ASIAN JOURNAL OF ELECTRICAL AND ELECTRONIC ENGINEERING
Vol. 2 No. 2 2022
E-ISSN: 2785-8189

Video-Based Vehicle Counting and Analysis using YOLOv5 and DeepSORT with Deployment on Jetson Nano

Abuelgasim Saadeldin*, Muhammad Mahbubur Rashid

Department of Mechatronics, Faculty of Engineering,
International Islamic University Malaysia,
Kuala Lumpur, Malaysia

*Corresponding author: ashaikh.saad.as@gmail.com

(Received: 12Th August 2022; Accepted: 28th August 2022)

Abstract— In recent years, the advancements in deep learning and high-performance edge-computing systems have increased tremendously and have become the center of attention when it comes to the analysis of video-based systems on edge by making use of computer vision techniques. Intelligent Transportation Systems (ITS) is one area where deep learning can be used for several tasks including highway-based vehicle counting systems whereby making use of computer vision techniques, an edge computing device and cameras installed in specific locations on the road, we are able to obtain very accurate vehicle counting results and replace the use of traditional and laborious hardware devices with modern low-cost solutions. This paper proposes and implements a modern, compact, and reliable vehicle counting system based on the most recent and popular object detection algorithm as of writing this paper, known as the YOLOv5, combined with a state-of-the-art object tracking algorithm known as DeepSORT. The YOLOv5 will be used in the following system to detect and classify four different classes of vehicles, whereas DeepSORT will be used to track those vehicles across different frames in the video sequence. Finally, a unique and efficient vehicle counting method will be implemented and used to count tracked vehicles across the highway scenes. A new highway vehicle dataset consisting of four vehicle classes, namely: car, motorcycle, bus, and truck, was collected, cleaned, and annotated with a total of 11,982 images published in the following study and used for the training of our robust vehicle detection model. From the results obtained over real-world highway surveillance videos, the following system was able to obtain an average vehicle detection mAP score of 96.1% and a vehicle counting accuracy of 95.39%, all while being able to be deployed on a compact Nvidia Jetson Nano edge-computing device with an average speed of 15 FPS which outperforms other previously proposed tools in terms of both accuracy and speed.

Keywords: *Vehicle Detection, Vehicle Tracking, Vehicle Counting, YOLOv5, Edge Deployment*

1. INTRODUCTION

In the past decade, the number of vehicles on the road has started to increase, leading to huge amounts of traffic congestion and various other safety concerns. The study of traffic flow analysis began to grow rapidly as researchers tried and identify ways of reducing congestion on the road by gathering essential information about moving vehicles required for developing better traffic management systems. In the past, physical hardware devices were placed underneath roadways and used to collect various data. Recently, with the advancements in technology and computer vision techniques, researchers began to actively look into vision-based solutions to help gather essential information required for maintaining constant traffic flows.

Vision-based solutions can provide us with more detailed information and are significantly easier to install and maintain compared to traditional physical hardware sensors. In this research, we will be making use of the current state-of-the-art object detection and tracking algorithms to develop a real-time highway-based vehicle counting system that will be low-cost, efficient and at the same time also compact enough to be deployed on low computationally expensive edge-computing devices. The YOLOv5, which is the most prominent object detection

DOI: https://doi.org/10.69955/ajoeee.2022.v2i2.34

algorithm, will be used as the base model for the development of our custom vehicle detection algorithm; meanwhile, Simple Online and Realtime Tracking with a Deep Association Metric (DeepSORT) will be used for the tracking of vehicles across different frames in the video sequence. Finally, a unique and efficient counting method will be implemented to count tracked vehicles across the highway scenes.

2. METHODOLOGY

This section presents the methodology of our proposed vehicle counting system. It explains how it has been developed from data collection up to deployment and describes the environmental setup used for testing the following system. This section also elaborates on how the custom vehicle detection architecture was developed and how it has been integrated with a robust DeepSORT object tracking algorithm and a lightweight OSNet ReID model for tracking detected vehicles across different frames in the video sequence. Finally, we introduce a new and efficient vehicle counting method which will be used for counting different vehicle classes as they cross through a virtual polygon area on the highway in real-time.

2.1 System Architecture

Although a vast majority of vehicle counting systems have been implemented in the past, new advancements in the field of AI, object detection and tracking continue to emerge, introducing new features and significant improvements from one another. Fig. 1 below shows a summary of our proposed vehicle counting system, which was developed by utilizing a custom vehicle detection algorithm based on the YOLOv5, combined with a DeepSORT object tracking algorithm and a uniquely developed vehicle counting method. The system is comprised of three main components, which are vehicle detection, tracking, and counting. The vehicle detection will be used for the detection and classification of four different classes of vehicles, the vehicle tracking will be used for the vehicle ID assignment and tracking and finally the vehicle counter will be used for the counting of tracked vehicles as they cross through a virtual polygon area in the highway in real-time.

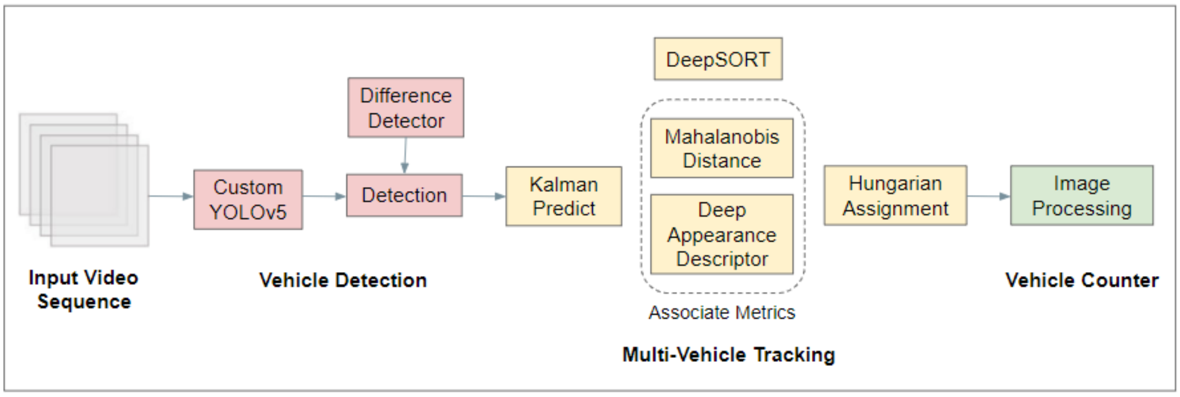


Fig. 1. Proposed System Architecture

2.2 Data Collection and Description

Highway traffic images from various sources, including open-source datasets, footages from CCTV cameras, and manually captured images, were the kinds of data used in this study. The main open-source datasets included the highway vehicle dataset [1] and MIO-TCD dataset [2]. The highway vehicle dataset is a large dataset containing vehicles captured from 23 surveillance cameras located in Hangzhou, China. Meanwhile, the MIO-TCD is a popular traffic dataset acquired from cameras deployed all over Canada and the United States. Open-source CCTV camera footage obtained from various locations at Shah Alam, Malaysia, was also dissected and used in the following dataset. Finally, manually captured images recorded by using a 12-megapixel smartphone camera situated on top of a pedestrian bridge and pointing directly towards a busy highway located in Kuala Lumpur, Malaysia was also utilized for generation of the vehicle dataset.

The generated vehicle dataset consists of four vehicle classes, namely: Car, Motorcycle, Bus, and Truck, which were captured from various locations, different timings throughout the day, varying weather conditions and have dramatic changes in scale, allowing for a robust model that is able to generalize well in different environments and weather conditions. The following dataset was annotated, exported in a YOLO text file format, and split into the train, valid and test sets with the following ratios 0.7, 0.2 and 0.1, respectively. Altogether 11,982 manually annotated images were generated and have been published at the following link: https://tinyurl.com/yc2eycje. In this dataset, cars accounted for 58.23%, motorcycles accounted for 7.35; buses accounted for 11.27% and trucks accounted for 23.15%. Table 1 shows detailed information about the generated dataset along with the number of instances for each of the four vehicle classes in each train, valid and test sets.

Table 1: Vehicle Dataset Information

Subset	Number of Images	Number of Cars	Number of Motorcycles	Number of Buses	Number of Trucks
All	11,982	23,360	1,950	2,975	9,286
Train	8,237	16,252	890	1,382	6,500
Validation	2,350	5,487	60	395	2,653
Test	1,184	2,336	45	198	929

2.2.1 Data Augmentation

Data augmentation was applied to the training images to increase the dataset's quality and allow for a highly generalizable model that can adapt to different environments and weather conditions. The data augmentation that was applied to the training images included the addition of random image rotations, adding noise, rain, and varying brightness, which resulted in an increase in the total number of training samples to 11,597 images, while the validation and test sets remained unchanged. Fig. 2 below shows a sample input image that has gone through the different augmentation techniques and the results are also displayed.

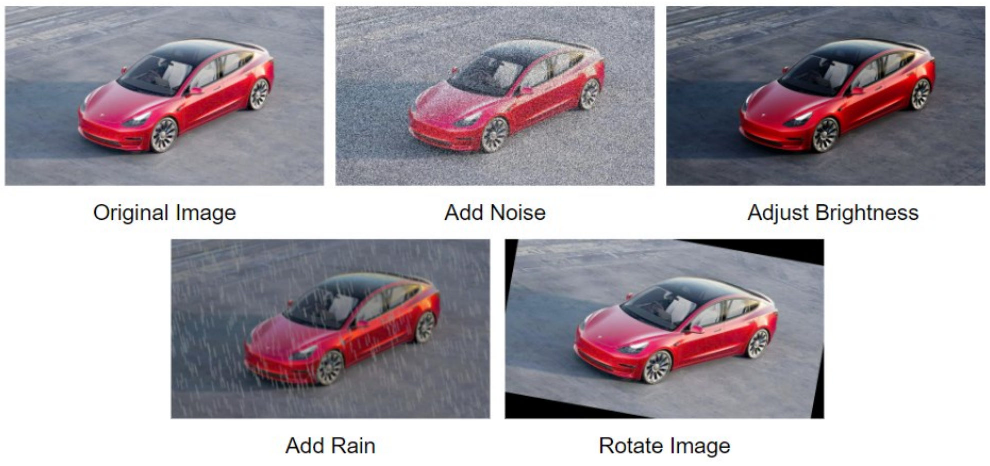


Fig. 2. Different Augmentations applied to Sample Image

2.3 Vehicle Detection

In this study, a CNN-based vehicle detection algorithm was developed and utilized. The algorithm is based on the YOLOv5 which is a popular object detection algorithm written in Python and which makes use of the PyTorch deep learning framework [3]. The YOLOv5 provides five different scales of their model, which are the Nano, Small, Medium, Large and Extra-large as shown in Fig. 3. The difference between each one of these scales is merely in the depth and width of the model which has been expanded, meaning that the overall structure of the model remains the same, however, the size and complexity of the model increases. The sample structures can also be modified by adding more neurons, increasing the number of hidden layers, and performing batch normalization or weight initialization. In our experiment, we made use of the smallest, fastest base model, the YOLOv5n and modified the structure by adding extra layers in order to improve the detection accuracy of smaller vehicle objects observed on highways while still maintaining a lightweight model able to be deployed on embedded devices.

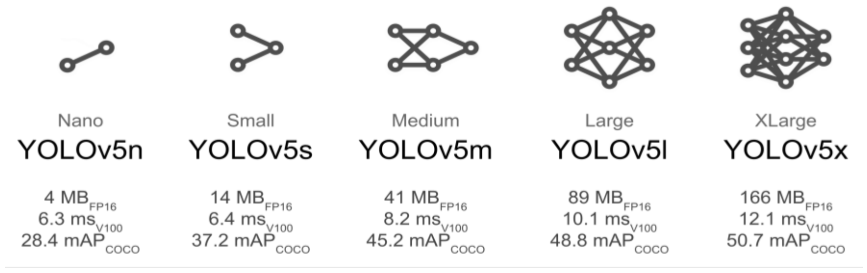
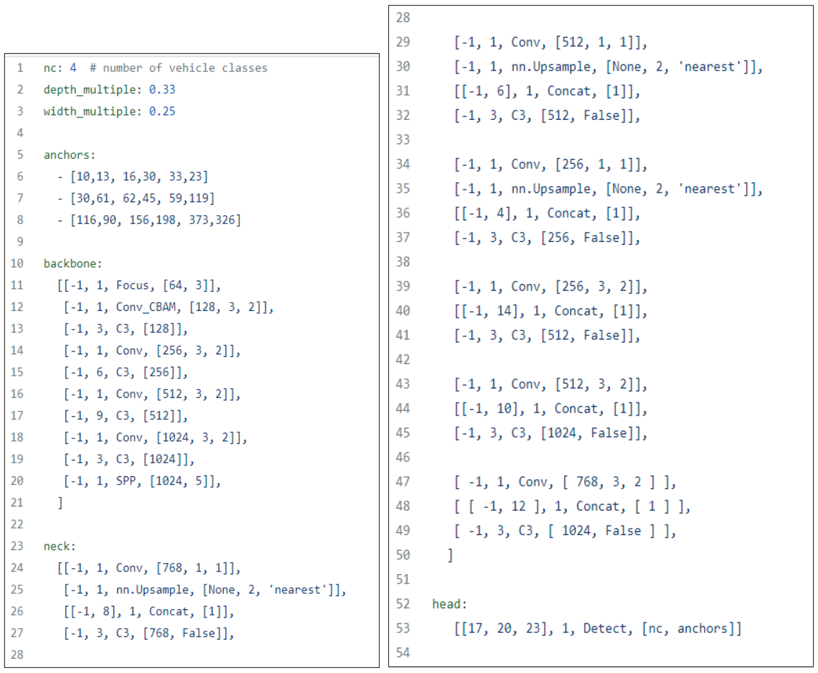


Fig. 3. The Different YOLOv5 Model Scales

The size of vehicles captured from a distance of approximately 5 meters above the ground can be relatively small. Therefore, accurately detecting the vehicles by using the default smallest YOLOv5n model architecture can be difficult. Hence, the addition of modules in order to increase the detection accuracy of smaller vehicle objects while still maintaining a fast and lightweight model is necessary. Thus, we have added an attention mechanism layer to the YOLOv5n model architecture along with additional up-sampling and down-sampling layers to the feature map in order to further improve the model's accuracy on smaller vehicle detections. The attention mechanism layer is a Convolutional Block Attention Module (CBAM) which replaces the original CONV module, allowing for more detailed information about passing vehicles by reducing the attention on roads and other complex backgrounds. Meanwhile, the up-sampling and down-sampling layers enhance the model's ability to detect vehicles at different scales and recognize different sizes and standards of the same vehicle. Fig. 4 below presents the improved model architecture where rows 12, 24-27 and 47-49 represent the replaced attention mechanism and additional small target detection layers, respectively.



```
1   nc: 4  # number of vehicle classes
2   depth_multiple: 0.33
3   width_multiple: 0.25
4
5   anchors:
6     - [10,13, 16,30, 33,23]
7     - [30,61, 62,45, 59,119]
8     - [116,90, 156,198, 373,326]
9
10  backbone:
11    [[-1, 1, Focus, [64, 3]],
12     [-1, 1, Conv_CBAM, [128, 3, 2]],
13     [-1, 3, C3, [128]],
14     [-1, 1, Conv, [256, 3, 2]],
15     [-1, 6, C3, [256]],
16     [-1, 1, Conv, [512, 3, 2]],
17     [-1, 9, C3, [512]],
18     [-1, 1, Conv, [1024, 3, 2]],
19     [-1, 3, C3, [1024]],
20     [-1, 1, SPP, [1024, 5]],
21    ]
22
23  neck:
24    [[-1, 1, Conv, [768, 1, 1]],
25     [-1, 1, nn.Upsample, [None, 2, 'nearest']],
26     [[-1, 8], 1, Concat, [1]],
27     [-1, 3, C3, [768, False]],
28
```

```
28
29     [-1, 1, Conv, [512, 1, 1]],
30     [-1, 1, nn.Upsample, [None, 2, 'nearest']],
31     [[-1, 6], 1, Concat, [1]],
32     [-1, 3, C3, [512, False]],
33
34     [-1, 1, Conv, [256, 1, 1]],
35     [-1, 1, nn.Upsample, [None, 2, 'nearest']],
36     [[-1, 4], 1, Concat, [1]],
37     [-1, 3, C3, [256, False]],
38
39     [-1, 1, Conv, [256, 3, 2]],
40     [[-1, 14], 1, Concat, [1]],
41     [-1, 3, C3, [512, False]],
42
43     [-1, 1, Conv, [512, 3, 2]],
44     [[-1, 10], 1, Concat, [1]],
45     [-1, 3, C3, [1024, False]],
46
47     [ -1, 1, Conv, [ 768, 3, 2 ] ],
48     [ [ -1, 12 ], 1, Concat, [ 1 ] ],
49     [ -1, 3, C3, [ 1024, False ] ],
50    ]
51
52  head:
53     [[17, 20, 23], 1, Detect, [nc, anchors]]
54
```

Fig. 4. Proposed Vehicle Detection Model Architecture

In order to accurately detect small-scale vehicles captured on the highway scenes, after the focus module in the backbone network, an attention mechanism Conv_CBAM module is added which replaces the original CONV module. By doing so, the algorithm is able to obtain more detailed information about passing vehicles by reducing inference on roads and other complex backgrounds. The CBAM module pays attention to the channel information which solves the loss problem caused by the different weights in the feature graph. It works by taking the output of the channel attention module. It then goes through two pooling operations and a convolution operation with a convolution kernel of size 7x7. Finally, a feature graph with the size of HxWx2 is obtained. The main innovation in this network is in the addition of the CBAM and up-sampling and down-sampling layers which allowed for the model to learn the spatial attention features of vehicles through the relationship between channel and space.

2.4 Vehicle Tracking

Once the vehicles have been detected by using the custom YOLOv5 algorithm, vehicle features are then extracted and a DeepSORT multi-object tracking algorithm is used for the matching of features with the other video frames in order to achieve a correlation between the same vehicle and other similar vehicles. The DeepSORT algorithm works by using a combination of the Kalman Filter and Hungarian algorithm for tracking. The Kalman Filter is used for making a prediction of the current state of a vehicle based on some previous value, along with providing the uncertainties of that prediction [4]; a Mahalanobis distance is then used to encompass the uncertainties made by the Kalman Filter. Finally, once the vehicle location has been derived, the Hungarian algorithm is then used for the vehicle association and ID attribution, assigning a unique identification to the vehicle and identifying if the vehicle present in the current frame is the same as that observed in the previous frame [5]. The block diagram of the discussed flow of data in the DeepSORT algorithm can be seen in the Fig. 5.

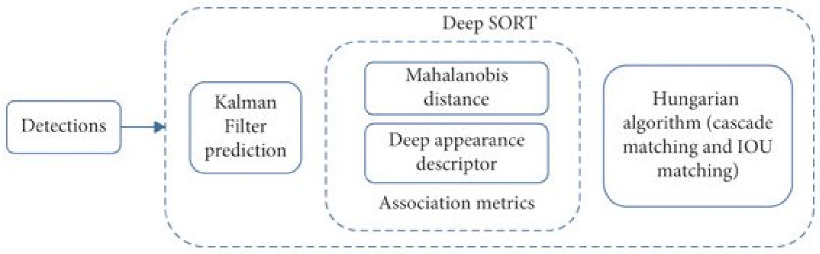


Fig. 5. Flow of Data in the DeepSORT algorithm [6]

The aim of the vehicle tracker is to be able to keep track of the detected vehicles and maintain their ID assignment as they move around in different frames in the video sequence. In order to accomplish this, we employed an Omni-Scale Network (OSNet) ReID model for the extraction of better representational feature vectors required for associating the vehicles in the highway scenes. OSNet specializes in multi-scale predictions and it designs its network with depth-wise separable convolutions, achieving an average mAP score of 84.9% on the Market 1501 dataset [7]. Furthermore, the combination of DeepSORT and OSNet ReID models is also lightweight, allowing for fast inference times and the ability to be deployed on edge-computing devices.

2.5 Vehicle Counting

Due to the very similar features observed in some vehicles, ID switches may occur from time to time during the tracking which may result in inaccurate vehicle counts. Therefore, in order to increase the robustness of the vehicle counter by not relying solely on the vehicle tracker and ID assignment for the counting, we introduced a "virtual polygon area" for accurately counting the total number of vehicles that have traveled across the highway. The virtual polygon area works by dividing the scenes into two areas, Zone 1 and Zone 2 as shown in Fig. 6. Zone 1 refers to the region that is located outside of the virtual polygon area, whereas Zone 2 refers to the region that is located within the specified polygon area. The user selects four points which will be used for drawing the virtual polygon area and the vehicle counting will be performed once the vehicles have crossed the highway and their center coordinates enter from Zone 1 into Zone 2 and the vehicle ID assignment remains unique.

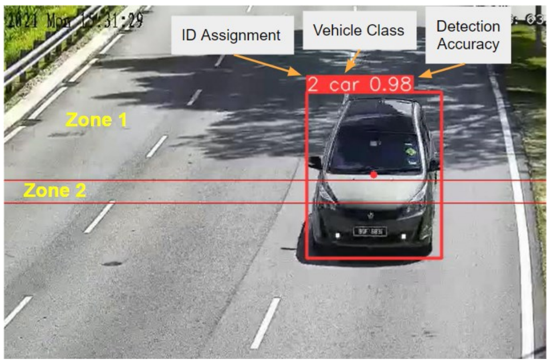


Fig. 6. Virtual Polygon Area used for Vehicle Counting

2.6 Model Deployment on Edge-Computing System

The Jetson Nano edge-computing device based on ARM architecture and produced by Nvidia was used for testing our vehicle counting system. The device is small, approximately 70 x 45 mm in size and brings in the performance of an efficient computer to the edge by use of a Single Board Computer (SBC). It is equipped with a Quad-core ARM Cortex-A57 CPU, 128 Nvidia CUDA Cores and 4GB of RAM with 64-bit LPDDR4 [8]. A Logitech C922 Pro HD Webcam, a 7-inch 1024 x 600 LCD Display and a portable 10,000mAh Mi Power bank were added to the device in order to form an edge-computing system platform. The flowchart used for counting the number of vehicles on the highway scenes is shown in Fig. 7. It works by firstly obtaining the video stream of the vehicles by using the Logitech webcam, and then the information is transmitted to the RAM of the edge-computing device. The Jetson Nano will then use this information as input and run inference of the vehicle detection, tracking, and counting. Finally, the output from the system will be the vehicle detection results as well as the number of vehicles that have crossed the highway shown separately for each of the vehicle classes on the 7” LCD display.

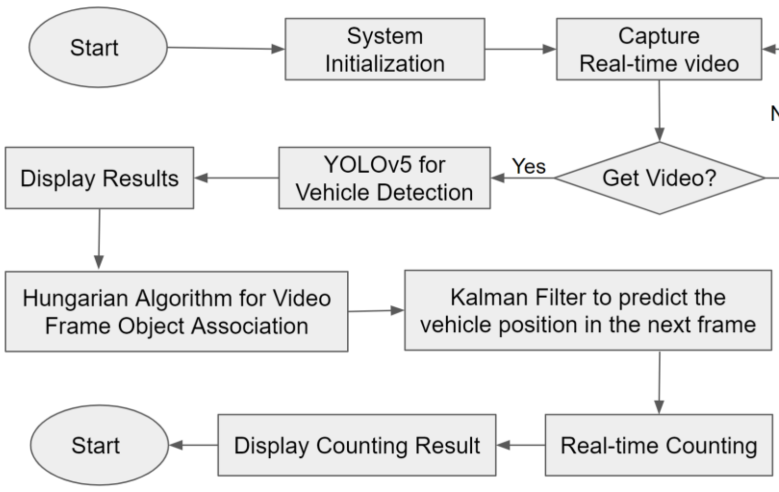

Start
System Initialization
Capture Real-time video
No
Get Video?
Yes
YOLOv5 for Vehicle Detection
Display Results
Hungarian Algorithm for Video Frame Object Association
Kalman Filter to predict the vehicle position in the next frame
Real-time Counting
Display Counting Result
Start

Fig. 7. Vehicle Counting System Flowchart

3. RESULTS AND DISCUSSION

This section presents the performance and evaluation of our proposed vehicle counting system, which was developed by utilizing a custom vehicle detection algorithm and combined with a DeepSORT vehicle tracking algorithm for detecting, tracking, and counting vehicles in highway scenes. It analyzes the performance of the trained vehicle detection model and evaluates the efficiency of the developed vehicle counting method. Furthermore, this section also discusses the vehicle counting experiments that have been conducted by utilizing the edge-computing system platform and live camera inferences obtained by using a Logitech webcam as well as using real-world highway surveillance videos obtained from YouTube. Finally, the results of our vehicle detection model and vehicle counting accuracy scores are presented, discussed, and compared with other similar systems.

3.1 Training Results

The vehicle detection model was trained by utilizing 8,237 images and 2,350 images were used for validation of the model’s performance. The Mean Average Precision, mAP@.5 and the loss function plots were the main metrics used for the evaluation of the trained models. Table 4 presents the model training results obtained after the successful training of four different models. The first model was trained by utilizing the default YOLOv5n model architecture and the original dataset. The second model was trained by utilizing the same default model architecture and by applying data augmentation to the training images. The third model was trained by modifying the YOLOv5n model architecture by adding a small object detection layer and a Convolutional Block Attention Module (CBAM) as well as applying data augmentation to the training images. Finally, the last model was trained by utilizing the same modified YOLOv5n model architecture and the original dataset without applying any augmentations. All of the models were trained for 300 epochs and by utilizing an Nvidia RTX 3060 laptop GPU.

Table 4: Trained Vehicle Detection Models

Data Augmentation	Small Object Detection Layer	CBAM	mAP@.5	Recall	Epochs
			93.2	89.9	300
√			93.6	90.2	300
√	√	√	95.8	92.0	300
	√	√	96.1	91.3	300

The training process for the vehicle detection model requires heavy computation and therefore, we utilized transfer learning techniques and made use of pre-trained weights which have originally been trained on the MS COCO dataset as the starting point for our vehicle detection training process. After the successful training of our highest accuracy model which took 13 hours and 27 minutes to complete, the loss function plots were saved and can be seen in Fig. 8, displaying three main types of losses which are the box, objectness and classification losses. The box regression loss represents how well the algorithm was able to locate the center of the vehicles and how well the predicted bounding boxes covered the detected vehicles. Objectness loss refers to the measure of the probability that a vehicle exists in a predicted region of interest and finally, classification loss refers to the ability of the algorithm to be able to predict the correct vehicle class given a detected vehicle object.

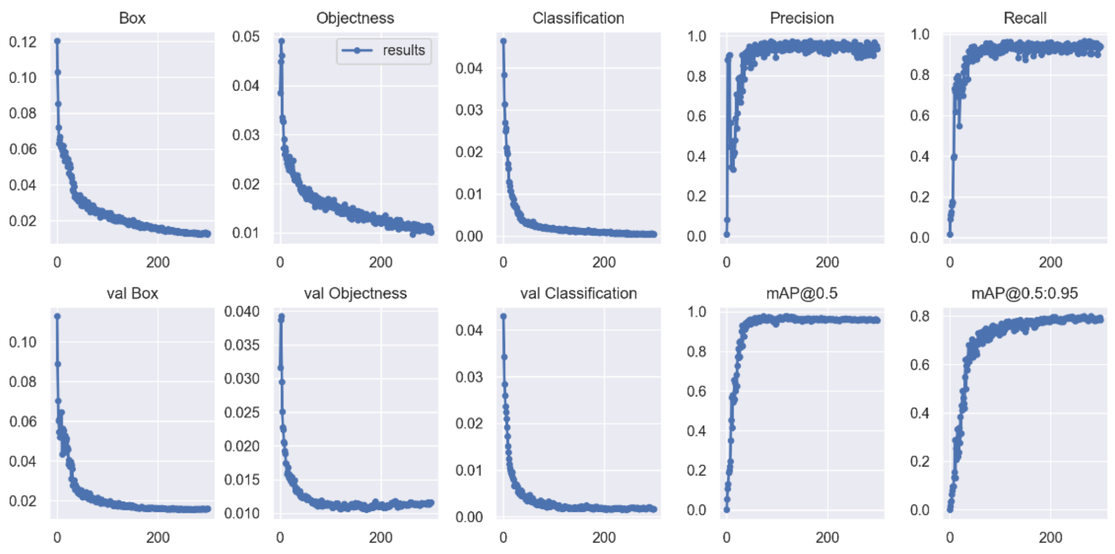


Fig. 8. Model Training Plots over Epochs Results on the Validation set

Precision and Recall are fundamental metrics that are used in object detection to determine whether a trained model is able to make good classifications of detected objects or not. The Precision equation is shown in Eq. (1) and makes use of the True Positive (TP) as well as False Positive (FP) detections in order to determine what proportion of positive vehicle identifications were actually identified correctly. The Recall equation is shown in Eq. (2) and makes use of the True Positive (TP) as well as False Negative (FN) detections to determine what proportion of actual positives were identified correctly. A high Precision and Recall value means that the model is capable of detecting all positive vehicles correctly and that it is able to make accurate classification of the detected vehicles. The Precision-Recall curve for the trained vehicle detection model is shown in Fig. 9.

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \tag{1}$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{2}$$

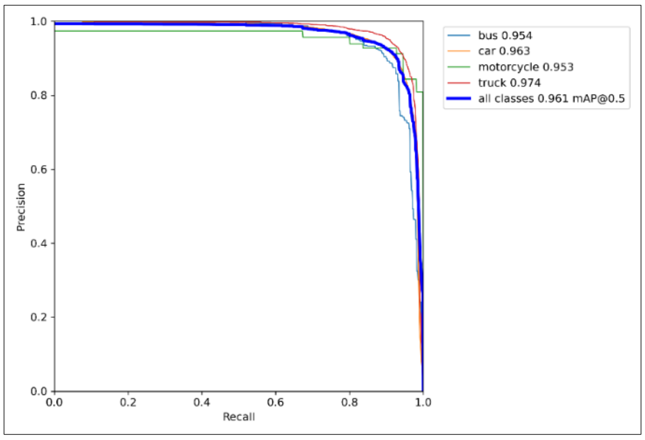


Fig. 9. Precision-Recall Curve

3.2 Experimental Results

An edge-computing system platform based on the Nvidia Jetson Nano was integrated and used for testing of our vehicle counting system in real-world test scenarios. The experimental setup was deployed at the Genting Sempah Highway located in Kuala Lumpur, Malaysia and was situated at a pedestrian bridge located approximately 5 meters above the ground. The Logitech C922 Pro HD webcam was pointed directly towards the busy highway and was used for obtaining the live video stream. The Jetson Nano was used for running inference of the trained models and for performing the vehicle detection, tracking, and counting. Finally, the vehicle detection results along with the vehicle counts for each of the vehicle classes, were displayed on a 7-inch LCD display in real-time. Fig. 10 shows the deployed edge-computing system platform used for testing our system.

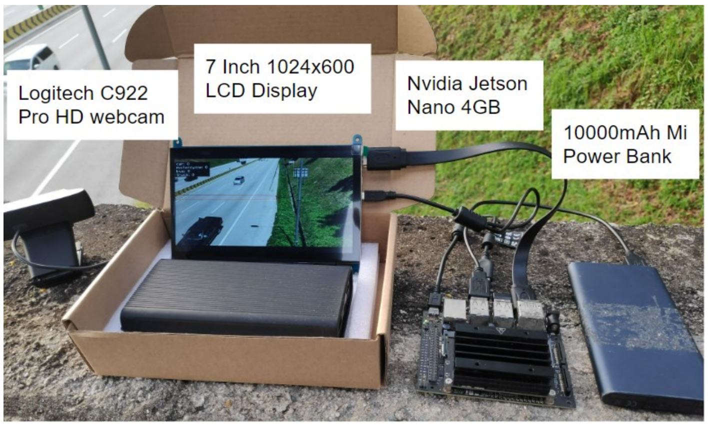


Fig. 10. Edge-computing system platform

Other than the live video inferences, several highway surveillance videos obtained from YouTube were also utilized as input for testing our vehicle counting system. The videos consisted of three one-sided highway videos comprised of various lengths between 0:12 and 01:35 and which are freely available online on YouTube. Table 4.3 summarizes the vehicle counting results obtained for each vehicle class on the video data and the inference results can also be viewed at the following link: https://tinyurl.com/yppcua3d. As can be seen from the inference results, the average speed for running the inference on the 4GB Nvidia Jetson Nano was 15 FPS, close to real-time, and the average vehicle counting accuracy obtained by the system was 95.39%.

3.3 Discussion

The loss function plots were found to have converged significantly faster when using the custom model architecture as compared to when using the default YOLOv5n model architecture. At around 200 epochs, the model had already converged with a loss approximately 9.37% lower than that observed when using the default YOLOv5n model architecture. Furthermore, the mean average precision, mAP@.5 is a true measure of the vehicle detection model performance. Fig. 8 shows that after 100 epochs of training, the mAP@.5 of the model gradually stabilizes, reaching approximately 96.02% and after 300 epochs reaching the highest mAP@.5 score which was 96.1%. Data augmentation was applied to the training images, which helped to increase the Recall of the model and the ability to detect most of the positive vehicles correctly; however, the Precision of the model was reduced which resulted in an increase in the number of false positive predictions.

Table 6: Vehicle Counting Performance of our System

Video Name	Duration (minutes)	Vehicle	Counting		Counting Error	Overall Accuracy
			Real	System		
Genting Highway Traffic Cars, Motorcycle, Trucks	01:35	Car Motorcycle Bus Truck	66 1 0 5	64 1 0 2	+2 0 0 +3	93.06%
Live Motorway Traffic in Aceh Indonesia	00:59	Car Motorcycle Bus Truck	24 4 0 1	23 4 0 2	+1 0 0 +1	93.10%
Sample Video Cars Traffic – 27260	00:12	Car Motorcycle Bus Truck	11 0 0 0	11 0 0 0	0 0 0 0	100%
					Average Accuracy	95.39%

The vehicle detection accuracy score was compared with the original YOLOv5n model architecture as the baseline which had achieved an average mAP@.5 score of 93.2% on the same validation set. The model accuracy score was also compared with other similar vehicle detection models such as [1] which made use of the YOLOv3 network and an ORB algorithm for the detection of vehicles in highway scenes and had obtained an average mAP@.5 score of 87.88%. Another method was that developed by [9] which made use of the YOLOv4 network for the detection of vehicles and achieved an average mAP@.5 score of 82.08%. Our proposed algorithm was able to surpass both of these models in terms of the average mAP score, however, because the data used for testing both of the following models have not been open-sourced, accurate comparisons with our proposed model were not possible. Moreover, our vehicle detection accuracy score was averaged on four vehicle classes, meanwhile, most of the other systems developed for the task of vehicle counting are only trained on a single vehicle class.

With regard to vehicle counting, our proposed vehicle counting method was able to achieve an average vehicle counting accuracy score of 95.39%. The vehicle counting accuracy was compared with other methods such as [10] which made use of a virtual distance measurement line counter for counting cars, motorcycles, buses and trucks similar to our system and had achieved an average counting accuracy of 92.20%. Another method was that developed by [1] which made use of a virtual line counter for the counting of cars, buses and trucks and achieved an average vehicle counting accuracy of 93.20%. Furthermore, not only is our proposed system able to obtain higher vehicle counting accuracy results, but it also consists of a lightweight vehicle detection model and a fast ReID model, achieving an average inference speed of 15 FPS on a compact 4GB Nvidia Jetson Nano and 67 FPS on an RTX 3060 laptop GPU.

4. CONCLUSION

In conclusion, our qualitative analysis has proven that our proposed vehicle counting system which was developed by utilizing a custom vehicle detection algorithm based on the YOLOv5n model architecture, combined with a DeepSORT object tracking algorithm and a lightweight OSNet ReID model, is suitable for the task of vehicle detection, tracking and counting in highway scenes. Furthermore, a custom “virtual polygon area” counting approach was also introduced which allowed for accurate and efficient vehicle counting results and avoided duplicate counts. The following system is lightweight and was able to be deployed in real-world scenarios by utilizing an edge-computing system platform based on the Nvidia Jetson Nano. It was able to achieve excellent vehicle detection as well as counting results and is ready to be implemented in real-world applications.

REFERENCES

[1] H. Song and H. Liang, "Vision-based vehicle detection and counting system using deep learning in highway scenes," vol. 4, 2019.

[2] Z. Luo *et al.*, "Details of the MIO-TCD dataset for vehicle MIO-TCD : A new benchmark classification and localization in press at IEEE Transactions on Image Classification challenge dataset Localization challenge dataset Click here to download the entire dataset :," pp. 1–9, 2018.

[3] G. Jocher *et al.*, "ultralytics / yolov5 : v6 . 1 - TensorRT , TensorFlow Edge TPU and OpenVINO Export and Inference," pp. 4–11, 2020.

[4] P. R. Gunjal, B. R. Gunjal, H. A. Shinde, S. M. Vanam, and S. S. Aher, "Moving Object Tracking Using Kalman Filter," *2018 Int. Conf. Adv. Commun. Comput. Technol. ICACCT 2018*, pp. 544–547, 2018, doi: 10.1109/ICACCT.2018.8529402.

[5] I. C. Vision, "Computer Vision for Multi-Object Tracking — Live Example," pp. 1–17, 2019.

[6] D.-L. Dinh, H.-N. Nguyen, T. Thai, and K.-H. Le, "Towards AI-Based Traffic Counting System with Edge Computing," *J. Adv. Transp.*, vol. 2021, pp. 1–15, 2021, doi: 10.1155/2021/5551976.

[7] F. Herzog, X. Ji, T. Teepe, S. Hörmann, J. Gilg, and G. Rigoll, "Lightweight Multi-Branch Network for Person Re-Identification," *Proc. - Int. Conf. Image Process. ICIP*, vol. 2021-Septe, pp. 1129–1133, 2021, doi: 10.1109/ICIP42928.2021.9506733.

[8] NVidia Corporation, "Jetson NANO Module," 2019, [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano

[9] M. A. Bin Zuraimi and F. H. Kamaru Zaman, "Vehicle detection and tracking using YOLO and DeepSORT," *ISCAIE 2021 - IEEE 11th Symp. Comput. Appl. Ind. Electron.*, pp. 23–29, 2021, doi: 10.1109/ISCAIE51753.2021.9431784.

[10] M. Fachrie, "A Simple Vehicle Counting System Using Deep Learning with YOLOv3 Model," *J. RESTI (Rekayasa Sist. dan Teknol. Informasi)*, vol. 4, no. 3, pp. 462–468, 2020, doi: 10.29207/resti.v4i3.1871.