Vehicle Detection and Tracking using YOLO and DeepSORT

Muhammad Azhad bin Zuraimi
Vehicle Intelligence and Telematics Lab,
Faculty of Electrical Engineering
Universiti Teknologi MARA
40450, Shah Alam, Selangor
azhadzuraimi@gmail.com

Fadhlan Hafizhelmi Kamaru Zaman
Vehicle Intelligence and Telematics Lab,
Faculty of Electrical Engineering
Universiti Teknologi MARA
40450, Shah Alam, Selangor
fadhlan@uitm.edu.my

2021 IEEE 11th IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE) | 978-1-6654-0338-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/ISCAIE51753.2021.9431784

***Abstract*— Every year, the number of vehicles on the road will be increasing. as claimed by a road transport department (JPJ) data in Malaysia, there were around 31.2 million units of motor vehicles recorded in Malaysia as of December 31, 2019. While as, from the mid-2017, there were around 28.18 million units of motor vehicles recorded in Malaysia. Consequently, accurate and fast detection of vehicles on the road is needed by using the volume of vehicles as valuable data for detecting traffic congestion which then benefits for traffic management. Using the implemented deep learning for vehicle detection, this paper project is using TensorFlow which is platform for machine learning and you only look once (yolo) which is object detection algorithm for real-time vehicle detection. By combining this two and other dependencies with python as programming language, the suggested method in this paper determine the improvement of YOLOv4 latest algorithm compared to the previous model in vehicle detection system. This vehicle detection also uses DeepSORT algorithm to help counting the number of vehicles pass in the video effectively. From this paper, the best model between YOLO model is Yolov4 which had achieved state-of-the-art results with 82.08% AP50 using the custom dataset at a real time speed of around 14 FPS on GTX 1660ti. Keywords— Vehicle detection; Deep SORT framework; YOLOv4; Deep learning; Convolution neural network.**

I. INTRODUCTION

In growing country like Malaysia, the number of vehicles on the road keep increasing for various factor such as economic, social, and cultural. Hence, the high number of vehicles on the road will cause traffic congestion [1] . Traffic congestion will increase travel time of the vehicles, high chances of accidents, suffering business with loss of productivity, slow down emergency vehicles which potentially putting lives at risk and air pollution [2]. In Malaysia there were several ideas of smart traffic management system that had been implement. The smart traffic management system should help fast, efficiently improve in safety, and traffic flow on the city. For example, MALAYSIA'S smart traffic system controller Sena Traffic Systems (Sena) that collaborating with Alibaba Cloud had built a smart traffic management system in Malaysia. The system potentially could diminish travel time by 12%, says Alibaba [3].

To calculate the number of vehicles, pass in certain area is troublesome and not efficient with human eyes. Human cannot evaluate every monitor at the same time and cannot focus efficiently all the time. Whereas there were few types of vehicles such as motorcycle, car, bus, and truck, which made it difficult to classifying and counting for human. Therefore, an approachable intelligent traffic surveillance system is required to aid humans attain smart traffic management. The function of this system firstly is to detect vehicle in a video. Then the vehicle will be classified according to the type such as motorcycle, car, bus, and truck. If the vehicle passthrough line added to the video, the system then counts it in the system. The speed and accuracy of the vehicle will be evaluated according to model YOLO with different version. YOLO object detection algorithm is the best to use to detect object with high accuracy in real-time than other deep learning object detection such as DPM and R-CNN [4]. The latest version of YOLO [5] should be the best in accuracy balance with the speed.

As vehicle detection is challenging for human to calculate, this is where a machine learning had great benefits to automatically learn and improve over time tracking, classifying, and counting the number of vehicles passing by certain area. By using darknet [6] which is open-source neural network framework transcribed in C and CUDA. Using C programming language boost speed in computation in CPU and GPU. While as, CUDA which is a parallel computing platform and application programming interfaces model by Nvidia in the GPU improve a lot of speed if working with programming language such as C. this framework is used to train model weight files YOLO which is the deep learning algorithm for vehicle detection. this deep learning algorithm for vehicle detection is state-of-the-art object detection neural network which can be apply in small and large network while maintain speed and accuracy. The model then uses different framework name as TensorFlow.

TensorFlow [7] is an open-source software platform for machine learning and it is free. This framework founded by google brain team can be written in a lot of programming language such as python, C++, CUDA. It also can be use in variety of platform for example are Linux, macOS, Windows, Android, and JavaScript. In this project, it uses python programming language which has lot of useful library such as NumPy. From python, it then code for combining machine

learning, DeepSORT algorithm [8], fps meter code, line bar counting vehicle code, and heatmap vehicles movement to perform this project as vehicle detection system.

II. Related work

In this section, it will explain related work with vehicle detection for traffic management. Nowadays, there were several methods to detect vehicles use for traffic management. It could be traditional machine vision[9] to compound of deep learning methods. Traditional machine vision in vehicle detection usually use the movement of a vehicle to distinct it from static background image [10]. Researchers have projected several traditional methods of vehicle detection right from the start of vehicle detection. The most used features in traditional methods back then are the Oriented Gradient Histogram (HOG) [11] and the Haar-like features [12]. the Oriented Gradient Histogram (HOG) method counts gradient orientation occurrences in localized portions of an image. By taking a rectangular part of an image and splitting the rectangle into multiple sections, a Haar-like feature is depicted. They are also visualized as adjoining rectangles in black and white. This traditional method however often results in high false positive rates.

After several years, a Convolutional Neural Network (ConvNet/CNN) [13]is introduced. A Convolutional Neural Network (ConvNet/CNN) is a deep learning algorithm that can receive an input picture, allocate significance parameter and biases to numerous aspects or objects in the picture and be capable to distinguish one from the other. A Convnet involved less pre-processing than other classification algorithm. With adequate training, this deep learning method had the ability to learn filters and characteristics while as traditional method cannot learn filters and characteristics overtime. This method is inspired from the linked form of neurons in the human brain. This method decreases the computational power necessary to sort out the data which made it even faster than traditional method.

The deep convolutional network CNN had two-stage method [14] which are the method uses various algorithms to create a candidate box of the object and the method to classifies the object through a convolutional neural network. YOLO framework [4] though is the one-stage methods which directly converts the object bounding box positioning issue into a regression issue for processing without generate a candidate box. The YOLO network breaks the picture into a defined number of grids. Each grid is accountable for estimating objects within the grid whose central points are. Then after several years the yolo framework has been developed it algorithm to version 4 which improve speed and accuracy of object detection.

III. Methodology

This section will explain in detail the methodology used within this project and flowchart following this system. It discusses the architecture of the method used to detect vehicle for traffic management. Fig. 1 shows the methodology flowchart following this system works.

A. Installation

Firstly, for this project to be complete, it needs to install dependencies and software following for this project. As this project use windows 10 as platform, Git Bash is installed to the machine to implement Linux code in windows. As mostly reference of this project using Linux and macOS which use Bash shell. Git Bash [15] is a package that install bash, several common bash utilities, and Git on windows operating system. The main programming language for this project is python. in python where its code for combining machine learning, DeepSORT algorithm, fps meter code, line bar counting vehicle code, and heatmap vehicles movement to perform this project as vehicle detection system.

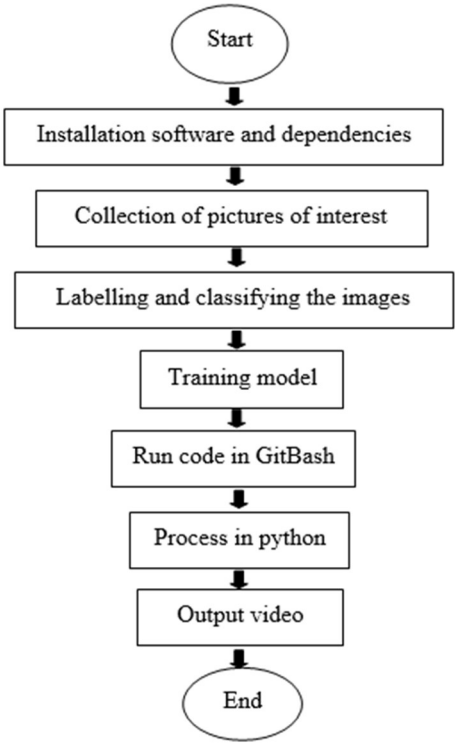

Fig. 1. Methodology flowchart

B. Collection of the images

After all dependencies and software are installed in the machine, this project needs to collect images related to the vehicles to use for training a model for vehicles detection. The number of vehicles in the images, the variety of vehicles types in the images, the occlusion images or not occlusion images, and the number total of vehicles will affect the accuracy of the model it will be used because how model learn from the images given. Figure 2 is occlusion type image whereas figure 3 is no occlusion occur in the image. To collect images of vehicles manually by downloading each image or snap a pic of object interested outside is hurdle, while using internet and application that automatically search for the images that we interested is much better. Using OIDv4 toolkit [16] make this project easier to collect images of vehicles. This toolkit using python

languages can gather images from open images dataset v4 by google which has 600 classes and larger than 1,700,000 images.

C. Labelling and classifying

After collecting a lot of images relating to the vehicles, the images need to be label. The labelling and classifying of the images are hassle if do it normally following to yolov4 format. In yolov4 format [17], .txt-file for each .jpg-image-file must be in the same directory and with the same name. in the txt file, the object number and object coordinate of the related image, must be in line like as: <object-class> <x> <y> <width> <height>. For example, inside the txt file, the format should be as shown in Fig. 4. Using OIDv4 toolkit [16] make labelling and classifying save time by automatically making annotation file in YOLOv4 format in a folder.

Fig. 2. Vehicle image with occlusion

Fig. 3. Vehicle image without occlusion

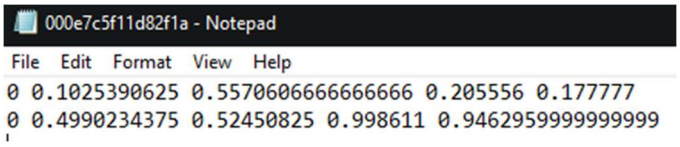

Fig. 4. YOLOv4 format for ground truth labelling

D. Training YOLO model

The images that had been labelled and classified then will be used to train as model in yolo. Using google collab is easier and faster to train model. Google collab [18] is free and write and using python language through the browser. It is extremely fast in collab because it can train model using expensive GPU given by collab. The GPU available by collab is random include Nvidia K80s, T4s, P4s and P100s. the GPU given for example Nvidia Tesla T4 have 16gb of memory size and 2560 CUDA core which can be used to compute complex parallel algorithm for training variety of YOLO model. By comparison training using google collab and personal computer that has GPU GTX 1660ti, google collab can train yolov4-tiny model in 6000 iteration in 4 hours in contrast of the personal computer with same amount of iteration and model in 8 hours. Using google collab notes ready by AIGuysCode [19] in GitHub, training the model is easier and organized. Just follow instruction and understand the flow of the coding, the training will be run in collab for several hours.

E. Run code in GitBash

Next when several different models are ready, the model will run in GitBash by calling python file. From this step, it required user input to the system. Firstly, in GitBash environment, user can install the dependencies according to what environment need and interest computation for running this system by the user. User can install dependencies in Pip python environment or Conda environment. In this project, it used Conda environment because Conda can easily creates, saves, loads and switch between environment on the local computer and had installed most basic library. Then we need to setup between CPU or GPU environment. GPU environment should be faster and recommended if computer had one GPU or more. In this system use GPU environment in which will install dependencies for instance tensorflow-gpu, OpenCV, lxml, tqdm, absl-py, matplotlib, easydict and pillow. The only difference between CPU and GPU environment is TensorFlow library package while other dependencies is same. Next, user need to change the model get from training into TensorFlow model. As the original model yolov4 is in darknet framework which written in c in Linux platform. Converting darknet model to TensorFlow helps running this system on python in windows platform efficiently. The other name for this type of converting is DarkFlow. Lastly, user can run the object tracker by input specified the type of input video, type of weight use, type of framework, tiny or normal weight type and output video file for example in Fig. 5. The calculation and algorithm will be run in the python which it will describe in the next section.

```
# Convert darknet weights to tensorflow model
python save_model.py --model yolov4

# Run yolov4 deep sort object tracker on video
python object_tracker.py --video ./data/video/test.mp4 --output ./outputs/demo.avi --model yolov4

# Run yolov4 deep sort object tracker on webcam (set video flag to 0)
python object_tracker.py --video 0 --output ./outputs/webcam.avi --model yolov4
```

Fig. 5. User input

F. Python program development

In this section, it will explain about how inside python coding role. It consists the main algorithm in this system such as setting of DeepSORT, frame-by-frame extract, what object to be detect, searching for region of interest (ROI), defining confidence score to be classify and boxing, filtering class to be detect, the performance of the system in frame per second and counting the vehicle pass if pass line of interest. DeepSORT is

extension of SORT algorithm [8]. Sort is simple online real-time tracking consist of 4 core components which are detection, estimation, association, and track identity creation and destruction. The estimation uses the framework of Kalman filter [20] whereas it will predict better even the object is occluded. The target association use Hungarian algorithm [21]. This algorithm will compute the intersection-over-union (IOU) distance between each detection and every predicted bounding box from the existing targets. The last algorithm work in SORT algorithm will assign either create unique identities or destroyed it. In DeepSORT there will be deep learning algorithm which helps reduces high number of identity switches and improve efficiency of tracking through occlusions in SORT algorithm.

G. Output video

After the system done calculate and process the video, the video output will be saved in specific place according to user input. The result of the video will have the bounding box if vehicles is detected in each frame, the fps on the top left side on the video, and the line of interest to calculate vehicles pass through it.

IV. RESULTS AND DISCUSSION

A. Datasets

To get the weight files, the datasets of pictures and classify of pictures must been obtained and organize which then should been trains into weight files. In this project, I had collected mostly around 1500 images each of different classes types which are cars, motorcycles, bus, and truck. All the training dataset is 7319 images dataset which been downloaded from google. And 750 images more datasets for validation datasets in each class's types. As we can see in Fig. 6 is the total images us for building our custom model. This validation images datasets are used to calculate maps for the training weight files. The validation datasets ratio between all datasets used is 30%. In this project, using around 30% of its recommendation from article.

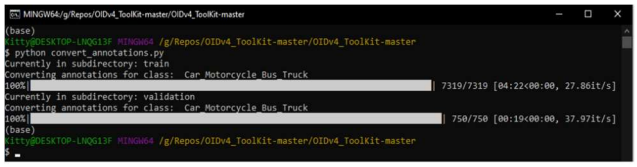

Fig. 6. The amount of pictures use in custom model

B. Weight

Weight is very important parameter inside neural a network. To make a good judgment for finding and classifying either the frame in the video had a vehicle or not is mostly based by weight file. As we know that the neural network is basically set of inputs, weight, and bias value. When an input reaches the nodes. It gets multiplied by a weight value and the value obtain then pass on to the next layer of the neural network or been observed. Sometimes the weights of neural network also had in the hidden layer. The weight file can be different in size according to the model architecture, weights, training configuration just like as loss function and optimizer, and state of optimizer to resume training directly from last checkpoint.

TABLE I. WEIGHT FILE MODEL SIZE

Weight file model	Source	File Size
yolov480.weights	AlexAB	251MB
yolov480-tiny.weights	AlexAB	23MB
yolov380.weights	Pjreddie	242MB
yolov380-tiny.weights	Pjreddie	34MB
yolov4.weights	custom	250MB
yolov4-tiny.weights	custom	22MB
yolov3.weights	custom	250MB
yolov3-tiny.weights	custom	33MB

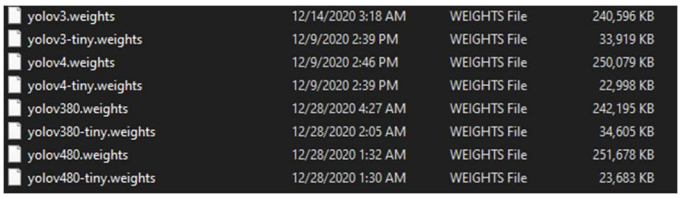

Fig. 7. Training weights file

From Table 1 and Fig. 7, it is shown that different models had different size. Yolov3 weights file is smaller than yolov4 weights file. While as, yolov3 tiny weights file is bigger than yolov4 tiny weights file. Size could be important in certain aspect such as using a weight files for smaller machine storage size like as raspberry pi or android phone.

C. Mean Average Precision

After weight file is ready, it can be evaluated to measure the quality of the model. mAP (mean Average Precision) is an evaluation metric combine of recall and precision for object detection. to measuring mAP (mean Average Precision) we need to reevaluate classification and localization. Classification is to recognize if an object which is vehicle in our project is present in the image and the class of the object as shown in Fig. 8. While as, localization is to predict the coordinates of the bounding box around the object when an object is present in the image just as shown Fig. 9.

Fig. 8. Classification example

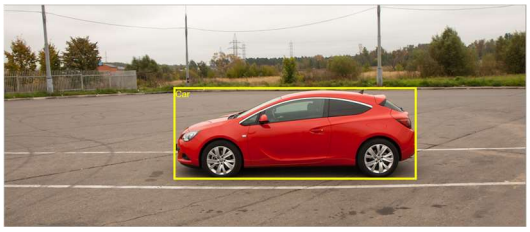

Fig. 9. Detection (localization) example

Object detection use perception of Intersection Over Union (IOU) to measuring intersection between two boundary which are predicted boundary vs ground truth boundary box. The equation is mention in Fig. 10. From Fig. 12, red boundary box is the predicted box whereas, yellow is the ground truth boundary box. For object detection missions, it should compute mean Average Precision (mAP) using IoU value for a given IoU threshold. For example in Fig. 11, if iou threshold is set 0.5, and the value given by the model is higher than 0.5 we count it as true positive. While, the value given by the model is lower than 0.5 is count as false positive as example in Fig. 12.

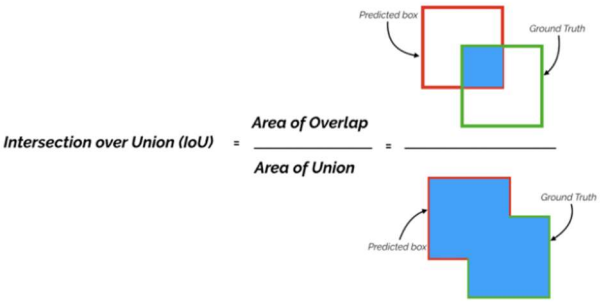

Fig. 10. IOU definition

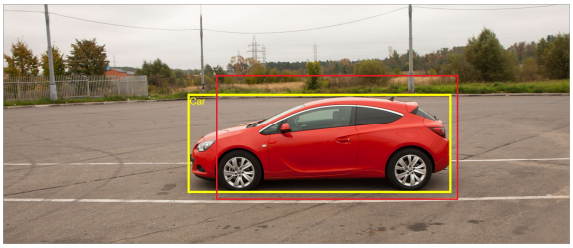

Fig. 11. Example of detection with IOU higher than 0.5

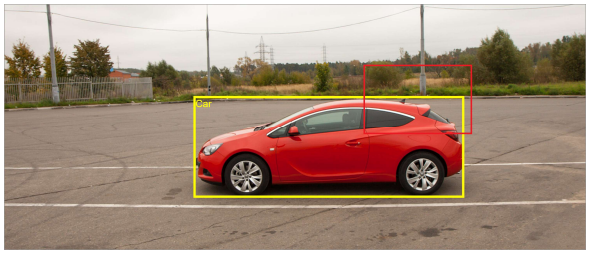

Fig. 12. Example of detection with IOU lower than 0.5

Depending on the various detection challenges that occur, the mean Average Precision or mAP score (1) is determined by taking the mean AP for all classes and/or general IoU thresholds. Annotation files can be in COCO or Pascal VOC data format for calculating mAP. COCO has 1.5 million object instances for 80 object categories. COCO stores annotations in a JSON file, while PASCAL use XML file. COCO needs to set an entire dataset for training, testing, and validation in single file, while Pascal, need to create a file for each dataset for training, testing, and validation. In the PASCAL VOC2007 challenge, an IoU threshold of 0.5 is determined for an AP for one object class. Thus, for all object classes, the mAP is averaged. In the COCO challenge to compute the general AP for the coco dataset, it requires to loop the calculation function for IOU 0.5 or 0.95[.50:.95] by 9 times.

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \quad (1)$$

The custom models use the COCO challenge format with IOU of 0.5 with custom validation datasets consists around 3000 of images. While AlexAB models use the COCO challenge format with IOU of 0.5 on Coco validation dataset 2017 which consist of 5000 images. Pjreddie models use the COCO challenge format with IOU of 0.5 on Coco test dataset 2017 which consist of 41000 images. Using different real-time single-stage object detection model which are yolov3, yolov3 tiny, yolov4 and yolov4 tiny to detect vehicles in video for mAP. Different detection model, classes, and validation dataset use to evaluate mAP, and the validation dataset will affect the accuracy and speed of the model. From a paper by AlexAB, it said in the paper that Yolov4 has 65.7% AP@50 from using official coco validation datasets 2017 which has around 1 GB size of images. While as, from a paper by Pjreddie, it said in the paper that Yolov3 has 57.9 AP@50 from using official coco test datasets 2017 which has around 6GB size of images. While as, our custom datasets which consist of 4 types of classes which are car, truck, motorcycles, and bus has around 250mb only. The calculation of mAP is too bias if we want to differentiate it between our custom model with AlexAB model and Pjreddie model. However, we can see on the Table 2, comparing the different custom model with same validation dataset is fair. From the Table 2, the accuracy of yolov4 is much better than yolov3 and yolov4-tiny also had greater accuracy than yolov3-tiny.

TABLE II. MAP PERFORMANCE FOR TESTED YOLO MODELS

Model	**mAP @0.5**	**Classes**	**Validation dataset**
YOLOv4(custom)	82.08%	4	Custom dataset
YOLOv4-tiny(custom)	76.14%	4	Custom dataset
YOLOv3 (custom)	80.32%	4	Custom dataset
YOLOv3-tiny(custom)	66.03%	4	Custom dataset
YOLOv4 (AlexAB)	67.9%	80	Coco validation dataset 2017
YOLOv4-tiny (AlexAB	40.2%	80	Coco validation dataset 2017
YOLOv3 (Pjreddie)	57.9%	80	Coco test dataset 2017
YOLOv3-tiny(Pjreddie)	33.1%	80	Coco test dataset 2017

To calculate the mean average precision of the model, the custom validation dataset use must unique and should not had in the training dataset for fair result calculation of mAP. However, in the case of our own dataset, the amount of dataset is small and may be different a lot than actual uses. Additionally, most of the prediction of the vehicles should be faces front of a vehicles or faces behind of the vehicles as our project using vehicle detection on the highway camera.

.

D. Performance of model

The performance of the model use can be measured using frame per second from the video process by the model. The video will be put in the python coding which use TensorFlow as framework to evaluate the speed of the model to detect the object from the video input. After it had been done processing the detection of the object by the model given to the framework, it will then save the video output on the save file. Table 3 shows the results obtained during 6s frame on video name cars.mp4. the hardware use for this experiment will impact the speed of the model. The graphic card use in this project is GTX 1660ti, CPU is intel i5-6500, 12 Gb of Ram at 2133Mhz and storage is Kingston A400 250gb SSD. As we can see in the table, the yolov4 has 14.12 fps which is lesser than yolov3. Whereas, yolov3-tiny is faster around 12.66 more fps than yolov4-tiny. The drawback of having better accuracy is lesser speed from conclusion in this table. It is because the model is more complex to detect the object for better accuracy. Presented here are Fig. 13, Fig. 14, Fig. 15, and Fig. 16 as a proof of the performance of the model from the video output given. At the videos, it can be seen the fps meter on the top left corner.

TABLE III. PROCESSING PERFORMANCE FOR TESTED YOLO MODELS

Model	FPS
YOLOv4	14.12
YOLOv4-tiny	40.11
YOLOv3	16.99
YOLOv3-tiny	52.77

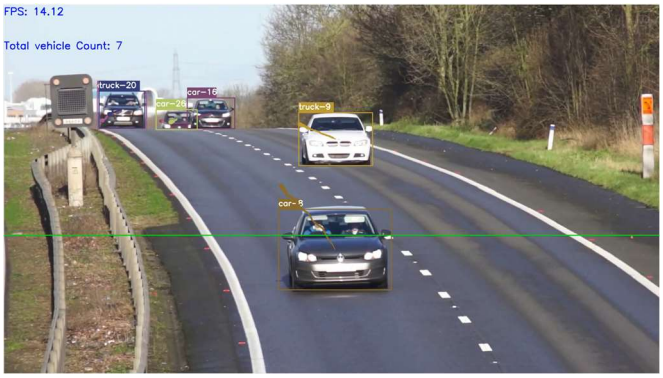

Fig. 13. YOLOv4 model cars video detection

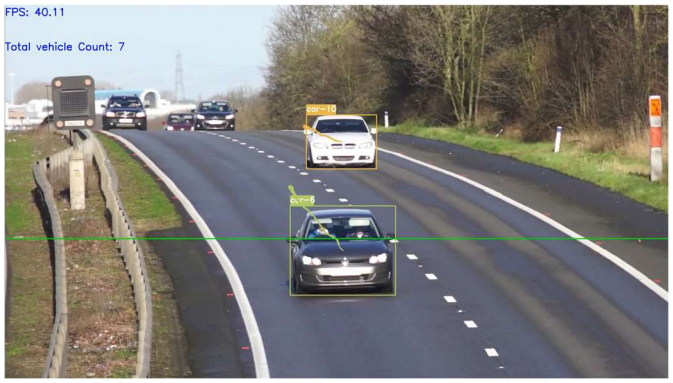

Fig. 14. YOLOv4-tiny model cars video detection

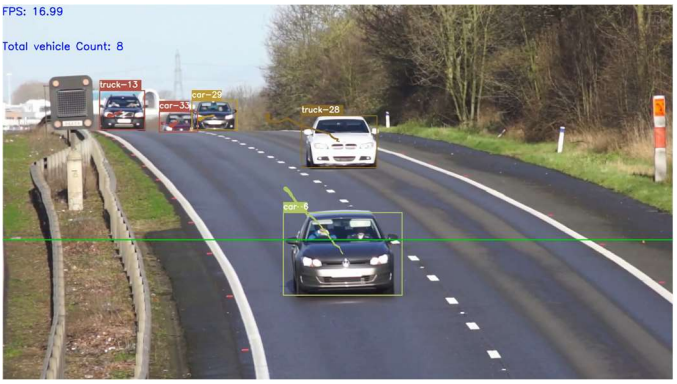

Fig. 15. YOLOv3 model cars video detection

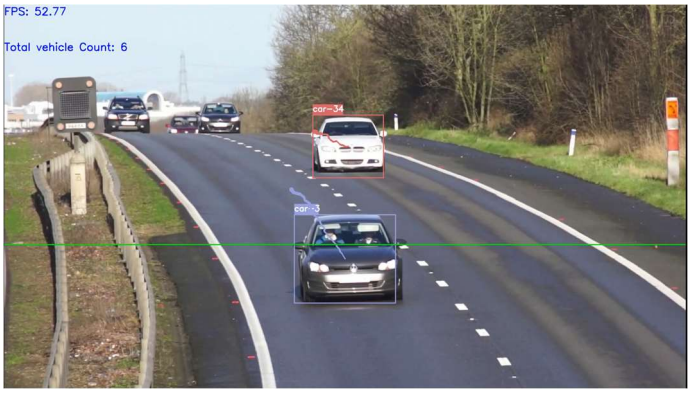

Fig. 16. YOLOv3-tiny model cars video detection

V. CONCLUSIONS

In summary, this vehicle detection and tracking method presented uses TensorFlow library with DeepSORT algorithm based on Yolov4 model. It can be proven that using Yolov4 and yolov4-tiny is acceptable and faster than previous one. It can be use in Realtime surveillance camera in the highway or recording video to evaluate the number of vehicles pass by according to what time it started recorded to last recorded. This data then can be used for traffic management by implementing answer if the place proven a lot of congestion or not. From the result obtained in this project, it is the best to use YOLOv4

model than previous model YOLOv3 if the system wants the highest accuracy with acceptable speed which that could achieve at 82.08% mAP@0.5 higher 2% than previous model in mAP@0.5. while as, if the system wants the best accuracy with the highest speed as possible because limitation in hardware or to process it in real-time, it is recommended to use YOLOV4-tiny model which it can achieve 76.14% mAP@0.5 higher 10% more mAP@0.5 than YOLOV3-tiny with performance of speed as 40fps using GTX 1660ti.

This system can be improved to be more adaptable for vehicle detection if using several suggestion ideas mention in this section. The first idea is using raspberry pi as the platform to put it in small places at the surveillance camera that detect vehicle passing such as in highway. This will make it more suitable to put than big machine at that place. Using the cloud computation to use as video tracking could be better and faster by using expensive GPU for computing this system. Also, it will make the computer use for running this system use lesser and small component as no GPU need in the system like as raspberry pi.

ACKNOWLEDGMENT

The author would like to thank Ministry of Education for the FRGS grant (600-IRMI/FRGS 5/3/ (081/2019)) and Faculty of Electrical Engineering, Universiti Teknologi MARA for the support.

REFERENCES

[1] Anthony Lim, "Vehicles Registrations in Malaysia - 31.2 mil as of 2019," *Paultan.org*, Apr. 02, 2020. https://paultan.org/2020/04/02/vehicles-registrations-in-malaysia-31-2-mil-as-of-2019/#:~:text=The transport ministry says that,last year%2C as Bernama reports. (accessed Jan. 08, 2021).

[2] T. Afrin and N. Yodo, "A survey of road traffic congestion measures towards a sustainable and resilient transportation system," *Sustain.*, vol. 12, no. 11, pp. 1–23, 2020, doi: 10.3390/su12114660.

[3] M. D. Chow, "Alibaba and Sena to develop smart traffic solution to ease congestion," *May 24, 2019 12:02 AM*. https://www.freemalaysiatoday.com/category/nation/2019/05/24/alibaba-and-sena-to-develop-smart-traffic-solution-to-ease-congestion/ (accessed Feb. 05, 2021).

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 779–788, 2016, doi: 10.1109/CVPR.2016.91.

[5] A. Bochkovskiy, C. Y. Wang, and H. Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv*, 2020.

[6] J. Xiong, W. Cui, W. Zhang, and X. Zhang, "YOLOv3-Darknet with Adaptive Clustering Anchor Box for Intelligent Dry and Wet Garbage Identification and Classification," *Proc. - 2019 11th Int. Conf. Intell. Human-Machine Syst. Cybern. IHMSC 2019*, vol. 2, no. 99, pp. 80–84, 2019, doi: 10.1109/IHMSC.2019.10114.

[7] Z. Zeng, Q. Gong, and J. Zhang, "CNN model design of gesture recognition based on tensorflow framework," *Proc. 2019 IEEE 3rd Inf. Technol. Networking, Electron. Autom. Control Conf. ITNEC 2019*, no. Itnec, pp. 1062–1067, 2019, doi: 10.1109/ITNEC.2019.8729185.

[8] M. I. H. Azhar, F. H. K. Zaman, N. M. Tahir, and H. Hashim, "People Tracking System Using DeepSORT," *Proc. - 10th IEEE Int. Conf. Control Syst. Comput. Eng. ICCSCE 2020*, no. August, pp. 137–141, 2020, doi: 10.1109/ICCSCE50387.2020.9204956.

[9] J. Wei, J. He, Y. Zhou, K. Chen, Z. Tang, and Z. Xiong, "Enhanced Object Detection with Deep Convolutional Neural Networks for Advanced Driving Assistance," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 4, pp. 1572–1583, 2020, doi: 10.1109/TITS.2019.2910643.

[10] V. Murugan, V. R. Vijaykumar, and A. Nidhila, "A deep learning RcNn approach for vehicle recognition in traffic surveillance system," *Proc. 2019 IEEE Int. Conf. Commun. Signal Process. ICCSP 2019*, pp. 157–160, 2019, doi: 10.1109/ICCSP.2019.8698018.

[11] S. Zhang and X. Wang, "Human Detection and Object Tracking Based on Histograms of Oriented Gradients," *2013 Ninth Int. Conf. Nat. Comput.*, pp. 1349–1353, 2013.

[12] B. F. Momin and T. M. Mujawar, "Vehicle detection and attribute based search of vehicles in video surveillance system," *IEEE Int. Conf. Circuit, Power Comput. Technol. ICCPCT 2015*, pp. 1–4, 2015, doi: 10.1109/ICCPCT.2015.7159405.

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pp. 580–587, 2014, doi: 10.1109/CVPR.2014.81.

[14] X. Hu *et al.*, "SINet: A Scale-Insensitive Convolutional Neural Network for Fast Vehicle Detection," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 3, pp. 1010–1019, 2019, doi: 10.1109/TITS.2018.2838132.

[15] "Git bash: Definition, commands, & getting started | Atlassian," *atlassian*, 2019. https://www.atlassian.com/git/tutorials/git-bash (accessed Jan. 08, 2021).

[16] A. Vittorio, "Toolkit to download and visualize single or multiple classes from the huge Open Images v4 dataset," *GitHub repository*. 2018, Accessed: Jan. 08, 2021. [Online]. Available: https://github.com/EscVM/OIDv4_ToolKit.

[17] Alexey, "GitHub - AlexeyAB/darknet: YOLOv4 - Neural Networks for Object Detection (Windows and Linux version of Darknet)," 2020. https://github.com/AlexeyAB/darknet (accessed Jan. 08, 2021).

[18] N. I. Hassan, N. M. Tahir, F. H. K. Zaman, and H. Hashim, "People Detection System Using YOLOv3 Algorithm," *Proc. - 10th IEEE Int. Conf. Control Syst. Comput. Eng. ICCSCE 2020*, no. August, pp. 131–136, 2020, doi: 10.1109/ICCSCE50387.2020.9204925.

[19] TheAIGuysCode, "GitHub - theAIGuysCode/yolov4deepsort," *GitHub repository*. https://github.com/theAIGuysCode/yolov4-deepsort (accessed Jan. 08, 2021).

[20] V. Murugan and V. R. Vijaykumar, "Automatic Moving Vehicle Detection and Classification Based on Artificial Neural Fuzzy Inference System," *Wirel. Pers. Commun.*, 2018, doi: 10.1007/s11277-018-5347-8.

[21] B. Sahbani and W. Adiprawita, "Kalman filter and iterative-hungarian algorithm implementation for low complexity point tracking as part of fast multiple object tracking system," *Proc. 2016 6th Int. Conf. Syst. Eng. Technol. ICSET 2016*, pp. 109–115, 2017, doi: 10.1109/FIT.2016.7857548.