A COMPARITIVE STUDY OF CONVOLUTIONAL NEURAL NETWORK MODELS FOR DETECTION, CLASSIFICATION AND COUNTING OF VEHICLES IN TRAFFIC

Azizi Abdullah and Jaison Raj Oothariasamy
Faculty of Technology and Information Science, Universiti Kebangsaan Malaysia,
azizia@ukm.edu,my, jaisonhealer18@gmail.com

Abstract: *Deep Learning based networks especially Convolutional Neural Network (CNN) models are widely used in vehicle detection, classification and counting system. On the other hand, transfer learning is a process of re-using a trained model to solve a problem similar to the one it was trained. Two ways of implementing transfer learning are direct usage of a model as a classifier and usage of a pre-trained model as a weight initialization for training with a new dataset. With recent development in the field of deep learning, many CNN models and architectures are available which makes the selection of a suitable model for performing vehicle detection, classification and counting a big challenge. Besides that, a tracking method is also required to track the vehicles in the video sequences so that the counting can be done as accurate as possible. In this project three types of CNN models i.e. SSD Inception, Faster R-CNN ResNet and Yolo DarkNet were tested on 10 traffic video samples using transfer learning methods (classifier). Those three models were selected for comparison based on the popularity and availability of many research studies in recent time. The models were compared based on number of vehicles detected, accuracy and processing time. A simple vehicle tracking method was also developed to aid the counting process. Results showed that Yolo DarkNet was the best performing model with an average accuracy of 66.29 % and average processing time of 0.264 seconds per-frame. It also achieved the highest accuracy of 96.32% in bright conditions. SSD Inception was the fastest at 0.135 seconds per-frame but the average accuracy was lowest at 14.53%. However, all three models performed poorly in low light conditions where Yolo DarkNet performed between 2.8% to 3.5% accuracy. To overcome this problem, another method of transfer learning (weight initialization) was used to re-train the Yolo DarkNet model with annotated images from low light traffic videos. The re-trained model achieved a maximum of 76.73% in accuracy in low light scenario.*

Keywords: *CNN, Transfer Learning, Deep Learning, Vehicle Detection*

I.INTRODUCTION

Rapid increase of road traffic density in major cities has raised the need for automated vehicle monitoring system specifically for traffic measurements and forecasting purposes. Generated statistics are used in road infrastructure planning, such as road widening, traffic intersection and fly-overs construction, foot bridges, underpasses, pedestrian paths building and adding motorcycle lanes.

In earlier days before the rise of machine learning, the vehicle counting was done manually. Counting was done by a person standing by the road side; using an electronic device to record the data using a tally sheet. In some cases, the person may do the counting by observing video footage captured by citycams or CCTVs placed above the road or highway. According to a study by Zheng et al. (2012), manual vehicle calculation is 99% accurate. This investigation is based on a manual calculation of the vehicle from a 5 minutes video recording. It was found that calculation errors are usually less than 1% while classification errors are more prominent, with an average of 4% to 5% error rate. Although manual method provides high accuracy, it requires huge amount of man power. Therefore, manual calculations are usually performed with only a small sample of data and the results are extrapolated for the whole year or season for long-term forecasts. Recent advancement in the field of computer vision have enabled the development of automated vehicle counting systems with the similar accuracy as human calculations. It can also be used for continuous monitoring and counting over time with less human intervention. Computer vision-based vehicle calculation methods have several advantages over manual or other automated calculation methods.

Two important advantages are cost and flexibility. This method is cost effective as it can count many instances at once or parallelly which means only one camera is required for multiple lanes or intersections. Secondly is flexibility. It is flexible to add or change areas where vehicles need to be counted by using filters to determine lanes or paths that need to be counted.

In general, the process of calculating vehicles based on computer vision can be divided into three stages: vehicle detection, tracking and counting. The first step, detection of vehicles can be accomplished with either Traditional Method or Deep Learning method. Traditional object detection methods are built on handcrafted features with shallow design that can be trained. However, when complex features are constructed with a combination of low-level image features and high-level contexts of object detection and scene classification, traditional model performance will stall (Zhao et al., 2019). Some of the most commonly used handcrafted methods are Background Subtraction, Optical Flow, SIFT (Scale Invariant Features Transform) and HOG (Histogram of Oriented Gradients). On the other hand, Deep Learning techniques learn categories gradually through the hidden layers in its architecture. For example, in face image recognition, it starts with a low level to identify bright and dark areas, then recognizes lines and shapes for facial recognition. Each neuron or node in the network represents one feature or aspect and together they give a full representation of the image. Each hidden node or layer is represented by a weight value that will influence the outcome (output) and this value can be changed during the learning process. In traditional method, many of the features used should be identified to reduce data complexity and simplify the learning process of algorithms with a clearer pattern. The advantage of the deep learning method is that it tries to learn the high-level features of the data in stages, so the extraction of the features does not need to be encoded. For an example in case of multiple object detection problems, deep learning techniques such as YOLO (You Only Look Once) (Joseph et al., 2015) capture images as inputs and provide the location and name of objects on output. But in traditional method that uses algorithms like SVM, a bounding box object detection algorithm is required first to identify all possible objects to have the HOG as input to the learning algorithm in order to recognize relevant objects. (Kashyap et al., 2019)

Deep learning has its own disadvantage too. The problem with deep learning is the reliance on large training datasets for networks to learn patterns in data (Chuanqi Tan et al., 2018). Therefore, a large dataset needed if a deep learning network needs to be trained from the scratch which is costly and time consuming (Pan et al., 2010). To overcome this problem, transfer learning may be used. Transfer learning is method to make use of the knowledge gained while solving one problem and applying it to a different but related problem. For example, knowledge gained while learning to recognize cars can be used to some extent to recognize trucks. This method saves training time and requires only a small amount of training data compared to training the model to recognize truck from the beginning. There are four ways to reuse a trained model (Jason Brownlee, 2019): as a classifier, standalone feature extractor, integrated feature extractor or used for weight initialization. With recent rapid innovations in architecture and training of convolutional Neural Networks, there are many pre-trained models made available on public domain with a permissive license for general use. These CNN models can be used by means of transfer learning (classifiers) to implement the vehicle detection.

The general availability of many pre-trained deep learning models might ease the implementation of an automated vehicle counting system, but the main challenge is to identify the best pre-trained models that can fit to a given custom dataset. The direct comparisons between them are difficult due to different base feature extractors, different default image resolutions and different hardware and software platforms that maybe used to develop the system. Each model may have its own advantages and disadvantages.

Some studies have been done to compare various available CNN models as detector in general such as Huang et. al (2016), Alvaro Arcos-Garc'ıa et. al. (2018), and Nikhil et al. (2017) to name few. There are also studies specifically on using deep learning models for vehicle counting systems such as Arinaldi et al. (2018), Singh Chauhan, Mayank et. al. (2019)

and Dey Bhaskar et al., (2019). Each study has varying results which highlights the strengths and weaknesses of each pretrained models. It seems the performance of models are greatly associated with the local dataset and the characteristics of the vehicle movement. Thus, there is no one CNN model that fits all situation as to provide the optimal detection result.

The next step after vehicle detection is the tracking phase. Tracking is important to identify the same vehicles that appear on consecutive frames as not to be counted more than once during the counting phase. An efficient method to track the vehicle need to be constructed. Finally, the selected model must be improved. This is necessary because the models used to develop the counting system is pre-trained with generally available datasets such as COCO dataset, ImageNet dataset or VOC dataset. This may cause the models to overfit and perform poorly in certain conditions mainly on poor lighting situations as mentioned by Dey Bhaskar et al., (2019).

This paper will address three main issues in construction of an effective vehicle counting system. Firstly, the selection of the best model to be use on the local custom dataset must be determined by means of comparison between few shortlisted models. Next is to address the vehicle tracking method to aid the counting mechanism and finally improvising the performance of the selected model to overcome the overfitting problem due to dataset used to pre-train the model. The improvisation is done using weight initialization method as mentioned by Jason Brown Lee (2019).

II. RELATED WORK

One of the biggest challenges in the field of applied Convolutional Neural Network is to identify the best architecture for implementation. The best CNN architecture must be able to provide the best results by accuracy and use efficient computation technics (Aghdam et al., 2017). This is because the accuracy metrics such as the mean Average Precision (mAP) might not provide a holistic picture. Other important metrices such as execution time and memory usage must be considered as well. Many researches are done to ease this selection by providing comparative studies on various CNN architectures.

In one study by Jonathan Huang et al., (2016), a comparison is presented on different combinations of meta-architectures (Faster R-CNN, R-FCN and SSD) with feature extractors (Inception Resnet V2, inception v2, Inception v3, MobileNet, ResNet101 and VGG). They used a high-end hardware with 32GB RAM, Intel Xeon E5-1650 v2 processor and a Nvidia GeForce GTX Titan X GPU card. Timings were reported on GPU for a batch size of one. The detection pipeline for all models (SSD, Faster R-CNN and R-FCN meta-architectures) were constructed on Tensorflow platform. This allows easy swapping of feature extractor architectures, loss functions. It also allows for easy portability to various other platforms for deployment. The performance of the models varies in terms of detection accuracy, GPU utilization time and memory utilization. The study also identified sweet spots on the accuracy/speed trade-off curve where gains in accuracy are only possible by sacrificing speed. Two CNN architecture that fit into such sweet spot was R-CNN with ResNet101 and R-FCN with ResNet 101.

Yadav et al., (2017) present a comparative study of CNN models using deep learning library of Tensorflow due its portability and ease of use. They used the COCO dataset for evaluation. The hardware used was Nvidia Titan X GPU card, on a 32 GB RAM device with Intel Xeon E5-1650 v2 processor. The images were resized, and the dimensions were either 300X300 or 600X600 pixels. The timings were averaged for 450 images. In this study, they found that SSD and R-FCN models are faster in processing compared to other CNN models. Meanwhile Faster R CNN was the slowest in detection where it took 100ms to process each image. But it was the one with highest accuracy.

Arcos-Garc'ıa et al., (2018) presented another comparison which was among combinations of four meta-architectures ((Faster R-CNN, R-FCN, SSD, dan YOLOv2) and six different extractors (Resnet V1 50, Resnet V1 101, Inception V2, Inception Resnet V2, Mobilenet V1, dan Darknet19). The experiments were using dataset which consists of street signboards. The study concluded Faster R-CNN Inception ReNet v2 as the best mAP. Yolov2 came second and R-FCN ResNet 101 was mentioned as a balanced architecture between accuracy and execution time. The advantages of Yolov2 against other models was stressed by Du (2018). In his research, he mentioned Yolov2 achieving 76.8mAP at 67FPS and 78.6 mAP at 40 FPS beating other models based on Faster RCNN and ResNet.

With the availability of pre-trained models, recently there has been many research and studies to compare the performance of deep learning methods and handcrafted methods for vehicle detection and counting systems as well. A study by Hardjono et al., (2018) presents a comparative investigation of vehicle counting using various machine learning methods and specifically compares handcrafted method with deep learning method. The comparison was done between Background subtraction, Viola Jones and Yolo using four datasets which were taken from existing CCTV image data. Quantitative evaluation, F1 and precision scores were obtained. Based on experiments, F1 scores for vehicle counting ranging from 0.32 to 0.75 have been successfully obtained for one low resolution dataset by using Background Subtraction and Viola Jones methods. This study also mentioned that Viola Jones method can improve F1 score, by about 39% to 56%, over Back Subtraction method. However, the use of YOLO yielded even better results, with F1 scores ranging from 0.94 to 1 and its precision ranges from 97.37% to 100%.

Another study by Arinaldi et al., (2018) concluded that the Faster RCNN is more suitable for the problem of detection and classification of vehicles in a dynamic traffic scene with moving vehicles. In this paper, a comparison of performance was done on system based on MoG background subtraction with SVM classifier and a system based on the Faster RCNN. A set of annotated images with six class labels were used to train the Faster RCNN model. The results of the experiments show that MOG with linear SVM produce highest counting accuracy of 59.1% and Faster R-CNN achieving 69.4%. It is also found that detection and classification of vehicles for night time data poses a challenge for the MoG background subtraction. This is mainly due to bright lights and various reflections on the traffic scene. On the other hand, Faster RCNN does not show such problem because it was trained to detect vehicles based on appearances and not based on the changes in pixel value which is sensitive to lighting and shadows.

While most of the vehicle counting research are based on cctv video footages, there are also studies on car detection and counting using UAVs (Unmanned Aerial Vehicles). One such research by Ammour et al., (2017) suggests a deep convolutional neural network (CNN) system that is already pre-trained on huge auxiliary data as a feature extraction tool, combined with a linear support vector machine (SVM) classifier to classify regions into "car" and "no-car" classes. The experimental results on five UAV images show that the proposed method outperformed handcraft methods, both in terms of counting accuracy (as high as 90%) and computational time (from hours to minutes).

A previous study on implementation of transfer learning has shown that the pre-trained classifiers can be improved by re-training the model with new annotated images (weight initialization method for transfer learning). Singh et al., (2019) presented a study on usage of trained Yolo models on vehicle counting. YOLO models which is pre-trained on the MS-COCO dataset was used. The researchers did a transfer learning by re-training it on the PASCAL VOC 2007 and the KITTI datasets. This is followed by another round of training using custom annotated datasets. Five different YOLO models were generated by such fine

tuning, each model using exclusively the training data from a single camera installation. For four installations, this yields four YOLO models. The fifth model is a combination of the training data from all installations. Around 75% MAP was achieved on an 80-20 train-test split using 5562 videoframes from four different locations.

Another research was on vehicle counting using transfer learning techniques on MobileNet models by Dey et al., (2019). A MobileNet model which was pre-trained on the ImageNet dataset with size of 224×224 pixels were adopted. With a limited set of training images, the accuracy of vehicle detection was 97.4%. As for traffic volume estimates (counting accuracy) at the intersections, it was 78%. There were two important highlights in this study. Firstly, the performance was bad in case of highly overlapping of vehicles (occlusion). Secondly, the detection results at night or under very low-illumination conditions are also poor. The author suggested that the overall performance and speed of computation, could be improved by using a GPU-based multi-scale vehicle segmentation technique with an analysis of each vehicle for their direction.

III. METHODS

The high-level methodology of this research is presented by Figure 1.

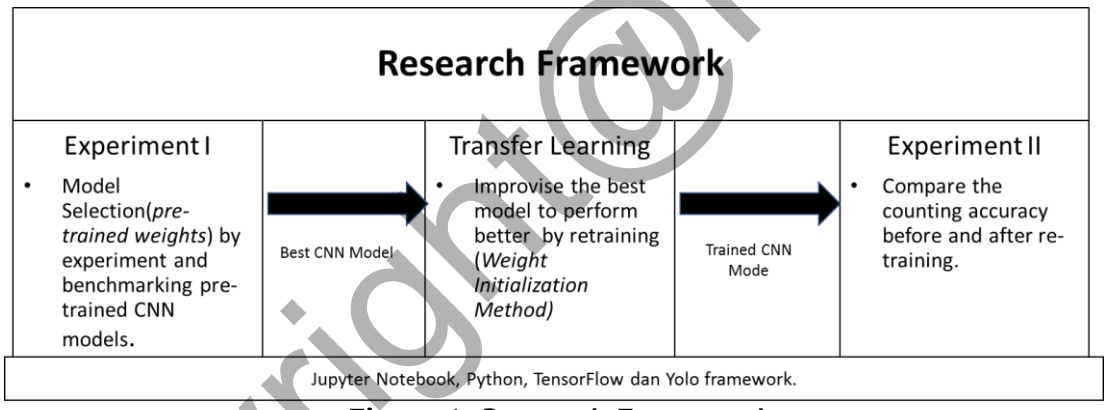

Figure 1. Research Framework

Three stages for the execution of the research. Firstly, the experiment to compare 3 CNN Models which are shortlisted and benchmarking. The best model is selected to be re-trained to improve the performance (on poor illumination) and overcome the overfitting problem. Finally experiment II to compare the performance of the re-trained model with the pre-trained model.

A. Experiment I

Three CNN models are shortlisted based on the past studies, availability of pre-trained models and ease of implementation on Tensorlow framework. The model are:

1. *SSD Inception V2*
2. *Faster RCNN ResNet101*
3. *YOLOv3 Darknet19*

*All the pre-trained models are trained on COCO dataset and available on the Tensorflow detection model zoo (2019) and TensorNets (*Tae Hoon Lee, 2018). *The overall design of this experiment is depicted by Figure 2:*

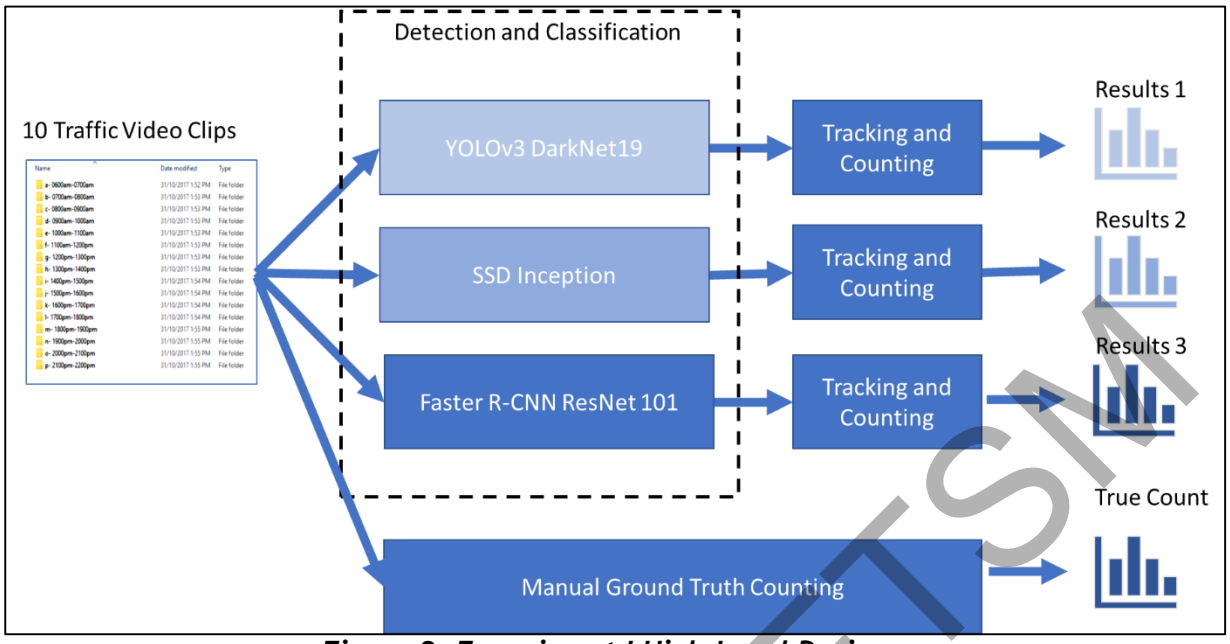

Figure 2. Experiment I High-Level Design

10 sample traffic video clips from the same location (a section of Kuala Lumpur to Kepong Highway route) for 10 different time of the day is used for this experiment. The file name and time it was recorded is listed on Table 1. Only vehicles flow on one direction was considered for the counting. A python script was developed on Jupyter Notebook to execute the experiment pipeline using Tensorflow framework.

Table 1. Details of Video Files Used in Experiment I

	Video File	Time
1.	P171003_060111_060614	6 a.m.
2.	P171003_083611_084112	8 a.m.
3.	P171003_101111_101611	10 a.m.
4.	P171003_110611_111111	11 a.m.
5.	P171003_123111_123612	12 p.m.
6.	P171003_132111_132611	1 p.m.
7.	P171003_140612_141112	2 p.m.
8.	P171003_160322_160822	4 p.m.
9.	P171003_190735_191235	7 p.m.
10.	P171003_214027_214528	9 p.m.

As for the tracking a simple method was developed to aid the counting process. In the process of detection and counting, the video clips are converted to frames and each frame are run through the detection and the output of the detection are bounding boxes with coordinates and object class. These bounding box coordinates can be used to determine the center point of each object and in this case vehicles. Assuming the first two frames of a video clip is depicted by the frames in the Figure 3.

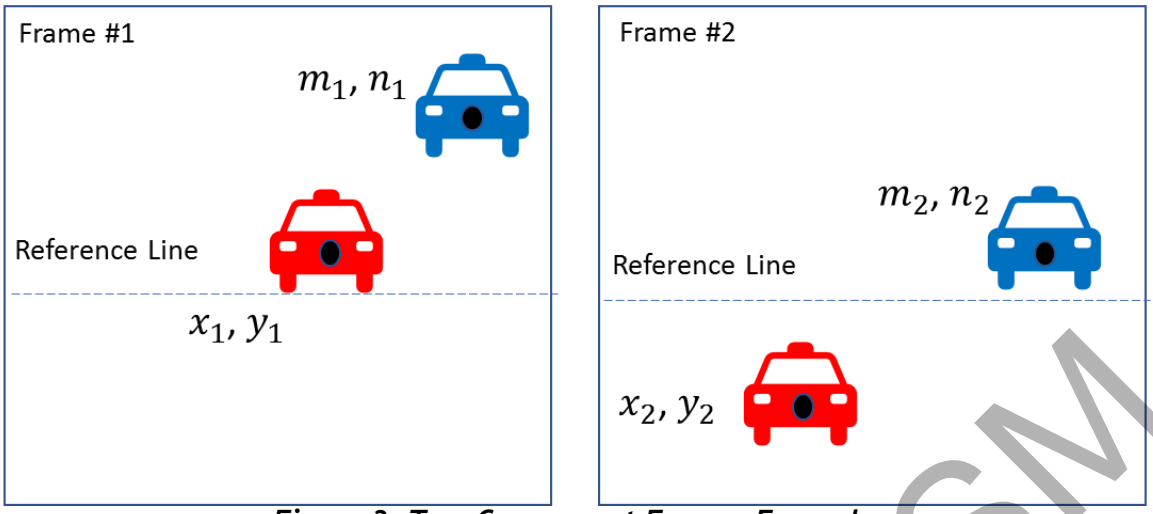

Figure 3. Two Consequent Frames Example

Assume two cars are appearing in consecutive frames and obviously in different positions given by the coordinates $(m_1, n_1), and\ (m_2, n_2)$ *for one car and coordinates* $(x_1, y_1)\ and\ (x_2, y_2)$ *for another. A reference line has been defined for all frames which will be used to determine if a car has passed or not in order to be counted in the vehicle counter.*

Let's take frame #1 first at current time t_0 *. The output from the detector gives two cars detected at coordinates as shown on the figure. So, an array is used to store these values as follows:*

$$J_m = [(m_1, n_1, car), (x_1, y_1, car)]$$

Since there were no frames previously (also considering this frame to be the first) no tracking can be done yet. So, the next frame is taken.
The output from the detector will be place in array J_m *replacing the previous values.*

$$J_m = [(m_2, n_2, car), (x_2, y_2, car)]$$

Meanwhile the values from previous frame is copied into another array J_{m-1} *(indicating previous m-1 frame):*

$$J_{m-1} = [(m_1, n_1, car), (x_1, y_1, car)]$$

Since there are values on both arrays J_{m-1} *and* J_m*, that means there are vehicles to be tracked. Euclidean distances (ED) are calculated between items in both arrays which will yield the result below:*

$$J_{dist} = [ED_{(m_1 n_1, m_2 n_2)}, ED_{(m_1 n_1, x_2 y_2)}, ED_{(x_1 y_1, m_2 n_2)}, ED_{(x_1 y_1, x_2 y_2)}]$$

The minimum distance for each point in frame #2 is determined to obtain the nearest pair from frame #1. This will result in pairs as shown in the Figure 4 and virtual line can be drawn between these pairs.

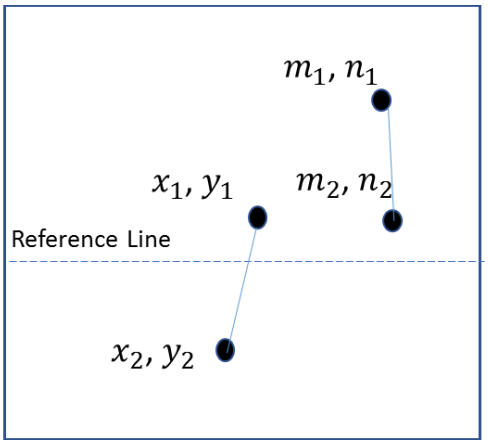

Figure 4. All Coordinates from Frame #1 and Frame #2

These virtual lines are checked if it has crossed the reference line. Assuming the reference line is at $y = y_{ref}$ then if $y_1 > y_{ref} > y_2$ then the virtual line can be said to have crossed the reference line. In such case, the vehicle will be counted as one in the counter:

$$person = 0,\ bicycle = 0,\ \boldsymbol{car = 1},\ motorcycle = 0,\ bus = 0,\ truck = 0$$

The process is repeated for next frame where J_{m-1} will be populated with points from frame #2 and J_m by points from frame #3.
This technique assumes the vehicle movement is on one direction with no occlusion.

The results of Experiment I is then averaged for each model and compared with ground truth result which was obtained by manual human counting.

B. Transfer Learning

The best model from Experiment I is YOLOv3 DarkNet19. Detailed result is presented on the next section. However, it was found that the performance of this model was worse in poor illumination or night video clips. To improve its performance, a re-training with annotated images was done. Figure 5 depicts the high-level procedures that were involved in setting up and executing the re-training.

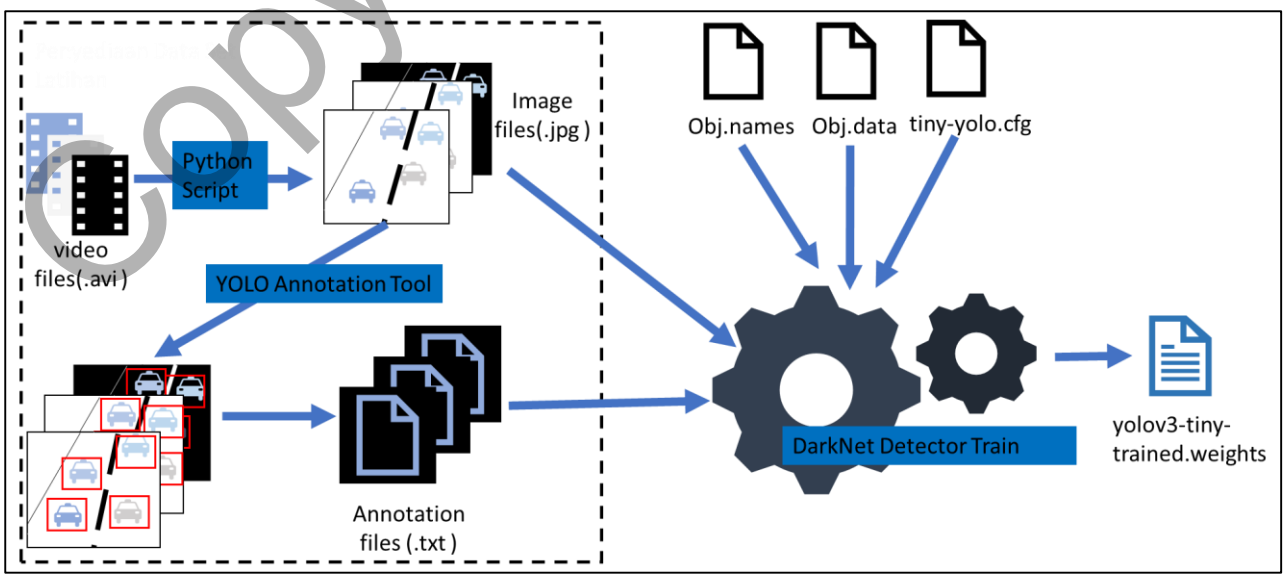

Figure 5. Re-Training Process Flow

The re-training is done using custom dataset. The process inside the dashed line box (in figure above) are the dataset preparation flow. Firstly, the video files (.avi) are converted to frames which is basically are frames (.jpg). Using the YOLO Annotation Tool (Manivanan Murugavel, 2018) *which is a Python executable program, the images of the vehicles are annotated and labelled. Bounding boxes are drawn on* images of vehicles on video frames and labelled accordingly. Figure 6 shows the GUI (Graphic User Interface) of YOLO Annotation Tool.

Figure 6. YOLO Annotation Tool GUI

The input of the Image Dir text box is the folder name containing all the frames. Pressing the Load button will load the first frame and these frames can be scrolled using "<<Prev" and ">>Next" buttons at the bottom of the GUI. For each frame, bounding boxes can be drawn on the vehicles using the cursor and labelled by selecting the option of the dropdown menu on top right of the GUI below the "Load" button. The output of the YOLO Annotation Tool is a set of Annotation files (.txt) which contains the bounding box coordinates as well as the class for each annotated vehicle in a frame. One text file is created for each image. All the images and text files need to be placed in one folder and the path given in obj.data file. In this research about 510 images from poorly illuminated video samples were used. Breakdown of total annotated vehicles are: bicycle 0, car 1866, motorcycle 457, bus 53 and truck 74.

Three additional files are required to perform the re-training. The files are obj.names which contains the classes that need to be trained, obj.data which has the pointers towards the location of the annotation files and images and finally the tiny-yolo.cfg file which is the model configuration file. The DarkNet Detector Train command then can be executed on the DarkNet framework to run the re-training process. The output of this process will be weight files for each 100th iteration. The final weight file that is produced when average loss ratio has saturated will be used for the Experiment II.

C. Experiment II

The framework of Experiment II is similar as Experiment I with only difference is on the model and the video clips that were used for the evaluation as shown on Figure 6.

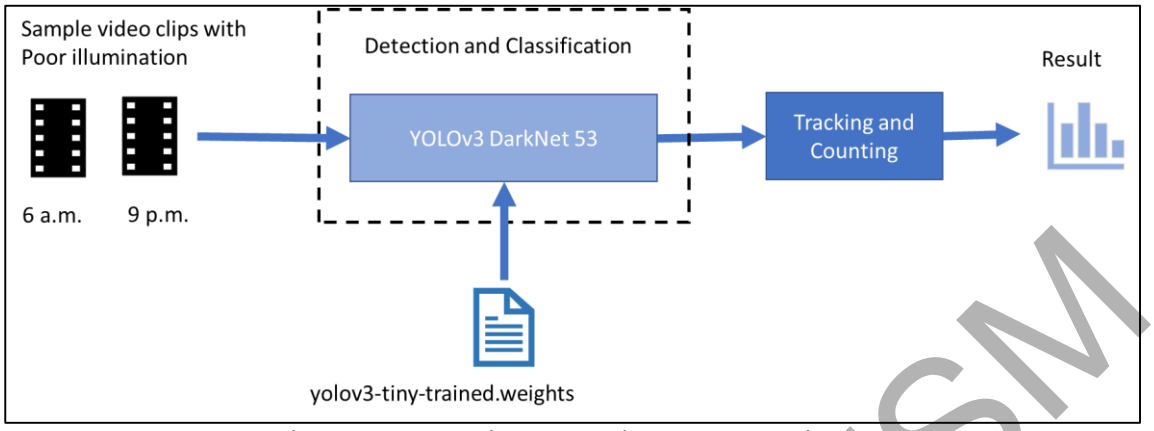

Figure 7. Experiment II High-Level Design

Two video clips were used one which is taken at early morning (6 a.m.) and another at night (9 p.m.). The results from this trained YOLOv3 model is compared with the result from Experiment I corresponding to the same video clip samples.

IV. FINDINGS AND ARGUMENT

A. Experiment I

Table 2. Experiment I Results

Model	Counting Accuracy (%)		Processing Time (sec per frame)	
	Average	Std. Dev.	Average	Std. Dev.
1. YOLOv3 DarkNet19	66.29	33.35	0.264	0.013
2. Faster RCNN ResNet101	38.12	26.26	0.532	0.037
3. SSD Inception	14.53	14.40	0.135	0.004

Experiment I resulted in YOLOv3 DarkNet19 as the architecture with the highest average counting accuracy for 10 sample videos tested as shown on Table 2. YOLOv3 DarkNet19 achieved 66.29 % on average compared to Faster RCNN ResNet101 which was second best at

38.12%. SSD Inception on the other hand recorded the fasters processing time of 0.135 seconds per frame but was the lowest in accuracy. The high standard deviation of the YOLOv3 DarkNet19 is due to huge spread of average accuracy between samples at poor illumination and day time. As shown on the line chart of Figure 7, YOLOv3 DarkNet19 achieves very high accuracy during daytime (10 a.m. to 2 p.m.) but at early morning (6 a.m.) and night (9 p.m.) the accuracy is very low which is similar to other models.

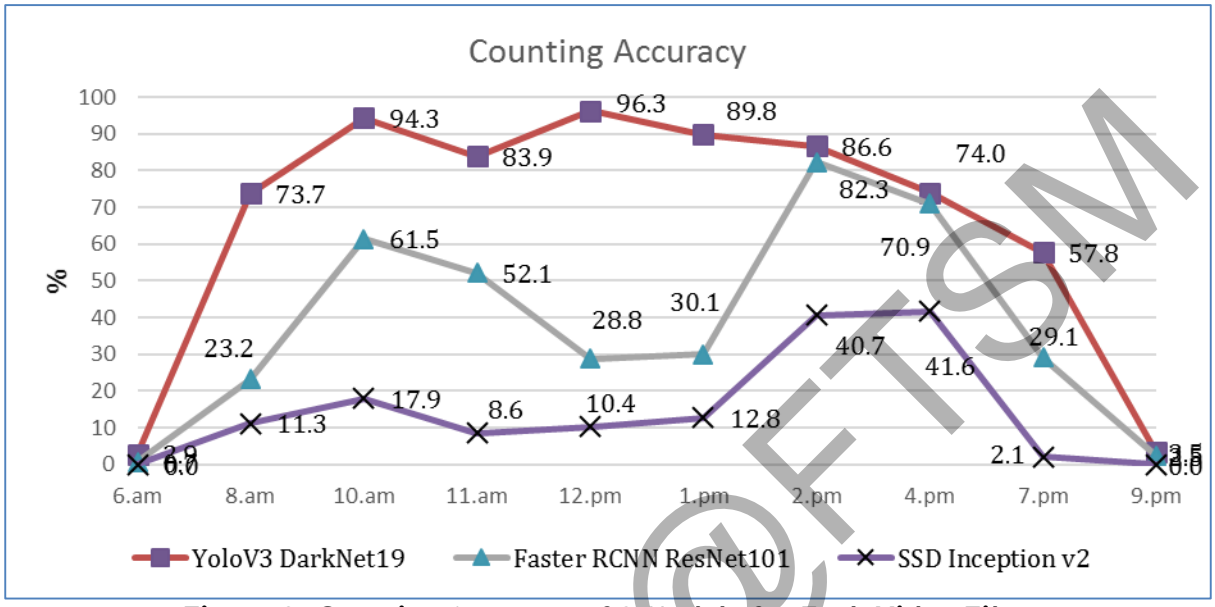

Figure 8. Counting Accuracy of 3 Models for Each Video Files

The overfitting of the pre-trained models as mentioned by Dey et al., (2019) can be seen appearing in all three models tested here. The comparison between Ground Truth (actual count of vehicles done manually) and all three models shows that in poor lighting conditions less than 10 vehicles were counted as shown on Table 3.

Table 3. Vehicle Count Per Model and Ground Truth For Each Video File

Video Time	Ground Truth	YOLOv3 DarkNet19	Faster RCNN ResNet101	SSD Inception v2
6 a.m.	140	4	1	0
8 a.m.	453	334	105	51
10 a.m.	262	247	161	47
11 a.m.	280	235	146	24
12 p.m.	299	288	86	31
1 p.m.	266	239	80	34
2 p.m.	322	279	265	131
4 p.m.	358	265	254	149
7 p.m.	237	137	69	5
9 p.m.	202	7	5	0

B. Experiment II

As explained earlier, transfer learning (weigh initialization method) was executed to re-train Yolov3 DarkNet53 model with images of vehicles of poor lighting to improve its detection and hence the counting accuracy of the overall system.The result of the

comparison shows that the counting accuracy has improved very significantly to more than 75% which is due to the model's ability to detect more vehicles in the dark environment. The line chart on Figure 8 shows this improvement.

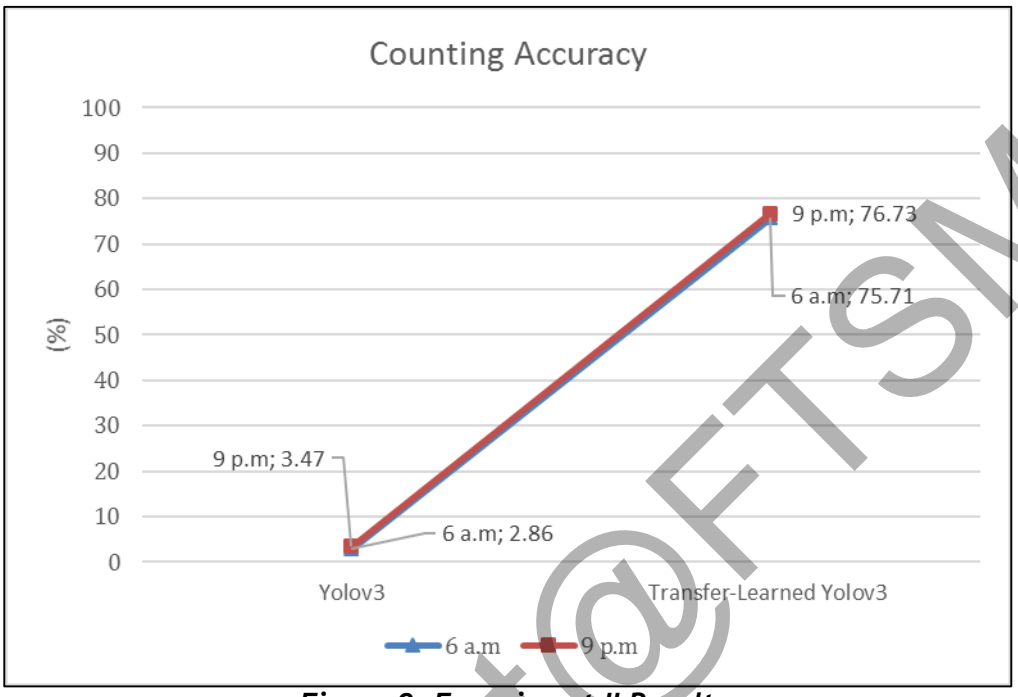

Figure 9. Experiment II Result

V.CONCLUSION

This paper has addressed the challenges in selection of best model for development of a vehicle counting system for a custom dataset. Comparison of three models (SSD Inception V2, YOLOv3 DarkNet19 and Faster RCNN ResNet101) which were pre-trained on COCO dataset showed YOLOv3 DarkNet19 achieving the best result. The results presented can be used as reference for future development of a similar counting system. However, the problem of overfitting (to pre-training COCO dataset) was obvious in video samples which were recorded in dark environment. The solution is to re-train the model with custom dataset from the dark environment using weight initialization method. The resulting model improves the counting accuracy very significantly. A tracking mechanism based on consecutive frames comparison was also proposed to aid the counting system. This mechanism may work only on vehicles moving on one direction without occlusion.

*The limitation faced in the experiments is the availability of models on the same framework (Tensorflow). TensorNets (*Tae Hoon Lee, 2018*) library was used in Experiment I to provide Yolo and DarkNet architecture. Only YOLOv3 DarkNet 19 was available in the TensorNets' repository during that time even though DarkNet53 was the latest detector. But for re-training the latest available DarkNet53 architecture was used. In future studies perhaps some uniformity can be done on the meta -architectures and detectors. Besides, the model used for re-training was a light-weighted version of YOLO which is called tiny-YOLO. This is due to limitation on the available hardware specification. To re-train YOLO the recommended minimum GPU memory is 4GB, any specification below that is only suitable for training tiny-YOLO (Manivannan Murugavel, 2018). Thus, it is recommended that future studies to consider the retraining of YOLO instead of tiny-YOLO to compare the performances.*

Acknowledgement

This work has been supported by the Malaysia's Ministry of Higher Education Fundamental Research Grant FRGS/1/2019/ICT02/UKM/02/8.

REFERENCES

Aghdam, Hamed & Heravi, Elnaz. (2017). Guide to Convolutional Neural Networks: A Practical Application to Traffic-Sign Detection and Classification, Springer 10.1007/978-3-319-57550-6.

Ammour, Nassim & Alhichri, Haikel & Bazi, Yakoub & Benjdira, Bilel & Alajlan, Naif & Zuair, Mansour. (2017). Deep Learning Approach for Car Detection in UAV Imagery, Remote Sensing. 9. 1-15. 10.3390/rs9040312.

Arcos-García, Álvaro & Alvarez-Garcia, Juan & Soria Morillo, Luis. (2018). Evaluation of Deep Neural Networks for traffic sign detection systems, Neurocomputing. 316. 10.1016/j.neucom.2018.08.009.

Arinaldi, Ahmad & Pradana, Jaka & Gurusinga, Arlan. (2018). Detection and classification of vehicles for traffic video analytics, Procedia Computer Science. 144, 259-268. 10.1016/j.procs.2018.10.527

Chuanqi Tan and Fuchun Sun and Tao Kong and Wenchang Zhang and Chao Yang and Chunfang Liu. (2018). A Survey on Deep Transfer Learning,Lecture Notes in Computer Science, 270–279, Springer International Publishing, 10.1007/978-3-030-01424-7_27

Dey, Bhaskar & Kundu, Malay. (2019). Turning Video data into Traffic data: An Application to Urban Intersection Analysis Using Transfer learning, IET Image Processing 13. 673-679. 10.1049/iet-ipr.2018.5985

Du, Juan. (2018). Understanding of Object Detection Based on CNN Family and YOLO, Journal of Physics: Conference Series. 1004. 012029. 10.1088/1742-6596/1004/1/012029

Hardjono Benny, Tjahyadi Hendra, Gracio, Mario, Widjaja Andree, Kondorura, Roberto, Halim and Andrew. (2018). Vehicle Counting Quantitative Comparison Using Background Subtraction, Viola Jones and Deep Learning Methods, IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)

Jason Brownlee. (2019). Deep Learning for Computer Vision: Image Classification, Object Detection, and Face Recognition in Python, Machine Learning Mastery

Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, Kevin Murphy. (2016). Speed/accuracy trade-offs for modern convolutional object detectors, arXiv:1611.10012

Joseph Redmon and Santosh Divvala and Ross Girshick and Ali Farhadi. (2015). You Only Look Once: Unified, Real-Time Object Detection, arXiv:1506.02640v5

Kashyap, Ramgopal, Kumar, A.V. Senthil. (2019). Challenges and Applications for Implementing Machine Learning in Computer Vision, Advances in Computer and Electrical Engineering, ISBN:9781799801825, IGI Global

Manivannan Murugavel. (23 Jun 2018). How to train YOLOv3 to detect custom objects, https://medium.com/@manivannan_data/how-to-train-yolov3-to-detect-custom-objects-ccbcafeb13d2

Manivanan Murugavel. (14 Dec 2018). YOLO Annotation Tool, https://github.com/ManivannanMurugavel/YOLO-Annotation-Tool

Pan, S.J., & Yang, Q. (2010). A Survey on Transfer Learning. IEEE Transactions on Knowledge and Data Engineering, 22, 1345-1359.

Singh Chauhan, Mayank & Singh, Arshdeep & Khemka, Mansi & Prateek, Arneish & Sen, Rijurekha. (2019). Embedded CNN based vehicle classification and counting in non-laned road traffic, arXiv:1901.06358

Tensorflow detection model zoo. (2019). https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md

Tae Hoon Lee. (2018). TensorNets, https://github.com/taehoonlee/tensornets

Yadav, Nikhil and Utkarsh Binay. (2017). Comparative Study of Object Detection Algorithms, IRJET Volume 4- Issue 11 - November 2017

Zheng, PJ & Mike, McDonad. (2012). An Investigation on the Manual Traffic Count Accuracy, Procedia - Social and Behavioral Sciences. 43. 226–231. 10.1016/j.sbspro.2012.04.095.

Zhao, Zhong-Qiu & Zheng, Peng & Xu, Shou-Tao & Wu, Xindong. (2019). Object Detection With Deep Learning: A Review, IEEE Transactions on Neural Networks and Learning Systems. PP. 1-21. 10.1109/TNNLS.2018.2876865