Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 9, Number 2, April 2025
DOI : https://doi.org/10.33395/sinkron.v9i2.14637
e-ISSN : 2541-2019
p-ISSN : 2541-044X

Vehicle Type Classification and Detection System using YOLOv7-tiny Model on Single-Board Computer

Faridatun Nadziroh[1)], Nihayatus Sa'adah[2)], Rahardita Widyatra Sudibyo[3)], Haniah Mahmudah[4)], Moch. Imam Rifai[5)]
[1,2,3,4,5)]Electrical Engineering, Telecomunications Engineering Program, Politeknik Elektronika Negeri Surabaya, Indonesia
[1)]faridatun@pens.ac.id, [2)]niha@pens.ac.id, [3)]widi@pens.ac.id, [4)]haniah@pens.ac.id, [5)]mochimamrifai@gmail.com

Submitted :March 12, 2025 | **Accepted** : March 26, 2025 | **Published** : April 14, 2025

Abstract: Transportation is playing an important role for human civilization, for example transportation is being used as distributing goods and products. Therefore, the total number of vehicles as a part of transportation will continue to increase every year. But in Indonesia, the majority of its people is still using their personal transport rather than public transportation. This is supported by the data of total number of vehicles in Indonesia from 2018 – 2020, which is shows that personal transport is still dominant than public transportation. The causes of traffic jams is a result of various factors, such as the roads are not designed to accommodate the increasing number of vehicles, insufficient traffic signs, and poor traffic management. The road traffic data is one of the aspects that could reduce traffic jams. The process of collecting road traffic data which is still done manually has several shortcomings, such as it takes a long time and there may be errors due to human error. This research has a goal to create a vehicle type detection and classification system that have a good detection accuracy and detection speed that can be run on single-board computer devices. YOLOv7-tiny model that performs detection and classification using input from video on the NVIDIA Jetson Nano device gets a True Positive (TP) score of 96.58%, a False Positive (FP) score of 0.98%, and a False Negative (FN) score of 2.44%. YOLOv7-tiny on the NVIDIA Jetson Nano device can run with an average Frame per Second (FPS) of 6 FPS.

Keywords: Object Detection; Vehicles Classification; Single-Board Computer; YOLOv7-tiny

INTRODUCTION

Transportation is without a doubt is playing an important role for human civilization, for example transportations is being used as distributing goods and products. But in Indonesia, many of its people is still using their personal transport rather than public transportation. We could see that from the data of total number of vehicles in Indonesia from 2018 – 2020, which is shows that personal transport like motorcycles and cars is still dominant than buses, and the total number of vehicles will be increased every year from previous year (Janik; Pusat Statistik, 2022).

The causes of traffic jams is a result of various factors. One of those factors is that the roads are not designed to accommodate the increasing number of vehicles. The other reason is insufficient traffic signs. Poor traffic management is also a major cause of traffic jams, which could lead to bottlenecks). Reducing traffic jams requires road traffic data, which can be used to evaluate existing traffic management systems and implement better traffic flow management strategies. Unfortunately, the process of collecting road traffic data is usually still done manually. By assigning several people to count the number of motorized vehicles that are passing by the road, then dividing them over a certain period. This process requires precision, and it still has several shortcomings, such as it takes a long time and there may be errors due to human error (Vazirani et al., 2016).

We often hear the term deep learning for several applications of machine learning, such as object detection and image classification. Deep learning is an algorithm designed to replicate the structure and function of the human brain, which is called artificial neural networks. A neural network consists of several layers that are connected to

*name of corresponding author

e-ISSN : 2541-2019
p-ISSN : 2541-044X

the next layer. This network is similar to the human brain, which is composed of neurons that send signals from one neuron to another. More layers will make the network deeper, where in the last layer there are results from a collection of inputs called the weights (Taye, 2023).

An example of deep learning implementation is You Only Look Once (YOLO) model. YOLO provides a different approach compared to other object detection models that reuse classifiers to perform detection. YOLO works by reframing object detection as regression to create bounding boxes along with class prediction probabilities. In YOLO, a neural network will predict the bounding box and its class prediction probability from an image in one evaluation (Redmon et al., 2016).

This research has a goal to create a vehicle type detection and classification system that is expected to have a good performance in terms of accuracy, precision, and detection speed on a single-board computer device so that it can be used as a reference for similar prototypes. In this research, the types of vehicles that will be detected and classified are the types of vehicles commonly found in Indonesia (Ariza & Baez, 2022).

The YOLO model was chosen in this research because it has good detection accuracy and precision, but the model is still light enough to be able to perform real-time detection and classification. In (Tan et al., 2021), the YOLO v3 model was compared with the RetinaNet and Single Shot Detector (SSD) models to perform real-time medicine type classification. The result is that YOLO v3 gets a Mean Average Precision (mAP) score lower than the other two models, which is 80.69% while RetinaNet and SSD get a score of 82.89% and 82.71%. However, the speed of the YOLO v3 model is superior, getting 69 Frames per Second (FPS) while RetinaNet and SSD get 22 and 49 FPS, respectively.

In this research, the YOLOv7-tiny model was chosen, because the YOLOv7-tiny model is the model that has the lightest computation compared to other YOLOv7 model variants. In addition, the YOLOv7 model is a model of the YOLO variant that has better performance than the previous YOLO variants prior to YOLOv7 (Wang et al., 2023). The model will be embedded in the NVIDIA Jetson Nano single-board computer.

LITERATURE REVIEW

Some researchers have done research related to this topic of study. From the following research (Manajang et al., 2020), a pre-trained YOLOv3 object detection model was assigned to perform motorized vehicle detection and classification. Tests were conducted on a computer with an AMD A9-9425 processor with 4 GB of RAM. The performance of the model was tested using several videos of 10 seconds to 1 minute duration. The average accuracy of detection and classification is 90.8%.

Research (Naftali et al., 2022) compares five one-stage object detection algorithms, including SSD MobileNetv2 FPN-lite and various YOLO versions, for street-level object detection using a modified Udacity Self Driving Car dataset. The results show that YOLOv5l achieves the highest accuracy (mAP@0.5 = 0.593), MobileNetv2 FPN-lite has the fastest inference time (3.20ms), and YOLOv5s offers the best balance between speed and accuracy. These findings indicate that the evaluated algorithms are suitable for real-time object detection in autonomous driving systems..

Research (Liu et al., 2019) compared the results of the optimization method applied to YOLOv3. The researchers used an optimization framework called TensorRT. Without optimization, the inference time of the model is 323 ms with mAP 52.48%. Alternatively, it can get a faster inference time of 154 ms with a mAP of 35.36% by using optimization techniques including loop function, kernel tuning, and Floating Point (FP) FP16 quantization.

Research (Balamuralidhar et al., 2021) implemented a model called MultEYE (Multi-task Entwined YOLO and ENet), which is an implementation of YOLO and ENet into an Unmanned Aerial Vehicle (UAV). The model was tested on a system with GPU NVIDIA Titan Xp and got the mAP 0.834 and 43.5 FPS. The model was also tested on an NVIDIA Jetson Xavier NX device, and the result is the model could achieve 29.41 FPS.

Research (Shafiee et al., 2017) developed a modified model of YOLOv2 network architecture that can be run on embedded computing devices. This model is called Fast YOLO. The model was tested on NVIDIA Jetson TX1 using input from a video. Fast YOLO is faster than YOLOv2, where Fast YOLO got 17.85 FPS while YOLOv2 got 5.4 FPS.

The difference between this research and research that has been done by several previous researchers is that the edge computing device or single-board computer that has been used in this research is the NVIDIA Jetson Nano. Classified objects are vehicles that correspond to document.

METHOD

This section will explain the methodology of this research and explain the design and implementation of the system.

A. YOLOv7 Model

This section will explain about the main focus and the network architecture of YOLOv7. The

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 9, Number 2, April 2025
DOI : https://doi.org/10.33395/sinkron.v9i2.14637
e-ISSN : 2541-2019
p-ISSN : 2541-044X

development of YOLOv7 has a main focus on optimizing the training process by optimizing modules and methods that can be used to increase accuracy but not the inference cost. The method is called trainable bag-of-freebies.

YOLOv7 changes its architecture in model scaling to obtain a model scaling method that does not require overcomplicated rules and less computational cost, by designing a new compound scaling method called Extended Efficient Layer Aggregation Networks (E-ELAN). E-ELAN has been designed by analyzing several factors such as memory access cost, the influence of input/output channel ratio, activation when performing model scaling, and gradient path. The objective of model scaling is to adjust some attributes of the model that can be used for specific needs.

YOLOv7 has several model variants developed for specific usages, such as a model suitable for use in edge computing device GPUs called YOLOv7-tiny, a model for consumer GPU purposes called YOLOv7, and a model for cloud computing GPU called YOLOv7-W6. From YOLOv7 research (Wang et al., 2023) YOLOv7 shows improvement in terms of detection accuracy and inference speed. We can see the result of comparison of YOLOv7 with other YOLO variants from the graphic in Fig. 1.

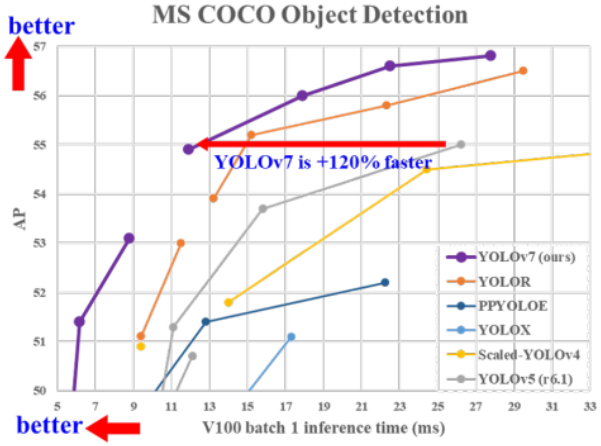

Fig 1. Comparison of YOLOv7 performance with other YOLO (Wang et al., 2023)

B. *System Design*

The system design for the detection and classification of vehicle types using the YOLOv7-tiny model consists of a USB camera that is used to record road traffic conditions as video and an NVIDIA Jetson Nano that is used to detect and classify the vehicle types. The output of detection and classification process is a video. The illustration of this system design is shown in Fig. 2.

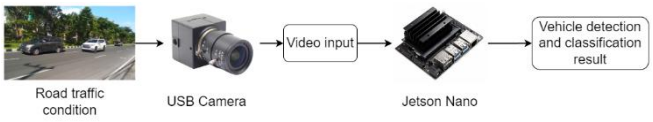

Fig 2. An illustration of the system design.

C. *Custom Dataset Description*

A dataset is a big set of data that is used as input by the object detection model to carry out the training process. In this research, the type of vehicle that will be classified by the YOLOv7-tiny model is in accordance with document, which is a type of vehicle that is commonly found in Indonesia. Because this type of vehicle is slightly different from the vehicle type dataset that is already publicly available, a custom dataset is needed that is more suitable for vehicle types in Indonesia. This custom dataset is self-collected by searching for images of vehicle types in Indonesia via the internet and additional images using personally obtained recordings of road traffic conditions.

The total number of images for the custom dataset is 23,243 images. After collecting the images for the custom dataset, the next step is to resize the entire image to 640x640 pixels. All of these images need to be labeled, which is done by labeling objects according to the correct type of vehicle. From 23,243 images there are total 40,968 labels of vehicles. Statistics of the distribution of each label or class in the dataset are shown in Table I.

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 9, Number 2, April 2025
DOI : https://doi.org/10.33395/sinkron.v9i2.14637
e-ISSN : 2541-2019
p-ISSN : 2541-044X

TABLE I. Dataset Label Distribution

Label	Type of Vehicle	Quantity of Lables	Percentage
1	Motorcycle, bajaj	10612	25.90%
2	Sedan, jeep, family car	13479	32.90%
3	Mini-bus, commuter car	1686	4.11%
4	Pick-up	3350	8.18%
5a	Small bus	1053	2.57%
5b	Big bus	2198	5.37%
6a	4 wheeled truck	2143	5.22%
6b	6 wheeled truck	2354	5.75%
7a	3 axle truck	1143	2.79%
7b	Double load truck	100	0.24%
7c	Truck trailer	746	1.82%
8	Non-motorized vehicle	2104	5.14%
Total		**40968**	**100%**

The custom dataset will be divided into 70% for the training process and 30% for the YOLOv7-tiny model evaluation process.

D. System Implementation

The system implementation of the YOLOv7-tiny model that will be embedded in the NVIDIA Jetson Nano device will be shown in the Fig. 3.

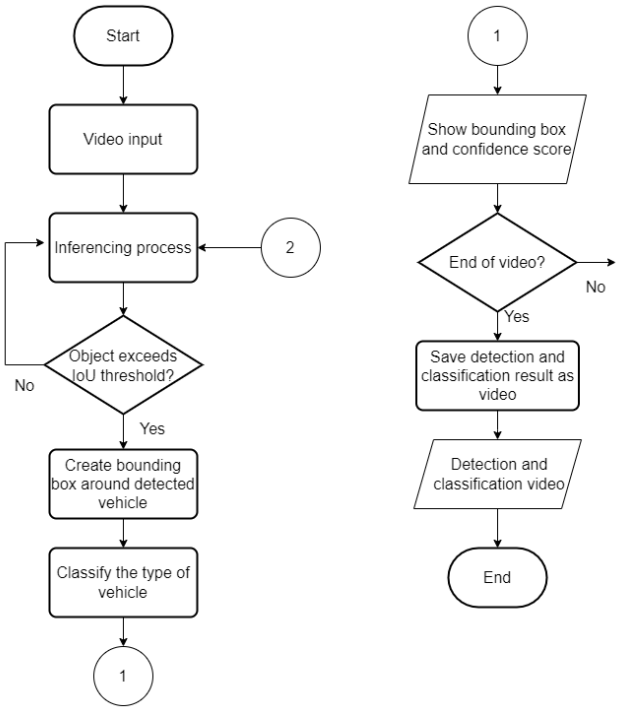

Fig 3. Flowchart of detection and classification process.

Fig. 3 shows a flowchart of detection and classification process. The input for the inference process is a video obtained from recording road traffic conditions. NVIDIA Jetson Nano will perform the inference process to detect and classify vehicles. When an object in the video frame exceeds the Intersection over Union (IoU) threshold, the object is considered a vehicle. Furthermore, the YOLOv7 model will create a bounding box around the object area. So that the results can be seen in the form of a bounding box and the value of the vehicle type prediction score. The result is a video that shows the bounding box on the vehicle that was successfully detected and classified.

E. Environment for Training and Training Scenarios

This training process is the process of retraining the YOLOv7-tiny model to be able to detect and classify vehicle types following the custom dataset. This training process is called transfer learning, where the pre-trained model is retrained using the custom dataset.

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 9, Number 2, April 2025
DOI : https://doi.org/10.33395/sinkron.v9i2.14637
e-ISSN : 2541-2019
p-ISSN : 2541-044X

The training process of the YOLOv7-tiny model is carried out on the Google Colab server because it provides a GPU that can be used for training object detection models for free. Below is a table of Google Colab server specifications shown in Table II.

TABLE II. Google Colab Specification

CPU	Intel Xeon CPU 2.20 GHz with 2 vCPU
GPU	NVIDIA Tesla T4 15 GB
Memory	12 GB
Disk space	107 GB (available capacity 77 GB)
Python	3.10.12
PyTorch	2.0.1+cu118
PyTorch CUDA	11.8

The training process is divided into several scenarios, that's because the training process is conducted by comparing different optimizers and also different batch sizes. The optimizers used during the training process are Stochastic Gradient Descent (SGD), Adam, and Root Mean Squared Propagation (RMSprop). From the three optimizers, the training process is carried out with different batch sizes, namely 2, 4, 8, and 16. Furthermore, the training scenario is named SGD 2, SGD 4, SGD 8, and SGD 16, where the number after the optimizer name indicates the number of batch sizes used during training to make it easier to mention. This Table III shows the training scenarios with its hyperparameters.

TABLE III. Training Scenarios with Optimizers and Batch Size

Parameter	**SGD**				**Adam**				**RMSprop**			
	2	**4**	**8**	**16**	**2**	**4**	**8**	**16**	**2**	**4**	**8**	**16**
Num Classes	12				12				12			
Epoch	100				100				100			
Initial Learning Rate	0,001				0,001				0,001			
One Cycle Learning Rate	0,1				0,1				0,1			

RESULT

In this section, we will test and analyze the system that has been made. Testing is carried out to know whether the system that has been designed can run properly with the expected performance. The test results will provide an overview so that it can be evaluated.

A. Evaluation of Training Scenarios

Every time one epoch is completed from the training process, there will be a total loss value. This value comes from the accumulated values of bounding box loss, objectness loss, and classification loss. The results of the total loss comparison during training for all training scenarios are shown in Fig 4.

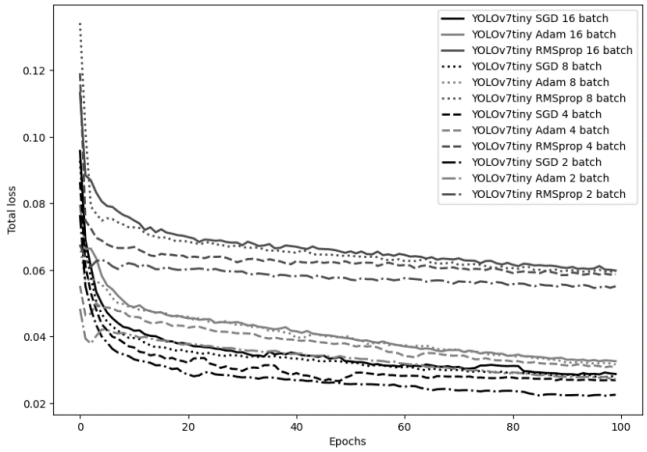

Fig 4. Total loss comparison for all training scenarios.

It can be seen from the graph above, that SGD 2 or the training process using SGD optimizer and batch size 2 has the lowest total loss. That means the model with this training scenario is the best in total loss during training. But that does not mean that training scenario is the best.

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 9, Number 2, April 2025
DOI : https://doi.org/10.33395/sinkron.v9i2.14637
e-ISSN : 2541-2019
p-ISSN : 2541-044X

The training process is considered complete if it has completed training up to 100 epochs. The evaluation process after finishing training is calculate the value of precision and recall. Precision and recall are obtained from the following equation (1) and (2) as shown below:

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \quad (1)$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \quad (2)$$

Precision is used as parameter to determine the level of accuracy between ground truth (real data) and the prediction results of the model. To determine how well a positive value is found, recall is employed. A metric called mAP is used to assess a model's performance by taking into account its recall, precision, IoU, and confusion matrix. Table IV below shows the results of the precision and recall metrics for all training scenarios.

TABLE IV. Comparison of Precision and Recall

Scenario	Precision	Recall
SGD 16	0.791	0.814
Adam 16	0.753	0.787
RMSprop 16	0.382	0.464
SGD 8	0.858	0.794
Adam 8	0.689	0.784
RMSprop 8	0.227	0.352
SGD 4	0.82	0.818
Adam 4	0.685	0.774
RMSprop 4	0.851	0.0619
SGD 2	0.85	0.774
Adam 2	0.602	0.777
RMSprop 2	0.927	0.0647

Another evaluation metric are mAP@.5 and mAP@.5:.95, which is shown in Table V.

TABLE V. Comparison of mAP

Scenario	mAP@.5	mAP@.5:.95
SGD 16	0.859	0.724
Adam 16	0.799	0.646
RMSprop 16	0.281	0.137
SGD 8	0.856	0.721
Adam 8	0.783	0.633
RMSprop 8	0.187	0.0889
SGD 4	0.853	0.711
Adam 4	0.752	0.604
RMSprop 4	0.0781	0.0334
SGD 2	0.835	0.694
Adam 2	0.726	0.582
RMSprop 2	0.0884	0.0325

Based on the results of Table IV and Table V, it can be seen that the highest precision and recall values are obtained by models with training scenarios using the SGD optimizer with a batch size of 8. The model gets a precision of 0.858 and a recall of 0.794. Then the model with the highest mAP@.5 and mAP@.5:.95 values, with mAP@.5 worth 0.859 and mAP@.5:.95 worth 0.724 is found in the model with a training scenario using the SGD optimizer with a batch size of 16.
Based on these results, the model with the training scenario using the SGD optimizer with a batch size of 16 will be selected as the model to be used for the process of detection and classification of vehicle types on the Jetson Nano.

*name of corresponding author

Sinkron : Jurnal dan Penelitian Teknik Informatika
Volume 9, Number 2, April 2025
DOI : https://doi.org/10.33395/sinkron.v9i2.14637
e-ISSN : 2541-2019
p-ISSN : 2541-044X

B. Detection and Classification on Jetson Nano

This detection and classification test uses a video sample with a duration of 2 minutes 16 seconds. The video was obtained by recording road traffic in the Jl. Manyar Kertoarjo. To record the video, the camera position is placed 425 cm above the ground with a camera facing angle of 45 degrees towards the road. Traffic conditions at the time of data collection were smooth and busy.
In this test, the model that will be embedded in Jetson Nano is a YOLOv7-tiny model that has been trained with SGD optimizer and batch size of 16. The result of detection and classification is shown in Table VI below.

TABLE VI. Detection and Classification Result

Type of Vehicle	TP	FP	FN	Actual Vehicle Count	Average FPS
1	132	0	5	137	6
2	59	1	0	60	
3	0	0	0	0	
4	5	0	0	5	
5a	0	0	0	0	
5b	1	0	0	1	
6a	0	0	0	0	
6b	1	1	0	2	
7a	0	0	0	0	
7b	0	0	0	0	
7c	0	0	0	0	
8	0	0	0	0	
Total	198	2	5	205	
Percentage	96.58%	0.98%	2.44%	100%	

Based on the table above, it can be seen that the YOLOv7-tiny model has been able to provide good detection accuracy performance by getting a score for True Positive of 96.58%. Although the results of the detection accuracy performance are good, the YOLOv7-tiny model is still not computationally lightweight enough because it still gets an average FPS of 6 FPS. So, there will be a long delay when used for real-time detection and classification processes.
The result of the detection and classification of vehicle types on Jetson Nano can be seen in Figure 5.

Fig 5. The result of vehicle type detection and classification.

CONCLUSION

The conclusions obtained from the testing and analysis process, several summaries can be drawn for the conclusion of this research : The results of testing the model with 3 different optimizers, namely SGD, Adam, and RMSprop, show that the SGD optimizer is the best compared to other optimizers. SGD optimizer is better than Adam optimizer and RMSprop because this optimizer is more suitable for the dataset used and the network architecture of YOLOv7-tiny. It should be noted that each optimizer will have different results depending on the network architecture, the dataset used, and for what purpose the model is used. The test results on Jetson Nano show that the YOLOv7-tiny model that has been trained with the SGD optimizer and batch size 16 shows good classification performance results, which are 96.58% true positive, 0.98% false positive, and 2.44% false negative.

*name of corresponding author

e-ISSN : 2541-2019
p-ISSN : 2541-044X

The average FPS obtained on Jetson Nano is 6 FPS. The FPS is fast enough if run on the Jetson Nano device, but not so fast to be used in real-time conditions.

REFERENCES

Ariza, J., & Baez, H. (2022). Understanding the role of single-board computers in engineering and computer science education: A systematic literature review. In *Computer Applications in Engineering Education* (Vol. 30, Issue 1, pp. 304–329). John Wiley and Sons Inc. https://doi.org/10.1002/cae.22439

Balamuralidhar, N., Tilon, S., & Nex, F. (2021). MultEYE: Monitoring system for real-time vehicle detection, tracking and speed estimation from UAV imagery on edge-computing platforms. *Remote Sensing*, *13*(4), 1–24. https://doi.org/10.3390/rs13040573

Janik, R. (2022). *Transport in the Beginnings of Civilizations in the Context of Symbolism and Education*. https://doi.org/10.16926/p.2024.33.02

Liu, L., Li, H., & Gruteser, M. (2019, August 7). Edge assisted real-time object detection for mobile augmented reality. *Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM*. https://doi.org/10.1145/3300061.3300116

Manajang, D. J. P., Sompie, S. R. U. A., & Jacobus, A. (2020). *Implementasi Framework Tensorflow Object Detection Dalam Mengklasifikasi Jenis Kendaraan Bermotor*.

Naftali, M. G., Sulistyawan, J. S., & Julian, K. (2022). *Comparison of Object Detection Algorithms for Street-level Objects*. http://arxiv.org/abs/2208.11315

Pusat Statistik, B. (2022). *Statistik daerah Kota Surabaya*. Https://Surabaya.Bps.Go.Id.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). *You Only Look Once: Unified, Real-Time Object Detection*. http://pjreddie.com/yolo/

Shafiee, M. J., Chywl, B., Li, F., & Wong, A. (2017). *Fast YOLO: A Fast You Only Look Once System for Real-time Embedded Object Detection in Video*. http://arxiv.org/abs/1709.05943

Tan, L., Huangfu, T., Wu, L., & Chen, W. (2021). Comparison of RetinaNet, SSD, and YOLO v3 for real-time pill identification. *BMC Medical Informatics and Decision Making*, *21*(1). https://doi.org/10.1186/s12911-021-01691-8

Taye, M. M. (2023). Understanding of Machine Learning with Deep Learning: Architectures, Workflow, Applications and Future Directions. In *Computers* (Vol. 12, Issue 5). MDPI. https://doi.org/10.3390/computers12050091

Vazirani, H., Kautsar, A., Adi, K., & Fisika, J. (2016). Implementasi Object Tracking Untuk Mendeteksi Dan Menghitung Jumlah Kendaraan Secara Otomatis Menggunakan Metode Kalman Filter Dan Gaussian Mixture Model. In *Youngster Physics Journal* (Vol. 5, Issue 1).

Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2023). *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*. https://github.com/

*name of corresponding author