracy when used on Indonesian roads, which have unique characteristics like a variety of vehicle types and inconsistent traffic patterns [8]. On Indonesia's roads, there are obstacles that differ from those in other countries, such as the presence of pedicabs, public minibusses, and the dominance of motorcycles [9].

YOLO has proven superior in autonomous driving because it provides an optimal balance between accuracy and object detection speed. Based on the results of the study, YOLOv5 achieved competitive mAP performance with real-time inference speed, making it more suitable for dynamic traffic environments compared to Faster R-CNN, which tends to be slow, and SSD, which has lower accuracy [10]. This research uses YOLOv12, which efficiently integrates attention mechanisms into a real-time object detection architecture. The main innovations of YOLOv12 include the application of Area Attention (A2) to reduce the computational complexity of self-attention, Residual Efficient Layer Aggregation Networks (R-ELAN) to improve gradient flow and training stability, and FlashAttention, which optimizes memory access during the inference process [11] [12].

Previous research shows that hyperparameter tuning on YOLOv4 can improve object detection performance, while also highlighting the need for validation on more diverse datasets and real-world environments [13]. However, many prior studies remain limited by insufficient class diversity, non-representative traffic conditions, and the absence of field testing. To address these gaps, this study develops and optimizes a YOLOv12-based object detection model for autonomous vehicle perception under Indonesian road conditions. The proposed dataset contains 29 object classes, including vehicles, pedestrians, and traffic signs, providing a broader representation of the heterogeneous traffic environment in Indonesia. In addition, the model is evaluated through real-world testing to measure its practical robustness and deployment feasibility. The main contributions of this study are the use of a representative Indonesian traffic dataset, systematic optimization of YOLOv12 hyperparameters, and real-world validation of detection performance.

2. MATERIALS AND METHOD

The methodology was systematically developed, from literature review and data collection to model evaluation.

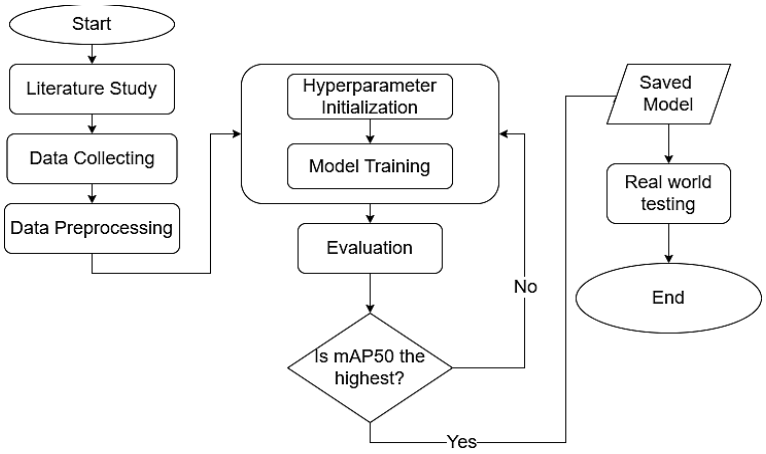


Figure 1. Research Flowchart

The flowchart in Figure 1 illustrates the research workflow that begins with a literature study, data collection, and data preprocessing. It then proceeds to the model development stage, which includes hyperparameter initialization and model training. After that, the model's performance is evaluated. If the mAP50 value has not yet reached its maximum, the process returns to the hyperparameter adjustment and training stage. If the mAP50 value is already the highest, the model is saved (saved model) and the process continues to real-world testing before finally ending (end).

2.1. Autonomous Vehicles

According to Hernandez et al., autonomous vehicle architecture is generally divided into two perspectives: technical and functional. Technical architecture describes the interconnection between

hardware and software components in a vehicle system. The hardware includes sensors (cameras, LiDAR, radar, GNSS, IMU), processing units (such as GPUs and FPGAs), vehicle-to-environment communication (V2X), and mobile platforms and actuators. All of these components are interconnected via high-speed internal network interfaces such as CAN, LIN, USB 3.x, and Gigabit Ethernet. The software architecture of the system relies on frameworks like AUTOSAR, ROS, ROS2, and RTOS. These are the building blocks for creating more complex applications. Think data collection, AI-driven algorithms, infotainment systems, and the real-time control software that's absolutely crucial. The relationship between these elements is illustrated in Figure 2, which shows how hardware and software interact within the technical architecture of an autonomous vehicle [2].

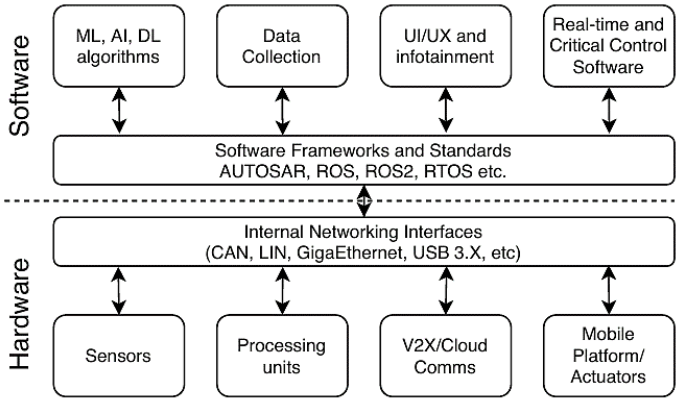


Figure 2. Technical architecture for autonomous driving systems

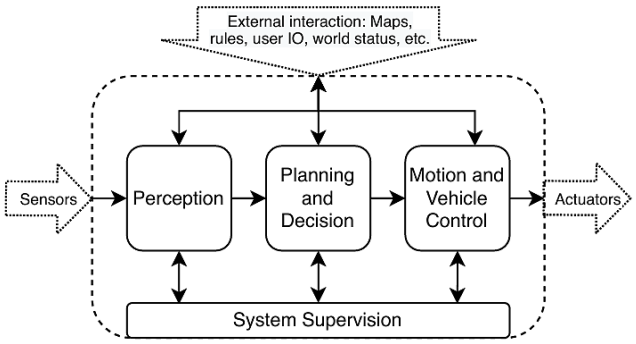


Figure 3. Functional architecture for autonomous driving systems

The functional architecture of autonomous driving systems emphasizes the flow of data processing, which is organized into four main stages: perception, planning and decision-making, vehicle control, and system monitoring.Sensors provide the data input for the perception stage, enabling the system to construct a representation of the environment and the vehicle's condition. Subsequently, this information is transmitted to the planning stage, where the vehicle's actions or trajectory are determined, and then implemented via actuator control. The system monitoring stage ensures that all functions are running safely and according to standards. The perception stage primarily concerns itself with localization and mapping, alongside object detection; both of these processes are significantly reliant on the integration of proprioceptive and exteroceptive sensors, facilitated by computer vision and data fusion methodologies. Figure 3 illustrates the flow of this information, thereby elucidating the function of each component and its interaction with external data sources, such as maps and traffic regulations [3].

2.2. Object Detection

Object detection algorithms based on convolutional neural networks (CNN) generally consist of a backbone for feature extraction, a neck for feature fusion, and a detection head for classification and localization. In autonomous driving, CNN-based detectors are widely used because they can automatically learn visual patterns from road scenes such as vehicles, pedestrians, and traffic signs. Two-stage detectors such as Faster R-CNN first generate candidate regions and then classify them, which increases computational complexity and inference time. Recent studies reported that although two-stage methods maintain strong accuracy, their slower processing speed makes them less practical for real-time driving systems that require rapid environmental perception and decision-making [14].

To satisfy real-time constraints, one-stage detectors such as SSD and YOLO directly predict object classes and bounding boxes in a single forward pass. SSD uses multi-scale feature maps to detect objects of different sizes, while YOLO performs simultaneous localization and classification through a unified detection pipeline. Recent comparative studies found that one-stage detectors offer a more suitable trade-off between latency and detection performance for intelligent transportation and autonomous vehicle applications, where continuous scene changes demand fast response time. Therefore, SSD and YOLO remain widely adopted approaches for traffic scene perception tasks [15].

YOLOv12, an advancement within the YOLO framework, is engineered to utilize attention mechanisms to improve visual context understanding, all while preserving swift inference capabilities. This model's architecture is built upon two fundamental components: the Residual Efficient Layer Aggregation Network (R-ELAN) and FlashAttention. R-ELAN is employed to optimize the feature fusion procedure, integrating residual pathways that promote a more consistent gradient flow throughout the training phase, especially when dealing with expansive models [11]. YOLOv12 architecture can be seen in Figure 4.

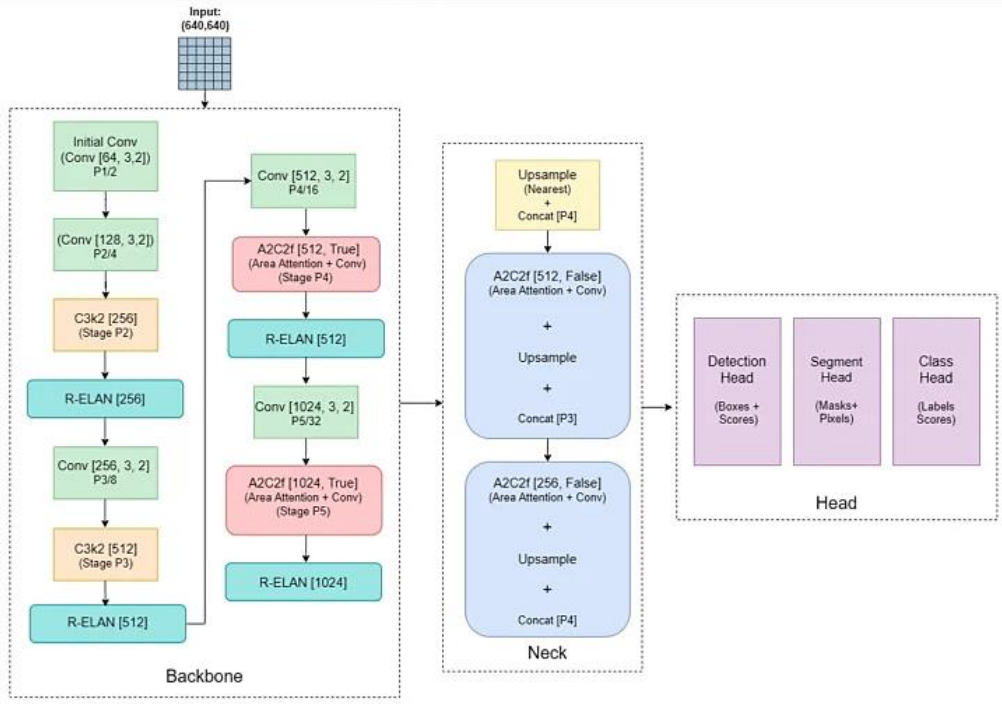


Figure 4. YOLOv12 architecture

At the same time, FlashAttention is used to speed up calculations in the attention mechanism by reducing the need to access GPU memory.With the combination of these two components, YOLOv12 is able to run attention at speeds equivalent to previous CNN-based models, but with higher accuracy [12]. The YOLOv12 research uses MS COCO 2017 as the standard dataset for training and evaluating performance. To evaluate the extent of the improvements, YOLOv12 was compared to several well-known real-time detection models. These included YOLOv6, YOLOv7, YOLOv8, YOLOv9, YOLOv10, and YOLOv11.Comparisons were made consistently across five model scales (N, S, M, L, X) with an input resolution of 640×640 [16].

2.3. Hyperparameter Tuning

Hyperparameter selection has a big effect on how well a model works in machine learning, especially deep learning. Hyperparameters are parameters that are set before training starts and don't change throughout training. Model parameters, on the other hand, are learned automatically [17].

Bischl et al. [18] say that hyperparameter tuning is the process of finding the set of variables that gives the optimum outcomes for model performance in terms of both accuracy and generalization. When using deep learning, it's common to change a few crucial hyperparameters. These are the number of epochs, the learning rate, the batch size, and the choice of optimizer.

2.4. Data Collecting

A dataset of 9,005 labeled pictures and 29 types of items connected to traffic in cities and on highways was created during the data collection phase. Roboflow's public repository provided the dataset. It was put together from several sources to better highlight how diverse Indonesian roadways are. The pictures

that were shot demonstrate how the light, weather, viewing angles, item scale, partial occlusion, traffic density, and faded traffic signs are in Indonesia. All of the notes were in the YOLO bounding-box format, so they could be used right away to train the model.

We divided the dataset into three parts: testing, training, and validation. This was done to assist build and test the model. The training set was used to find the parameters, the three data divisions can be seen in Table 1.

Table 1. Dataset Distribution for Training, Validation, and Testing

Dataset	*Data Training*	*Data Validation*	*Data Testing*	Total
Vehicles	3.179	873	434	4.486
Traffic signs	3.078	292	149	3.519
Pedestrian	800	100	100	1.000

The dataset's object classes are grouped into several categories. The pedestrian category consists of the Pedestrian class. The "prohibition" signs cover a variety of restrictions: No Stopping, No Entry, No Parking, No Right Turn, No Left Turn, Stop, and No U-Turn. Traffic and warning signs, meanwhile, encompass signals like the Green Light, Yellow Light, and Red Light, along with Traffic Light Warnings, Pedestrian Crossing Warnings, General Warnings, Railway Crossing Warnings, and T-Junction Left Warnings. Guidance signs, in contrast, include Keep Left, Pass Either Side, Parking Area, Bus Stop, U-Turn Point, and the Pedestrian Crossing Sign. Meanwhile, the vehicle category consists of Ambulance, Bus, Bicycle, Car, CNG, Motorcycle, and Other Vehicle.

Figure 5. Example of faded traffic sign images

Figure 5 shows a traffic sign that has faded colors and low contrast, which is a common sight on Indonesian roads. This condition makes it difficult for computer vision-based object detection models to accurately recognize the shape and symbols of traffic signs, especially when the model is trained using a standard dataset with better quality traffic signs.

2.5. Data Preprocessing

Figure 6. Example of images before and after resizing

In Figure 6, you can see an example of training data photos that have been preprocessed by being made the same size. All of the photographs in the collection have been shrunk to 640 × 640 pixels, which is the same size for all of them. The images were all made to be the same size so that the model could process them the same way every time it trained. Also, three separate datasets were combined to make one single, unified dataset. This process needed to make sure that the class labels and annotation structures were all the

same. To make sure everything was the same, the method needed to standardize the definitions of the labels in all the datasets. At the same time, all the notes were standardized. This enabled the YOLOv12 model to reliably learn from the data during training. So, bringing the datasets together increased the variety of the training data while keeping the labeling the same, which is very important.

2.6. Modeling

This stage focuses on developing and optimizing the YOLOv12 model for autonomous vehicle navigation. Experiments were used to evaluate various combinations of optimizers, batch sizes, and learning rates. Each configuration was trained and validated to measure its effect on model accuracy.

Table 2. Hyperparameter configuration

Hyperparameter Tuning	Variation
Optimizer	[Adam, SGD]
Batch Size	[8, 16]
Learning rate	[0.01, 0.001]

To achieve optimal training performance, hyperparameter selection was conducted within the model, assessing variations in the optimizer, batch size, and learning rate; these parameters directly influence the model's stability and convergence in object detection applications (see Table 2) [19]. The optimization algorithms employed were Adam and Stochastic Gradient Descent (SGD). Adam is a widely adopted optimizer in neural network training due to its ability to dynamically adjust learning rates for each parameter, which helps the model converge more quickly [21]. In contrast, stochastic gradient descent (SGD) was used as a baseline, given its straightforward and consistent method for updating weights [22]. The batch sizes were 8 and 16, and the learning rates were 0.01 and 0.001, selected to evaluate the effects of gradient stability and convergence speed during training. A learning rate that is too high has the potential to cause divergence, while a lower value provides more stable weight updates [19] Each hyperparameter combination is evaluated using precision, recall, and mean Average Precision (mAP), and the configuration with the highest mAP is selected. Research device specifications are shown in Table 3.

Table 3. Research Device Specifications

Component	Specification
Python version	3.12.12
Processor	Intel(R) Xeon(R) CPU @ 2.00GHz (2 Cores / 4 Threads)
Pytorch version	2.8.0+cu126
Ultralytics Version	8.4.41
GPU Model	Tesla P100-PCIE-16GB

2.7. Evaluation

A confusion matrix is used to evaluate the performance of a classification model by comparing the predicted results with the actual labels. The matrix consists of True Positives (TP), False Positives (FP), and False Negatives (FN). This approach has also been applied to measure model performance and obtain accuracy values from classification results in previous studies [20]. In object detection, true negatives are not used in performance evaluation because the task focuses on identifying and localizing objects, and there are a large number (even an infinite number) of background bounding boxes that are irrelevant to evaluation [21]. Based on these values, the evaluation metrics used in this study include precision and recall, which are formulated as Equations 1 and 2.

$$Precision = \frac{\text{TP}}{\text{TP+FP}} \tag{1}$$

$$Recall = \frac{\text{TP}}{\text{TP+FN}} \tag{2}$$

In addition, the object detection model's performance is evaluated using Mean Average Precision (mAP), which is calculated by averaging the Average Precisions (APs) across all classes from the precision–recall curve. Mathematically, mAP is formulated 3.

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{3}$$

where N is the number of classes, and APi is the average precision for class I [23].

3. RESULT AND DISCUSSION

3.1 Result

The testing focused on the effect of training parameter variations, such as optimizer, batch size, and learning rate, on the model's ability to detect objects. The experimental results are presented in the form of evaluation metrics and visual examples of detection results, which are then analyzed to identify the model's strengths and limitations.

Table 4. Comparison of YOLOv12 and SSD models

No	Eksperimen	Precision	Recall	mAP50	mAP50-95
1	SSD	0.923	0.708	0.816	0.639
2	YOLOv12	0.949	0.955	0.978	0.831

The results in Table 4 comparing SSD and YOLOv12 show that YOLOv12 performs significantly better at object detection. SSD achieved an mAP50 of 0.816 and an mAP50–95 of 0.639, while YOLOv12 reached much higher scores, with an mAP50 of 0.978 and an mAP50–95 of 0.831. This indicates that YOLOv12 is more precise and produces more consistent results across different Intersection over Union (IoU) thresholds. Because of its stronger overall performance, YOLOv12 was chosen for further improvement through hyperparameter tuning to enhance its detection capabilities (see Table 5).

Table 5. Hyperparameter configuration experiment results

No	Optimizer	Batch Size	Learning rate	Precision	Recall	mAP50	mAP50-95
1	Adam	8	0.01	0.862	0.791	0.881	0.698
2	Adam	8	0.001	0.941	0.912	0.961	0.802
3	Adam	16	0.01	0.853	0.801	0.882	0.7
4	Adam	16	0.001	0.965	0.93	0.973	0.824
5	SGD	8	0.01	0.967	0.931	0.972	0.824
6	SGD	8	0.001	0.957	0.948	0.973	0.836
7	SGD	16	0.01	0.965	0.93	0.972	0.827
8	SGD	16	0.001	0.952	0.955	0.974	0.834

The baseline model had a precision of 0.949, a recall of 0.955, a mAP50 of 0.978, and a mAP50–95 of 0.831. The tuned model had a precision of 0.952, a recall of 0.955, a mAP50 of 0.974, and a mAP50–95 of 0.834. These results show that tweaking the hyperparameters made the precision and mAP50–95 better, which means that the model was more consistent at detecting objects across a range of IoU thresholds. The baseline model, on the other hand, had a slightly higher mAP50. Also, both models had the same recall value, which means they were equally good at finding relevant things. In general, the comparison shows that changing the hyperparameters affected performance on numerous evaluation criteria, with each model having strengths in distinct areas.

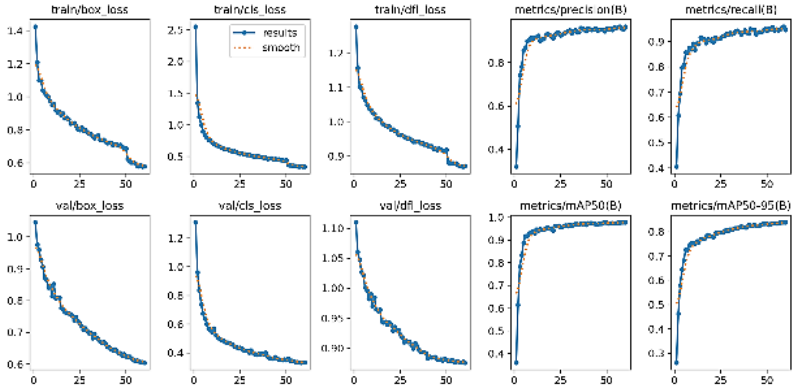
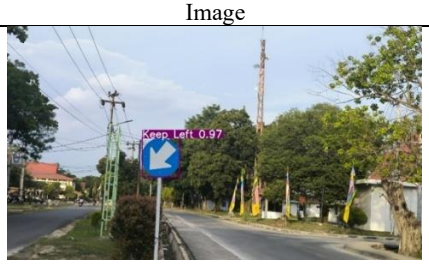
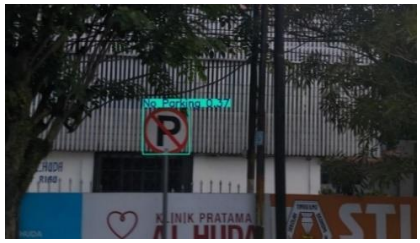
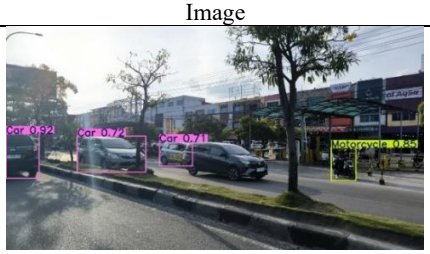


Figure 7. The best model training results

Based on the model training results over 60 epochs in Figure 7, the box loss, classification loss, and objectness loss values on both the training and validation sets gradually decreased as the epochs progressed. In addition, precision and recall increased during the early epochs and then tended to stabilize toward the end of training, indicating that the model had converged. After the training process was completed, testing was conducted using the best-performing model by collecting image data from several areas in Pekanbaru, including Universitas Riau, Soekarno Hatta Street, and HR. Soebrantas Street.

Table 6. Model testing results

No	Image	Detected Label	Description
1		Keep Left	Successfully detected
2		Pedestrian	Successfully detected
3		General warning, pedestrian	Successfully detected
4		Green Light	Successfully detected
5		No parking	Successfully detected
6		U-Turn Point	Successfully detected
7		Car, pedestrian, motorcycle	There are undetected cars and motorcycles due to object overlapping, which makes it difficult for the model to separate objects that are close to each other

No	Image	Detected Label	Description
8		Car, Motorcycle	The model showed good accuracy in identifying objects, but it sometimes failed to detect objects that were hidden by others
9		Pedestrian	The model accurately identified pedestrians. However, it struggled to detect motorcycles at a distance, which was likely due to their smaller size.
10		Pedestrian, car	One car was successfully detected, while the other car was not detected due to overlapping with the car in front of it, whereas the pedestrian object remained detected
11		T-Junction Left Warning	The model detects traffic signs well, but motorcycles in the surrounding area are not detected.

Based on the model testing in Table 6 using test data from several areas in Pekanbaru in Table 4, the results show that the model successfully detected most objects according to the expected labels, with several cases where vehicle objects were not detected due to overlapping, partial occlusion, or small object size.

3.2 Discussion

The tests were about how changing the training parameters, like the optimizer, batch size, and learning rate, affected the model's capacity to find objects. We used assessment metrics and visual detection examples to show the experimental outcomes. Then we looked at them to figure out what each model did well and what it didn't. When comparing SSD and YOLOv12, YOLOv12 performed better at detecting objects. It got a mAP50 of 0.978 and a mAP50–95 of 0.831, while SSD got a mAP50 of 0.816 and a mAP50–95 of 0.6396. These results show that YOLOv12 gave more accurate and consistent detection results over a range of intersection over union (IoU) criteria. So, YOLOv12 was chosen for more optimization by changing its hyperparameters.The hyperparameter tuning tests evaluated several combinations of optimizer, batch size, and learning rate to determine whether they could improve model performance. When comparing the baseline model to the best hyperparameter configuration, the baseline model had a precision of 0.949, a recall of 0.955, a mAP50 of 0.978, and a mAP50–95 of 0.831. The tuned model, on the other hand, had a precision of 0.952, a recall of 0.955, a mAP50 of 0.974, and a mAP50–95 of 0.834. These results show that tuning the hyperparameters improved precision and mAP50–95, indicating that the model was more consistent in detecting objects across different IoU thresholds. The baseline model, on the other hand, had a slightly higher mAP50. Also, both models had the same recall, indicating they were equally good at finding relevant items. In general, the comparison shows that tweaking the hyperparameters affected performance on a number of evaluation criteria, with each model having strengths in distinct areas.

The experimental results indicate that the optimizer substantially influences the performance of the object detection model. In this study, we employed Adam and Stochastic Gradient Descent (SGD) as optimizers, changing the batch size and learning rate each time. The test findings reveal that the two optimizers behave very differently across all the measures used to evaluate them.When the learning rate was dropped to 0.001, the Adam optimizer's performance improved a lot, as seen by higher accuracy, recall,

mAP50, and mAP50–95 values. Adam's performance tended to go down when the learning rate was higher (0.01), which shows that this optimizer is sensitive to the choice of learning rate. On the other hand, the SGD optimizer always did better than Adam in almost all of the tests. The Stochastic Gradient Descent (SGD) optimizer showed higher mAP50 and mAP50–95 scores when the learning rate was set to 0.001, regardless of whether the batch size was 8 or 16.This result shows that SGD is better at generalizing. SGD can do better than Adam because its non-adaptive update rule keeps gradient noise during training, which helps the optimizer avoid abrupt minima and find flatter solutions that work better in general. Adam's adaptive scaling and momentum averaging make heavy-tailed gradient noise less of a problem, which makes it tougher to get out of acute basins.

Then, Adam usually converges faster in the beginning of training, but its ability to generalize is often less than SGD's. This is why hybrid approaches that move from Adam to SGD in later stages are needed. These results show why SGD might do better at detecting things in the end, even though it takes longer to converge [24].The best model was made with the SGD optimizer, a learning rate of 0.001, and a batch size of 16. It had a precision value of 0.952, a recall value of 0.955, a mAP50 value of 0.974, and a mAP50–95 value of 0.834. We chose the model because it had a nice balance between precision and recall. It also had the best detection success rate of 0.50, which was better than the other setups. Table 3 illustrates how well the model did on the test data, which reveals how effectively it can see traffic signs. The signs tell you to do things like "Keep Left," "General Warning," "Green Light," "No Parking," "U-Turn Point," and "T-Junction Left Warning." The pedestrian identification system worked well in a number of situations, which shows that the model is very effective at recognizing how people see things. On the other hand, the model can only recognize cars and motorcycles under specified situations. When items are close together or partially obscured, they can overlap, which can lead to mistakes in detection. Because of this, the model has trouble correctly figuring out where each object's edges are. The model also has trouble identifying distant vehicles, which look smaller, because of the resolution of the data and the amount of visual information it has. After resizing an image, small objects are often missed because they take up very few pixels, and when they are downsampled again, they lose important spatial features. Overlapping objects are also harder to separate because of partial occlusion and Non-Maximum Suppression (NMS), which suppresses nearby valid detections [25]. To get over these problems, you can increase the resolution of the input images, balance and enlarge the training dataset, use data augmentation techniques, and switch to better detection architectures with stronger multi-scale feature fusion [26].

3. CONCLUSION

In conclusion, this study's findings confirm that both the choice of model and the tuning of training parameters have a clear impact on object detection performance. From the experiments conducted, YOLOv12 generally performs better than SSD, particularly in terms of accuracy and consistency. It reached an mAP50 of 0.978 and an mAP50–95 of 0.831, suggesting that it is more reliable across different detection scenarios. The hyperparameter experiments also show that performance is quite sensitive to the choice of optimizer, learning rate, and batch size. Among the configurations tested, SGD with a learning rate of 0.001 and a batch size of 16 gave the most balanced results, achieving a precision of 0.952 and a recall of 0.955, along with strong mAP scores. That said, the model is not without limitations, as it still struggles to detect small, distant, overlapping, or partially occluded objects, especially vehicles.

Looking ahead, several improvements could be considered. One straightforward step would be to use higher-resolution images, which may help the model better capture smaller or more distant objects. Improving the dataset is also important; adding more varied and well-balanced examples of vehicles, pedestrians, and traffic signs could make the model more robust. In addition, trying out different data augmentation techniques may help the model adapt better to changing environmental conditions. Future work could also explore newer detection approaches or more effective ways of combining features across scales. Finally, it may be worth investigating hybrid optimization strategies, for example, by combining the faster convergence of Adam with the more stable generalization typically associated with SGD.

REFERENCES

[1] M. A. R. Pohan, “Studi Literatur Sistematis Potensi Kendaraan Otonom dalam Transformasi Transportasi oleh Pemerintah Daerah di Indonesia,” *Jurnal Teknologi dan Komunikasi Pemerintahan*, Oct. 2025, doi: 10.33701/jtkp.v7i1.4958.

[2] G. Velasco-Hernandez, D. J. Yeong, J. Barry, and J. Walsh, “Autonomous Driving Architectures, Perception and Data Fusion: A Review,” in *Proceedings - 2020 IEEE 16th International Conference on Intelligent Computer Communication and Processing, ICCP 2020*, Institute of Electrical and Electronics Engineers Inc., Sep. 2020, pp. 315–321. doi: 10.1109/ICCP51029.2020.9266268.

[3] F. M. Barbosa and F. S. Osório, “Camera-Radar Perception for Autonomous Vehicles and ADAS: Concepts, Datasets and Metrics,” Mar. 2023, doi: 10.48550/arXiv.2303.04302.

[4] S. Akhauri, L. Zheng, T. Goldstein, and M. Lin, "Improving Generalization of Transfer Learning Across Domains Using Spatio-Temporal Features in Autonomous Driving," Sep. 2021, doi: 10.48550/arXiv.2103.08116.
[5] B. Prastiyo, "Inspeksi Keselamatan Jalan di Ruas Jalan Nasional Kota Jambi," *Himpunan Pengembangan Jalan Indonesia*, vol. 10, no. 1, pp. 45–52, 2024, doi: 10.26593/jhpji.v10i1.7647.45-52.
[6] N. Y. A. Ula, B. R. Martha, and S. Hadi, "Inspeksi Keselamatan Jalan pada Jalan Ahmad Yani di Kota Magelang," *Jurnal Teknik Gradien*, vol. 17, no. 01, pp. 13–24, 2025, doi: 10.47329/teknik_gradien.v17i01.1392.
[7] Y. Zhu and W. Q. Yan, "Traffic sign recognition based on deep learning," *Multimed. Tools Appl.*, vol. 81, no. 13, pp. 17779–17791, May 2022, doi: 10.1007/s11042-022-12163-0.
[8] O. V. Putra and I. N. Gustri, "Sistem Deteksi Marka Jalan Berbasis Convolutional Neural Network," *Journal of Computer Engineering, Network, and Intelligent Multimedia*, vol. 1, no. 1, pp. 1–13, Feb. 2023, doi: 10.59378/jcenim.v1i1.2.
[9] A. Mulyanto, W. Jatmiko, P. Mursanto, P. Prasetyawan, and R. I. Borman, "A new indonesian traffic obstacle dataset and performance evaluation of yolov4 for adas," *Journal of ICT Research and Applications*, vol. 14, no. 3, pp. 285–298, 2021, doi: 10.5614/ITBJ.ICT.RES.APPL.2021.14.3.6.
[10] N. Yinkfu, S. Nwovu, J. Kayizzi, and A. Uwamahoro, "Comparative Analysis of YOLOv5, Faster R-CNN, SSD, and RetinaNet for Motorbike Detection in Kigali Autonomous Driving Context," Oct. 2025, doi: 10.48550/arXiv.2510.04912.
[11] R. Khanam and M. Hussain, "A Review of YOLOv12: Attention-Based Enhancements vs. Previous Versions," *arXiv preprint arXiv*, Apr. 2025, doi: 10.48550/arXiv.2504.11995.
[12] M. A. R. Alif and M. Hussain, "YOLOv12: A Breakdown of the Key Architectural Features," *arxiv preprint*, Feb. 2025, doi: 10.48550/arXiv.2502.14740.
[13] G. D. Deepak and S. K. Bhat, "Optimizing YOLOv4 Hyperparameters for Enhanced Vehicle Detection in Intelligent Transportation Systems," *International Journal of Intelligent Transportation Systems Research*, Dec. 2025, doi: 10.1007/s13177-025-00519-3.
[14] S. Y. Mohammed, "Architecture review: Two-stage and one-stage object detection," Sep. 01, 2025, *Elsevier B.V.* doi: 10.1016/j.fraope.2025.100322.
[15] M. Carranza-García, J. Torres-Mateo, P. Lara-Benítez, and J. García-Gutiérrez, "On the performance of one-stage and two-stage object detectors in autonomous vehicles using camera data," *Remote Sens. (Basel).*, vol. 13, no. 1, pp. 1–23, Jan. 2021, doi: 10.3390/rs13010089.
[16] Y. Tian, Q. Ye, and D. Doermann, "YOLOv12: Attention-Centric Real-Time Object Detectors," *arXiv preprint arXiv*, 2025, doi: 10.48550/arXiv.2502.12524.
[17] T. Yu and H. Zhu, "Hyper-Parameter Optimization: A Review of Algorithms and Applications," *arXiv preprint arXiv*, Mar. 2020, doi: 10.48550/arXiv.2003.05689.
[18] B. Bischl *et al.*, "Hyperparameter Optimization: Foundations, Algorithms, Best Practices and Open Challenges," *WIREs Data Mining and Knowledge Discovery*, vol. 13, Nov. 2021, doi: 10.48550/arXiv.2107.05847.
[19] O. G. Ajayi, P. O. Ibrahim, and O. S. Adegboyega, "Effect of Hyperparameter Tuning on the Performance of YOLOv8 for Multi Crop Classification on UAV Images," *Applied Sciences (Switzerland)*, vol. 14, no. 13, Jul. 2024, doi: 10.3390/app14135708.
[20] R. Farid Abdillah and D. Ramadhani, "Nominal Detection of Rupiah Banknotes with Audio Output Using MobileNetV2 Transfer Learning Method," Jurnal Ilmu Komputer dan Informasi, vol. 19, no. 1, pp. 99–106, Mar. 2026.
[21] G. Y. Kim and M.-H. Oh, "Adam Optimization with Adaptive Batch Selection," *arXvic*, Dec. 2025, doi: 10.48550/arXiv.2512.06795.
[22] H. Guo, J. Jin, and B. Liu, "Stochastic Weight Averaging Revisited," *Applied Sciences (Switzerland)*, vol. 13, no. 5, Mar. 2023, doi: 10.3390/app13052935.
[23] R. Padilla, W. L. Passos, T. L. B. Dias, S. L. Netto, and E. A. B. Da Silva, "A comparative analysis of object detection metrics with a companion open-source toolkit," *Electronics (Switzerland)*, vol. 10, no. 3, pp. 1–28, Feb. 2021, doi: 10.3390/electronics10030279.
[24] P. Zhou, J. Feng, C. Ma, C. Xiong, S. Hoi, and W. E, "Towards Theoretically Understanding Why SGD Generalizes Better Than ADAM in Deep Learning," *arXiv preprint arXiv*, Nov. 2021, doi: 10.48550/arXiv.2010.05627.
[25] Y. Shi, Y. Jia, and X. Zhang, "FocusDet: an efficient object detector for small object," *Sci. Rep.*, vol. 14, no. 1, Dec. 2024, doi: 10.1038/s41598-024-61136-w.
[26] Y. Li, K. Wu, W. Kang, Y. Zhou, and F. Di, "Multi-object detection for crowded road scene based on ML-AFP of YOLOv5," *Sci. Rep.*, vol. 13, no. 1, Dec. 2023, doi: 10.1038/s41598-023-43458-3.