Proceedings of CONF-SEML 2026 Symposium: Computational Analysis and Modeling in Complex Intelligent Systems
DOI: 10.54254/2755-2721/2026.GU33496

Optimizing Neural Networks for Real-Time Vehicle Tracking on Resource-Constrained Edge Devices

Yuhao Zhou

Faculty of Science, Toronto Metropolitan University, Toronton, Canada
y12zhou@torontomu.ca

Abstract. Real-time vehicle detection is an essential ability for intelligent transportation systems. However, to achievive high reliable and low-latency tracking locally on edge devices is extremely challenging due to severe hardware constraints. Highly compressed models (such as FOMO 0.1) suffer from severe under-detection, missing almost 40% of the vehicles. On the other hand, complex detectors like YOLO-Pro, is able to deliver strong spatial precision for medium targets (mAP@50 of 0.79), but still would struggle with scale variations, which hinders them from robust real-world application. In this work,this paperinvestigate the optimal trade-off between model capacity and accuracy by evaluating three architectures (MobileNetV1, MobileNetV2-based FOMO, and YOLO-Pro) across two vehicle datasets under different circumstancces. The evaluation shows that moderate capacity scaling yields the best performance. MobileNetV1 at a 0.25 width multiplier reaches 98.1% accuracy, and FOMO 0.35 achieves an 87.6% F1-score. Performance is assessed using confusion matrices, F1-scores, and mAP for both spatial localization and class separation. Keeping width multipliers between 0.25 and 0.35 successfully bridges the gap between extreme compression and feature retention, making the approach a possible solution for resource-constrained edge deployment.

Keywords: Vehicle Detection, Lightweight Neural Networks, MobileNet, Width Multiplier

1. Introduction

Vehicle detection is an important task in modern transportation systems. Traffic vehicle detection continues to be a crucial computer vision problem as intelligent transportation systems develop, supporting applications like autonomous driving, traffic monitoring, and vehicle safety [1]. Vehicle detection has undeniably become an important part of modern intelligent transportation systems. For city planners and traffic management centers, stationary cameras powered by computer vision are no longer just passive recording devices; they actively track lane density, automatically flag red-light violations, and instantly spot dangerous anomalies like stalled cars. Vehicle detection is a prevailing computer technology, which uses deep learning techniques to analyze the visual surroundings [2]. The autonomous driving and traffic monitoring are significant branches of vehicle detection. It has become more and more mature in practical applications. Visual object detection is divided into two things which are classification and localization. In the case of the classification process, the model

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2026.GU33496

matches the image with categories stored in the dataset. In the case of the detection process, the vehicle is located from the background by using bounding box algorithms.

In the early field of object detection, the previous technology was using manual feature engineering, referring to some basic knowledge of edges and simple shapes. However, the traditional feature extraction cannot meet the shortage in real-world roadside environments. For example, the cluttered images cannot be identified with high accuracy, an important reason is the traditional technology cannot deal with the fluctuations in natural lighting and unpredictable camera angles by itself. Over the past decade, various computer vision tasks have been addressed effectively by using Convolutional Neural Network (CNN). In [3], Redmon proposed the YOLO model. In [4], Ren proposed the Faster R-CNN model. While, in [5], Howard proposed the MobileNet model, which implements depthwise convolutions and reduces the requirement of massive cloud servers. And in [6], these CNN frameworks have pushed visual detection into intelligent monitoring and military systems. But this paper hopes to explore the CNN in the condition of unconstrained roadside environments, compared with the traditional cloud computing advantage.

In this regard, this paper designs a CNN pipeline to recognize vehicles. The model makes a judgment according to the structural patterns (such as wheels and chassis lines) to decide whether the input images contain vehicles. To compare the detection ability of different CNN in this domain, this paper used MobileNet architectures and YOLO architecture. This paper designs network using the Edge Impulse platform to identify the input roadside imagery, and the network will use transfer learning to extract pre-trained visual features. According to the features, the system will distinguish vehicles from environmental noise like fences and trees. In the end, the proposed models aim to prove that compressed deep learning methods are highly viable for continuous deployment in resource-constrained intelligent transportation systems.

2. Method

2.1. Dataset preparation

In this study, this paper used a primary vehicle dataset to evaluate the classification approach [7]. The dataset provides images of vehicles and non-vehicles shown in Figure 1 and Figure 2. The dataset consists of 396 vehicle images and 395 non-vehicle images for the training process. And this paper kept 200 images for the testing process to prevent bias. The vehicle samples cover different angles like front, side, and rear. The non-vehicle images feature roads, vegetation, and traffic signs. To pre-process the data, this paper first resized the images to 96x96 pixels. Another step was to convert them to grayscale. This paper did this because this paper needs to minimize the computational overhead and keep the classification model lightweight.

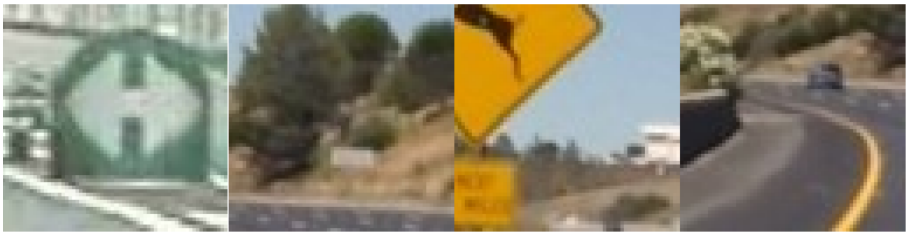


Figure 1. Non-vehicle examples [7]

DOI: 10.54254/2755-2721/2026.GU33496



Figure 2. Vehicle examples [7]

This paper also used a secondary dataset to evaluate the object detection tasks. This dataset contains complex traffic scenes with multiple vehicles shown in Figure 3. The secondary dataset consists of 627 images. On average, there are about 15 vehicles in each image. For these images, this paper kept the resolution at 640x640 pixels. This paper did not convert them to 96x96 pixels because this paper needs to preserve the spatial details to detect multiple small objects. For data preprocessing, this paper normalized the pixel values for all images. Normalization can help to stabilize the training process. Another important step for data preprocessing was data augmentation. In the dataset, this paper applied some standard methods like random flips, rotations, and brightness shifts. In this way, this paper reduced overfitting and made the model more robust. This paper also kept the test set strictly separated during the training to make sure the evaluation is unbiased.

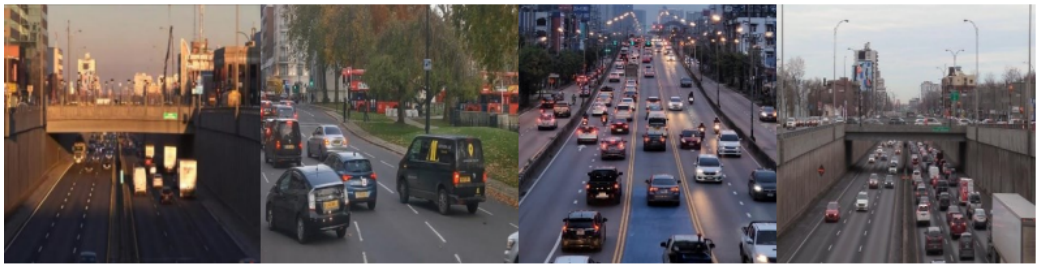


Figure 3. Complex traffic example [7]

2.2. Edge impulse-based CNN model

To build the models, this paper used a cloud-based platform called Edge Impulse [8,9]. This platform is built for developing TinyML systems on limited hardware. This paper managed all the steps here, which includes data formatting, neural network training, and final edge deployment. For the baseline classification model, this paper used a method called transfer learning. By using transfer learning, the models do not need to learn basic edges and shapes from the beginning. Instead, this paper just fine-tuned the pre-trained filters to recognize the specific vehicle targets.

For the classification model structure, this paper selected MobileNetV1 and this paper tested fthe width multipliers which are 0.1, 0.2, 0.25, and 0.35. MobileNet uses depthwise separable convolutions [5]. This function helps to slash computational costs and keep the representational power at the same time. Combined with the grayscale images, this paper trained this classification model for 40 epochs and the learning rate was 0.001. During the training process, this paper also monitored the validation accuracy and the loss metrics. In this way, this paper can make sure the model is learning the features correctly and check if there is any overfitting problem.

After finishing the classification model, this paper started training the detection models to tackle the complex multi-object traffic scenes. This paper trained two FOMO (Faster Objects, More Objects) models. These two models are based on MobileNetV2 with 0.1 and 0.35 multipliers. At the

DOI: 10.54254/2755-2721/2026.GU33496

same time, this paper also trained a standard YOLO-Pro model to compare the baseline. The FOMO framework is designed for very constrained devices. It delivers real-time localization and consumes up to 30 times less compute and memory compared to the conventional YOLOv5 setups [10]. Now, this paper is able to deploy these robust classification and detection systems directly into the real-world traffic environments.

3. Results and discussion

The performance of MobileNetV1 models have an input resolution of 96×96 and varying width multipliers (0.1, 0.2, 0.25, and 0.35) shown in Table 1.

Table 1. MobileNetV1 training result

Model	Accuracy	Loss
MobileNetV1 0.1	95.0%	0.16
MobileNetV1 0.2	96.9%	0.06
MobileNetV1 0.25	98.1%	0.06
MobileNetV1 0.35	100.0%	0.00

From the training results, this paper can see the MobileNetV1 0.1 model has the lowest accuracy. As it's shown in the confusion matrix shown in Table 2, the model predicts 7.9% of the vehicle samples as non-vehicles. At the same time, it mistakes 2.4% of the non-vehicles for vehicles. This may be because the network does not have enough depth to extract the features of vehicles for real application.

Table 2. MobileNetV1 confusion matrix

MobileNetV1 0.1	Non-Vehicle	Vehicle	MobileNetV1 0.2	Non-Vehicle	Vehicle
Non-Vehicle	97.6%	2.4%	Non-Vehicle	98.8%	1.2%
Vehicle	7.9%	92.1%	Vehicle	5.3%	94.7%
F1 Score	0.95	0.95	F1 Score	0.97	0.97
MobileNetV1 0.25	Non-Vehicle	Vehicle	MobileNetV1 0.35	Non-Vehicle	Vehicle
Non-Vehicle	100%	0%	Non-Vehicle	100%	0%
Vehicle	3.9%	96.1%	Vehicle	0%	100%
F1 Score	0.98	0.98	F1 Score	1.00	1.00

By adjusting the width multiplier to 0.2, the result becomes much better. The overall accuracy increases to 96.9%, and the validation loss decreases to 0.06. From the confusion matrix, this paper can see the improvement clearly. The vehicle misclassifications drop to 5.3%, and the non-vehicle errors drop to 1.2%. This shows that adding moderate capacity can help extract features much better.

Furthermore, the multiplier of 0.25 has an even better and more stable result. The model gets an accuracy of 98.1% and the loss is 0.06. In addition, it gets 100% accuracy on the non-vehicle class, so there are no false positives. The vehicle misclassification is only 3.9%. This means it has a strong generalization ability on unseen data. This paper thinks this model has the best balance between accuracy and hardware cost.

In addition to the classification models, this paper also evaluated the FOMO MobileNetV2 models with 0.1 and 0.35 multipliers and a YOLO-Pro model. As is shown in Table 3 and Table 4,

DOI: 10.54254/2755-2721/2026.GU33496

the FOMO 0.1 model has an overall F1 score of 74.1%. It predicts the background samples 100% correctly, but it classifies many vehicles wrongly. Actually, 39.5% of the vehicle samples are predicted to be background, and its vehicle F1 score is 0.74. This may be because the 0.1 model is too small to find the discriminative features, which leads to high false-negative errors.

However, the FOMO 0.35 model has a much better performance. The overall F1 score becomes 87.6%, and the vehicle detection accuracy is 82.9%. From the confusion matrix, this paper can see that the vehicle-to-background error is reduced to 17.1%, and the vehicle F1 score is 0.88. This shows that the bigger model width can help the model separate the different classes. It is a much better candidate for real-time tracking.

Table 3. FOMO MobileNetV2 result

Model	F1 Score
FOMO MobileNetV2 0.1	74.1%
FOMO MobileNetV2 0.35	87.6%

Table 4. FOMO MobileNetV2 confusion matix

FOMO MobileNetV2 0.35	Background	Vehicle
Background	100.0%	0.0%
Vehicle	17.1%	82.9%
F1 Score	1.00	0.88
FOMO MobileNetV2 0.1	Background	Vehicle
Background	100.0%	0.0%
Vehicle	39.5%	60.5%
F1 Score	1.00	0.74

Finally, this paper also tested the YOLO-Pro model to see its localization ability. From Table 5,this paper can see that it achieves a mAP@50 of 0.79 and a precision score of 89.4%. But the YOLO-Pro model is very sensitive to the object scale. It has a good mAP of 0.43 on medium objects, but it only has 0.25 on small objects and 0.27 on large objects. Besides, it has a peak recall of 0.49 at 100 detections. This indicates that the YOLO-Pro model is precise, but it still needs more optimization to improve the localization for all the spatial dimensions.

Table 5. YOLO-Pro

Metrics	VALUE
mAP	0.37
mAP@[IoU=50]	0.79
mAP@[IoU=75]	0.28
mAP@[area=small]	0.25
mAP@[area=medium]	0.43
mAP@[area=large]	0.27
Recall@[max_detections=1]	0.05
Recall@[max_detections=10]	0.30

DOI: 10.54254/2755-2721/2026.GU33496

Table 5. (continued)

Recall@[max_detections=100]	0.49
Recall@[area=small]	0.37
Recall@[area=medium]	0.54
Recall@[area=large]	0.39
Precision score (legacy)	89.4%

The results indicate that while the FOMO 0.1 is ideal for extreme resource constraints, but it shows limited detection performance. The FOMO 0.35 provides a superior performance to complexity ratio. For applications requiring precise spatial localization, YOLO-Pro demonstrates high precision, particularly for medium-scale targets, making it the most suitable candidate for complex visual environments.

4. Conclusion

This research addressed the critical need to balance detection accuracy with computational efficiency in edge-based vehicle tracking systems. This paper evaluated the FOMO, YOLO-Pro, and MobileNetV1 models to improve the performance of resource-constrained vehicle tracking. Extensive experiments were conducted to evaluate the proposed method. Experimental results showed that the models configured with moderate width multipliers (such as 0.25 and 0.35) achieve a robust balance, performing comparable to or even better than high-capacity baselines in terms of efficiency and accuracy under various scale settings. In the future work, this paper plans to deploy these trained models into physical edge hardware, such as a Raspberry Pi 4 or an ESP32 microcontroller. Even if the FOMO model consumes very little memory, the real outdoor environment is still a big challenge. The edge camera may face extreme weather like heavy rain or snow, which will cover the lens and reduce the image quality. Therefore, this paper might need to collect more data sets in bad weather conditions to train the model again.

References

[1] Minh, T. H., et al. (2025). Robust traffic vehicle detection under real-world conditions with misclassified vehicles minimization and weighted box fusion. In 2025 International Conference on Multimedia Analysis and Pattern Recognition (MAPR) (pp. 1–6).

[2] Mehune, P., & Dorle, S. (2024). Robust and adaptive neural networks for self-driving cars in challenging road conditions. In 2024 2nd International Conference on Computer, Communication and Control (IC4) (pp. 1–5).

[3] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. arXiv.

[4] Ren, S., He, K., Girshick, R. B. & Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama & R. Garnett (eds.), NIPS (p./pp. 91-99) .

[5] Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. ArXiv, abs/1704.04861.

[6] Gao, C., Zhai, Y., & Guo, X. (2021). Visual Object Detection and Tracking System Dfsign based on MobileNet-SSD. 2021 7th International Conference on Computer and Communications (ICCC), 589-593.

[7] Dincer, B. (2021). Vehicle detection image set [Data set]. Kaggle. https: //www.kaggle.com/datasets/brsdincer/vehicle-detection-image-set

[8] Janapa Reddi, V., Elium, A., Hymel, S., Tischler, D., Situnayake, D., Ward, C., ... & Quaye, J. (2023). Edge impulse: An mlops platform for tiny machine learning. Proceedings of Machine Learning and Systems, 5, 254-268.

[9] Mihigo, I. N., Zennaro, M., Uwitonze, A., Rwigema, J., & Rovai, M. (2022). On-device iot-based predictive maintenance analytics model: Comparing tinylstm and tinymodel from edge impulse. Sensors, 22(14), 5174.

DOI: 10.54254/2755-2721/2026.GU33496

[10] Moreau, L. (2024). Announcing FOMO (faster objects, more objects). The Edge AI Blog - Edge Impulse. https://www.edgeimpulse.com/blog/announcing-fomo-faster-objects-more-objects/