Vehicle Counting using Deep Learning Models: A Comparative Study

Azizi Abdullah[1], Jaison Oothariasamy[2]
Center For Artificial Intelligence Technology
Faculty of Information Science and Technology
Universiti Kebangsaan Malaysia
43600 Bandar Baru Bangi Selangor, Malaysia

***Abstract*—Recently, there has been a shift to deep learning architectures for better application in vehicle traffic control systems. One popular deep learning library used for detecting vehicle is TensorFlow. In TensorFlow, the pre-trained model is very efficient and can be transferred easily to solve other similar problems. However, due to inconsistency between the original dataset used in the pre-trained model and the target dataset for testing, this can lead to low-accuracy detection and hinder vehicle counting performance. One major obstacle in retraining deep learning architectures is that the network requires a large corpus training dataset to secure good results. Therefore, we propose to perform data annotation and transfer learning from an existing model to construct a new model for vehicle detection and counting in the real world urban traffic scenes. Then, the new model is compared with the experimental data to verify the validity of the new model. Besides, this paper reports some experimental results, comprising a set of innovative tests to identify the best detection algorithm and system performance. Furthermore, a simple vehicle tracking method is proposed to aid the vehicle counting process in challenging illumination and traffic conditions. The results showed a significant improvement of the proposed system with the average vehicle counting of 80.90%.**

***Keywords*—CNN; transfer learning; deep learning; object detection; vehicle detection**

I. Introduction

In the earlier days before the rise of machine learning, the process of vehicle counting was done manually. It was performed by a person standing by the roadside; 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 city cams or closed-circuit television (CCTV) placed above the road or highway. According to a study in [1], manual vehicle calculation performance is 99% accurate. This investigation is based on manual calculation of various vehicles from a 5 minutes video recording. Although the manual method provides high accuracy, it requires an extensive amount of human resources. Besides, it tends to be error-prone, especially on severe traffic flow and multiple road lines. 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.

Vision-based vehicle detection through highly cluttered scenes is difficult. At present, this approach can be categorized into traditional and complex deep learning methods. Recently, deep learning networks (DLN) based on convolutional neural networks (CNN) have obtained state-of-the-art performance on many machine vision task. Therefore, researchers began to use it for vehicle detection and counting. In the deep learning architecture, it learns categories gradually throughout the hidden layers. For example, in face image recognition, it starts with identifying low level features such as bright and dark areas and then proceeds to recognize lines and shapes for facial recognition. Each neuron or node represents one feature and combination of those nodes will give a full representation of the image. The 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. All these layers are learned in hierarchical order and it is very crucial to determine the high-level features of the data to make an accurate decision. The overall approach mentioned has shown high accuracy in classifying objects. Zhang et al. [2] proposed a vehicle counting system that utilized a deep learning network. The system was implemented for a static image and detect vehicles in every frame. However, there is no information stated in this paper about the flow of moving vehicles.

In the literature, many works have utilized pre-trained DLN models via transfer learning methodology for vehicle detection. In [3] used a pre-trained model via transfer learning, i.e. Yolo on vehicle counting. The model is trained using the standard MS-COCO dataset. After that the researchers re-train the model on different datasets, namely PASCAL VOC 2007, KITTI and user's custom annotated dataset. The mean accuracy precision detection is around 75% achieved on an 80-20 train-test split using 5562 video frames from four different highway locations. Another research on vehicle counting was using MobileNet [4]. A MobileNet model which was pre-trained on the ImageNet dataset with a size of 224×224 pixels for each image. With a limited set of training images, the accuracy of vehicle detection was 97.4%. As for traffic volume estimates or counting accuracy at the intersections, it was 78%. There were two crucial observations in this study. First, the performance was unsatisfactory in cases of a highly overlapping vehicles such as occlusion due to partial information. The detection performance results at night or under very low-illumination conditions are also poor. In [5] proposed to use YOLOv3 Darknet-53 for vehicle detection and counting system. The results have shown that DLN can provide higher detection and counting accuracies, especially for the detection of small vehicle objects.

Following this, some studies have been conducted to compare various available CNN models as the detector for vehicle counting systems in general, such as [6] [7] and [8] to name a few. There are also studies specifically on using deep learning

models for vehicle counting systems such as [9] [3] and [10]. Each study has varying results which highlighted the strengths and weaknesses of each pre-trained models. It seems that the model's performance is highly correlated with the local dataset and the characteristics of the vehicle movement. Therefore, there is no single CNN detector model that fits for all situation and providing the optimal detection result. In [8] presents a comparative study of CNN detector models using deep learning library of TensorFlow which provide portability and ease of use. They used the COCO dataset for evaluation.

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 model from among sets of similar pre-trained models that can perform well on intended datasets. The direct comparison to determine the optimal model is difficult due to different environment settings used in experiments. Thus, a fair comparison using a similar environment for performance evaluation is needed. One possible problem with the pre-trained models performance is that the use of standard benchmark datasets for training and completely different dataset for testing. It is common experience for the user to get poor results from the query of desired objects. Thus, instead of using the benchmark datasets, one needs to re-train the model on other large custom data for networks to learn patterns optimally [11]. But, re-training on the large data can be costly and time-consuming for deployment [12]. For example, training a deep learning algorithm on huge datasets is time-consuming and computing-intensive to secure good performance results. Therefore, one possible solution is to use a pre-trained model and transfer learning for better weight scaling and convergence speed-ups. Thus, inspired by the work of [13], a set of images with different illumination is used to re-train the existing model via transfer learning. For vehicle counting, a simple method is proposed, where the coordinate locations for each vehicle are detected in every frame. The Euclidean distance is used to computed between frames of a given video sequence for tracking and trajectory estimation. A virtual reference line is constructed, and the vehicle is counted if it crosses the line.

The contribution of this paper is as follows: (1) we compare the most widely used TensorFlow's object detection model zoo, namely, Faster R-NN, SSD and Yolov3 for vehicle counting application on urban traffic volumes. (2) we demonstrate the effectiveness of using data annotation tool for vehicle detection via transfer learning. TensorFlow's detection model zoo that trains on the standard datasets such as COCO alone is not the best to describe real-world vehicle traffic conditions, but re-training the model efficiently can enhance its ability in detecting features. (3) we propose a simple vehicle counting system that uses a virtual reference line and Euclidean distance for tracking and trajectory estimations.

The rest of the paper is organized as follows. Section 2 describes the fundamental principles of deep neural network and its application to object detection. Section 3 describes our system for vehicle counting with a focus on TensorFlow's object detection model zoo with simple tracking and counting algorithms. Experimental results on the urban traffic volumes on different conditions, i.e. morning, day and night are shown and discussed in Section 4. Section 5 concludes the paper.

II. Related Work

With recent advancements in deep learning, computer vision applications such as object classification and detection can be developed and deployed more effectively. These applications have been proposed and shown significant performance improvements and enabling real-time processing of streaming data for analytic and making decision.

A. Deep Neural Networks

Deep architectures are useful in learning and have shown impressive performance for example in the classification of digits in the MNIST dataset [14]; CIFAR [15] and ImageNet [16] for object classifications. In this scheme, the lowest layer, i.e. feature detectors are used to detect simple patterns. After that, these patterns are fed into deeper, following, layers that form more complex representations of the input data. There are several approaches to learn deep architectures. One of the most frequently used in computer vision is convolution neural networks (CNNs), where the networks preserve the spatial structure of the problem by learning internal feature representations using small squares of input data. Features are learned and used across the whole image, allowing for the objects in the images to be shifted or translated in the scene and still detectable by the network. This is one of the reasons why deep architecture is so useful for object recognition such as in picking out digits, faces, objects and so on with different challenging conditions. Thus, to get a good classification result, the network is trained with a vast number of images such as using ImageNet [16] as the dataset to classify pictures. Besides the classification task, the deep architecture is widely used for object detection that draws a bounding box around each object of interest in the image and assigns them a class label. The bounding box indicates the position and scale of every instance of each object category. There are several approaches to object detection in computer vision such as Faster R-CNN, YOLO and SSD.

Typically, deep convolutional neural network models may take days or even weeks to train on huge datasets for good performance. A way to reduce the training time is to re-use the model weights from pre-trained models that were trained using millions of natural images such as from ImageNet dataset. Such a methodology is called transfer learning. In this technique, the constructed models can be downloaded and used directly, whereby a neural network model is first trained on a problem similar to the one we have chosen. One or more layers from the pre-trained model are then used in a new model trained on the problem of interest. The pre-trained model has the advantage that it is already learned a rich set of image features. Besides, the model is transferable to the new task by fine-tuning the network. In this case, the model can be re-trained on a small number of images such that the network weights are small adjusted to support the new task. Thus, it has the benefit not only to decrease the training time for a neural network model but also can result in lower generalization error. For example, in [17] use ImageNet initialized models for object detection on the Pascal VOC dataset challenge, [18] use ImageNet initialized models for semantic segmentation. Other works that utilized the ImageNet dataset for training deep learning models for image classification such as in [19], [16].

B. Deep Learning for Object Detection

In general, object detection is a task in computer vision that involves identifying the presence, location, and type of one or more intended objects in a given test image. It is a challenging problem that consists of three main processes, namely object recognition, localization and classification. In recent years, deep learning techniques have been applied to many vehicle detection problems and show promising results such as on standard benchmark datasets and in computer vision challenges.

Several approaches are using deep learning techniques for object detection. Shaoqing Ren et al. [17] proposed a method, namely Faster R-CNN to improve both speeds of training and detection of the existing Fast R-CNN [20]. The method consists of two modules, namely, (a) region proposal network, where the convolutional neural network is used for proposing regions and the type of object to consider in the region and (b) Fast R-CNN for extracting features from the proposed regions and outputting the bounding box and class labels. Faster R-CNN has proven to be efficient for object detection and secured the first-place on both the ILSVRC-2015 and MSCOCO-2015 object recognition and detection competition tasks. Joseph Redmon et al. [21] proposed an algorithm namely, you only look once (YOLO) for object detection. The algorithm is claimed to be much faster than the standard R-CNN [22] and achieving object detection in real-time. The authors then further improved the model performance and referred to as YOLO v2 [23] and YOLO v3 [24]. Another widely used model for object detection in the industry is the single-shot multi-box detector (SSD) [25]. It improves R-CNN [22] detection speed by eliminating the need of the region proposal network.

III. Proposed Method

The proposed method consists of three main steps. The first step is to detect and draw bounding boxes around vehicles for every n-frames employing transfer learning with the deep CNN architecture. The detection algorithms were inspired by the works of [17], [25] and [21] that introduced Faster R-CNN, SSD and YOLO, respectively. Next, the trajectory of each vehicle is extracted by tracking corner points through n frames. In this step, a simple method is introduced to identify the trajectory of each vehicle found in the first step. Finally, a simple counting algorithm to count the number of vehicles on the street is proposed. Details of the algorithms are as follows:

A. Faster R-CNN

As stated before, this method was proposed by Shaoqing Ren et al. [17] which aims to improve both computational speeds and the detection accuracy of existing Fast R-CNN [20]. This technique mainly comprises of two modules which are region proposal network and Fast R-CNN for extracting features from the proposed regions. Similar to Fast R-CNN, the image is provided as an input to a convolutional network which will output a set of convolutional feature maps on the last convolutional layer. Instead of using a selective search algorithm on the feature map to identify the region proposals, a separate network is used to predict the region proposals. In this case, a sliding window of size n $\times$ n is run spatially on these feature maps. For each sliding window, a set of 9 anchors are generated which all have the same centre, but with three different aspect ratios and three different scales. Finally, the n $\times$ n spatial features extracted from those convolution feature maps are fed to a smaller network which performs classification and regression. The predicted region proposals are then reshaped using a region pooling layer which is then used to classify the image within the proposed region and predict the offset values for the bounding boxes. The regressor output determines the position, width and height of the predicted bounding box. The proposed method results outperform Fast R-CNN on a detection speed of 0.2 seconds on each image. Fig. 1 shows the Faster R-CNN model architecture. In this model, ResNet101 [26] CNN architecture is used for extracting in-depth features and classification.

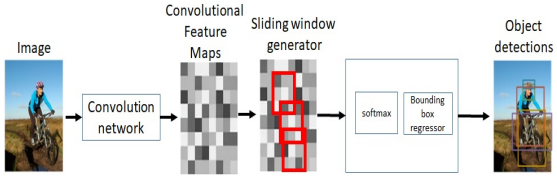

Fig. 1. Faster R-CNN Generates Anchors of Different Ratios and Scales for each Sliding Windows on Convolutional Feature Map. After that, the Output of Regressor Determines a Predicted Bounding Box.

B. Single Shot MultiBox Detector (SSD)

The SSD architecture was published in 2016 by researchers from Google for object detection in real-time [25]. It uses VGGNet convolutional neural network [19] as the base net for feature extraction. In contrast to Faster R-CNN, SSD improves the detection speed by eliminating the need of the region proposal network. In SSD, it provides a set of different default boxes with varying scales for object detection. These features (multi-scale features and default boxes) are used to recover the drop in the object detection accuracy.

Furthermore, each element of the feature map has several default boxes associated with it. The feature map sets came from different layers of the CNN network. A typical CNN network gradually shrinks the feature map size and increase the depth as it moves to the deeper layers. The deep layers cover larger receptive fields and construct more abstract representation, while the shallow layers only cover smaller receptive fields. By utilizing this information, it is possible to detect small objects in shallow layers and large objects in deeper layers. For detection, any default box with an Intersection over Union (IOU) of 0.5 or higher with a ground truth box is considered a positive sample. Fig. 2 shows the single-shot multi-box model architecture. In this model, Inception [27] CNN architecture is used for extracting in-depth features and classification.

C. You Only Look Once (YOLO)

Joseph Redmon et al. [21] proposed an algorithm namely, you only look once (YOLO) for object detection. The algorithm is claimed to be much faster than the standard R-CNN [22] and achieving object detection in real-time. In contrast to the previous schemes, YOLO uses a neural network to predict the bounding boxes and class labels for each bounding

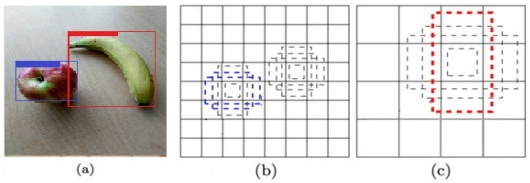

Fig. 2. (a) Original Image with Two Ground Truth Boxes, (b) Two of the 8x8 Boxes (Blue Color) are Matched with the Apple, and (c) One of the 4x4 Boxes (Red Color) is Matched with the Banana. It is Important to Note that the Boxes in the 8x8 Feature Map are Smaller than those in the 4x4 Feature Map. In Total, SSD has Six Different Feature Maps, and Each Map Responsible for a Different Scale of Objects, Enabling it to Detect Objects Cover a Large Range of Scales.

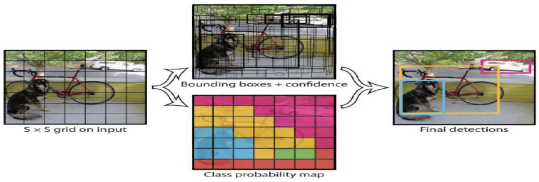

Fig. 3. Summary of Predictions made by YOLO Model. Taken from Joseph Redmon et al. in (2015) YOLO Paper.

box directly. YOLO works by taking an image and splits the input image into a grid of cells. Then, each grid cell predicts a bounding box if the centre of a bounding box falls within it. Each grid cell uses a confidence value to predict a bounding box that involves spatial coordinate x,y and the width and height of the box. For each bounding box, the network calculates a class probability value and offset values for the bounding box. The bounding boxes having the class probability map above a threshold value are then combined into a final set of bounding boxes and class labels. Fig. 3 shows the YOLO model architecture taken from [21]. In this model, Darknet [24] CNN architecture is used for exacting in-depth features and classification.

D. Vehicle Tracking

A simple vehicle tracking algorithm is proposed in this work. The process starts with converting video clips to frames. The output from the object detection model is bounding boxes with coordinates and class labels. The coordinates can be used to determine the centre point of each object and in this case, vehicles. Assuming the first two frames of a video clip is depicted in Fig. 4.

In this figure, assume that we have two vehicles at different location and frame. Here, by indicating (m_1, n_1) and (m_2, n_2) for the first vehicle coordinates in the first and second frame respectively. And (x_1, y_1) and (x_2, y_2) for the next vehicle coordinates in the first and second frame respectively. These coordinates are midpoints of the bounding boxes provided by the object detector. For example, let's say the object detector detects a vehicle in a video frame and draws a bounding box at (x_{start}, y_{start}) and (x_{end}, y_{end}), where x_{start} and y_{start} are the x and y coordinates of the upper left corner of the bounding box respectively. And x_{end} and y_{end} are the x and y coordinates of the lower right corner of the bounding box respectively. Thus, the midpoint of the coordinate

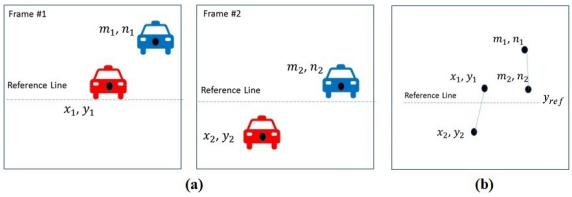

Fig. 4. (a) Sample Images from Two Consequent Frames, i.e. Frame#1 and Frame#2. (b) The Vehicle is Tracked from the Minimum Displacement from the First Frame to the Second Frame.

is $(x_{start} + (\frac{x_{end} - x_{start}}{2}), y_{start} + (\frac{y_{end} - y_{start}}{2}))$. Next, the euclidean distance is computed for each point from the first frame to the next frame resulting in four different distance values. After that the minimum displacement for each point in frame #2 is determined to obtain the nearest pair from frame #1 (Fig. 4(a)). This will result in pairs, as shown in Fig. 4 and virtual trajectory lines can be seen between these pairs (Fig. 4(b)). For counting, a reference line (dot-line) is defined in these frames, which will be used to determine if a car has passed or not to be counted in the vehicle counter algorithm.

To briefly explain the concept of the euclidean tracking algorithm suppose the number of vehicles is $\{x_i : i = 1, ..., L\}$ in the first frame and $\{y_i : i = 1, ..., K\}$ in the second frame. The goal of the euclidean tracking algorithm is to identify the nearest pair as follows:

$$TRACK(L) = \sum_{i=L} \min_{1 \leq j \leq K} ||x_i - y_j||^2$$

Thus, the tracking algorithm works by iterating from the first until all point pairs in the second frame are visited. After that, assign each observation in the first frame to the closest distance point in the second frame.

E. Vehicle Counting

The counting method is based on the vehicle regional bounding box marks and the virtual reference line. This technique assumes that the vehicle movement is in a direction. For counting, each detected vehicle in the detection step is assigned with a unique label and tracked until it reaches the virtual line. In this work, we have used five different class labels, namely bicycle, car, motorcycle, bus and truck. And all these labels are categorized as vehicle object and will be used in the counting system. After that, each vehicle position is checked whether it has crossed the horizontal reference line (y_{ref}) at the y-axis as drawn in Fig. 4(b). If it passed the line, then it will be counted as one. In this case, y_2 $coordinate\ value > y_{ref}$ $coordinate\ value$ can be said to cross the reference line.

F. Data Annotation

The need for efficient image recognition is crucial to be used in various application, such as for vehicle detection. In the literature, deep convolutional neural network models have shown remarkable achievement on many computer recognition tasks. However, these models are heavily reliant on big dataset of images taken to form a variety of conditions, such as

different orientation, location, scale, illumination, etc. Unfortunately, many existing deep learning models were trained in a limited set of image conditions, which can increase problems of overfitting and hinder generalization performance. For instance, a poorly trained deep learning network would give high vehicle detection on the daytime condition but provide a poor performance on the night time. Thus, different types of illumination conditions would affect the model's performance in detecting vehicle objects. This results to lower accuracy obtained for vehicle counting systems.

Inspired by the work of [13], the data annotation tool is used to increase the variability of training images for generalization performance detection models. This tool consists of three main steps, namely, (a) main process - used to draw the bounding box, i.e. top-left and bottom-right points on image objects for yolo training; (b) convert - to transform the bounding box points into yolo input scheme, i.e. class id, x, y, width height of the image objects, whereby x and y is the centre point of the bounding box; (c) the process - used to split the train and test dataset for yolo training. In this work, about 510 new image objects from the video samples are used. The breakdown of total annotated vehicle images are as follows: bicycle (0), car (1866), motorcycle (457), bus (53) and truck (74).

IV. Experimental Results and Analysis

Our experiments contain two stages. In the first stage, we compare the proposed object detection algorithms, namely Faster R-CNN, SSD and Yolov3 on a set of videos. Based on the results of the first stage, we further extend the experiments by applying data augmentation using a data annotation tool to improve the detection performance. All the pre-trained models are trained on the COCO dataset and available on the TensorFlow detection model zoo (2019) and TensorNets [28]. The counting process takes some time, and it depends on the number of image frames and system configuration. We have performed experiments on Intel i5-8250 CPU @ 1.60GHz with 8GB memory and GeForce MX150 GPU with 6GB memory. In this work, vehicle accuracy counting is used to evaluate the performance of each detection model. It is determined by counting vehicles with ROI passing the reference line in image frames. The vehicle counting accuracy (VCA) is computed in equation (1) as follows:

$$VCA(\%) = \frac{Number\ of\ Detected\ Vehicles}{Total\ Number\ of\ Vehicles} \times 100 \quad (1)$$

A. Dataset

In this work, 10 sample traffic video clips from the same location in two different times of the day (day and night) are used in the experiments. The video was recorded in Kuala Lumpur, Malaysia from 06 a.m to 09 p.m under clear-sky condition. Fig. 5 shows some examples of day and night images with different traffic volume and day-night illumination variations. Table I shows the video list recorded from a CCTV camera and time information used in the experiments. Only vehicles flow in one direction is considered for counting. These videos are then converted to frames and each frame becomes the input to object detection algorithm while, the output are

Fig. 5. Some Image Examples under Clear-Sky Condition in Kuala Lumpur, Malaysia. Top Half shows the Day Condition and Bottom Half shows the Night Condition.

TABLE I. Detail of Video Files used in Experiment 1. The Videos are Taken from 06 a.m to 09 p.m. The Value in Bracket shows the Ground-Truth Number of Vehicles for Each Session.

Video Files with time			
Video File	Time (a.m/p.m)	Video File	Time (a.m/p.m)
Video 1	06 a.m (140)	Video 6	01 p.m (266)
Video 2	08 a.m (453)	Video 7	02 p.m (322)
Video 3	10 a.m (262)	Video 8	04 p.m (358)
Video 4	11 a.m (280)	Video 9	07 p.m (237)
Video 5	12 p.m (299)	Video 10	09 p.m (202)

bounding boxes with coordinate and object label. After that, three detector models are used for comparison to be selected as the best vehicle detector for the counting system. These model are chosen based on the popularity in both past studies and availability of pre-trained models. Besides, they are widely used in industries for ease of implementation, especially on TensorFlow framework. These models are (a) Faster R-CNN (b) SSD and (c) YOLOv3. The first experiment (Experiment 1) investigates the best deep detector models for vehicle counting system in the day and night conditions for benchmarking. The second experiment (Experiment II) looks at improving the best method in the first experiment on selected conditions of traffic flow.

B. Experiment I

The best model in Experiment I is YOLOv3 and a detailed result is presented in the next section. Experiment I resulted in YOLOv3 as the architecture with the highest average counting accuracy for 10 sample videos tested, as shown in Table II. However, it was found that the performance of this model was worse in poor illumination, especially in the morning and night conditions. The YOLOv3 scores average vehicle counting accuracy of 66.29% compared to Faster R-CNN, which obtains the second-best average accuracy of 38.12%. On the other hand, SSD recorded the fastest processing time of 0.135 seconds per frame but has the lowest accuracy of 14.53%. The high standard deviation of the YOLOv3 and other models is due to the high variation of illumination change, especially in the morning and night conditions. As shown in Table III, YOLOv3 achieves very high accuracy during the daytime (10 a.m. to 2 p.m.) but in the early morning (6 a.m.), and night (9 p.m.) the accuracy is very low which is similar to other models. The overfitting of the pre-trained models can be seen appearing in all models tested here. Fig. 6 shows some detection results using different detection models, i.e. SSD, Faster R-CNN and Yolo V3. The accuracy vehicle counting comparison between all three models on two different time conditions (day and night) is shown in Table III.

Fig. 6. Some Detection Results in Experiment I using Different Models i.e. SSD, Faster R-CNN and YoloV3, respectively.

Fig. 7. Some Detection Results using YOLOv3 DarkNet Detector Model (a) Top Half shows Detected Vehicle before Retraining (b) Bottom Half shows Detected Vehicle after Retraining using a Data Annotation Tool.

TABLE II. AVERAGE COUNTING ACCURACY AND PROCESSING TIME FOR EACH MODEL

Model	Counting Accuracy (%)	Processing Time (frame per second)
YOLOv3 DarkNetv19	66.29±33.35	0.26 ±0.013
FasterRCNN ResNet101	38.12±26.26	0.532±0.037
SSD Inception	14.53±14.40	0.135±0.004

TABLE IV. THE AVERAGE DETECTION ACCURACY BEFORE AND AFTER DATA ANNOTATION

Detection Model	Before annotation (%)	After annotation (%)
YOLOv3 DarkNet19	66.29 ± 33.35	80.90 ± 11.62

C. Experiment II

The best performing model in experiment I is selected for further evaluation in Experiment II. The previous results have shown that the pre-trained models have some problems with a high variation of illumination in morning and night conditions. Thus, to overcome the problem, the data annotation technique is proposed. After that we compare the performance of the retrained model with the pre-trained model using the suggested data annotation tool. The retraining is done using a custom dataset taken from the video files. Firstly, the video files of type AVI are converted to a series of image frames of JPG type. Using the YOLO Annotation Tool [13], 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. To simulate the real traffic flow, two video clips were used one which is taken in the early morning (06 a.m.) and another at night (09 p.m.). In this work, about 510 images from poorly illuminated video samples were used. Breakdown of total annotated vehicles is bicycle (0 image sample), car (1866 image samples), motorcycle (457 image samples), bus (53 image samples) and truck (74 image samples). In this software, three additional files are required to perform the retraining. The files are (a) obj.names - contains the classes that need to be trained (b) obj.data - the pointers towards the location of the annotation files and images and (c) tiny-yolo.cfg - is the model configuration file. The retraining process can be executed by using the training command to YOLOv3 DarkNet framework in the YOLO data annotation tool. The output of this process will be weight files for each 100th iteration.

TABLE III. THE OVERALL ACCURACY VEHICLE COUNTING ACCURACY FOR ALL DETECTOR MODELS.

Video time	YOLOv3 DarkNet19 (%)	Faster RCNN ResNet101 (%)	SSD Inception v2 (%)
06 a.m	2.86	0.71	0.00
08 a.m	73.73	23.18	11.26
10 a.m	94.27	61.45	17.94
11 a.m	83.93	52.14	8.58
12 p.m	96.32	28.76	10.37
01 p.m	89.84	30.08	12.78
02 p.m	86.65	82.30	40.68
04 p.m	74.02	70.95	41.62
07 p.m	57.82	29.11	2.11
09 p.m	3.47	2.47	0.00

The final weight file that is produced when the average loss ratio has saturated will be used for Experiment II. Then, results from this retrained YOLOv3 model is compared with the result in Experiment I that corresponding to the same video clip samples. The experiment shows that the counting accuracy has improved very significantly with counting accuracy from 2.86% to 75.71% and 3.47% to 76.73% in the morning (06 a.m) and night (09 p.m) conditions respectively. This is due to the model's ability to detect more vehicles in poor illumination conditions. Fig. 7 shows the average counting accuracy results of YOLOv3 before and after retraining using a data annotation for vehicle counting system. The counting system improves very significantly with average accuracy from 66.29% to 80.90%. Table IV shows the average detection accuracy on the standard Yolo V3 and the proposed data augmentation on Yolo V3.

V. CONCLUSION AND FUTURE WORK

This paper addresses the challenges in the selection of the best model for the development of a vehicle counting system for a custom dataset. Comparison of three models (Faster R-CNN ResNet101, SSD Inception, YOLOv3 DarkNet) which were pre-trained on the COCO dataset showed that YOLOv3 DarkNet19 is achieving the best result. The results presented can be used as a reference for future development of a similar counting system. However, YOLOv3 DarkNet19 performs worse in the morning and night condition of the custom dataset. Thus, the solution is to retrain the model with a custom dataset from the poor illumination condition environment using a data annotation tool and employs transfer learning with the weight training 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 in one direction without occlusion. In future studies, perhaps some uniformity

can be done on the meta -architectures and detectors. Besides, the model used for retraining was a light-weighted version of YOLO, which is called tiny-YOLO. This is due to the limitation on the available hardware specification. To retrain YOLO the recommended minimum GPU memory is 4GB, any specification below that is only suitable for training tiny-YOLO. Thus, it is recommended that future studies need to consider the retraining of YOLO instead of tiny-YOLO to compare the performances.

ACKNOWLEDGMENT

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

REFERENCES

[1] P. Zheng and M. Mike, "An investigation on the manual traffic count accuracy," in *8th International Conference on Traffic and Transportation Studies (ICTTS 2012)*, 2012.

[2] Z. Zhang, K. Liu, F. Gao, X. Li, and G. Wang, "Vision-based vehicle detecting and counting for traffic flow analysis," *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 2267–2273, 2016.

[3] M. S. Chauhan, A. Singh, M. Khemka, A. Prateek, and R. Sen, "Embedded cnn based vehicle classification and counting in non-laned road traffic," in *ICTD '19*, 2019.

[4] B. Dey and M. K. Kundu, "Turning video into traffic data - an application to urban intersection analysis using transfer learning," *IET Image Processing*, vol. 13, pp. 673–679, 2019.

[5] H. Song, H. Liang, H. Li, Z. Dai, and X. Yun, "Vision-based vehicle detection and counting system using deep learning in highway scenes," *European Transport Research Review*, vol. 11, pp. 1–16, 2019.

[6] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/accuracy trade-offs for modern convolutional object detectors," 2016.

[7] Á. A. García, J. A. Álvarez, and L. M. Soria-Morillo, "Evaluation of deep neural networks for traffic sign detection systems," *Neurocomputing*, vol. 316, pp. 332–344, 2018.

[8] N. Yadav and U. Binay, "Comparative study of object detection algorithms," *IRJET*, vol. 11, 2017.

[9] A. Arinaldi, J. A. Pradana, and A. A. Gurusinga, "Detection and classification of vehicles for traffic video analytics," in *INNS Conference on Big Data*, 2018.

[10] B. Dey and M. K. Kundu, "Turning video into traffic data – an application to urban intersection analysis using transfer learning," *IET Image Processing*, vol. 13, pp. 673–679, 2019.

[11] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, "A survey on deep transfer learning," *ArXiv*, vol. abs/1808.01974, 2018.

[12] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, pp. 1345–1359, 2010.

[13] M. Murugave., "How to train yolov3 to detect custom objects," https://medium.com/@manivannan_data/how-to-train-yolov3-to-detect-custom-objects-ccbcafeb13d2, 2018.

[14] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010.

[15] A. Krizhevsky, V. Nair, and G. Hinton, "Learning multiple layers of features from tiny images," Tech. Rep., 2009. [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS 2012*, 2012.

[17] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 1137–1149, 2015.

[18] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015.

[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[20] R. B. Girshick, "Fast r-cnn," *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448, 2015.

[21] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788, 2015.

[22] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587, 2013.

[23] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6517–6525, 2016.

[24] ——, "Yolov3: An incremental improvement," *ArXiv*, vol. abs/1804.02767, 2018.

[25] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *ECCV*, 2016.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[27] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, 2015.

[28] T. H. Lee., "Tensornets," https://github.com/taehoonlee/tensornets, 2018.