Journal of the Society of Automotive Engineers Malaysia
Volume 4, Issue 1, pp 61-72, January 2020
e-ISSN 2550-2239 & ISSN 2600-8092

Deep Learning-Based Single-Shot and Real-Time Vehicle Detection and Ego-Lane Estimation

M. A. A. Abdul Matin[*1], A. S. Ahmad Fakhri[1], H. F. Mohd. Zaki[**1], Z. Zainal Abidin[1], Y. Mohd Mustafah[1], H. Abd Rahman[2], N. H. Mahamud[1], S. Hanizam[1] and N. S. Ahmad Rudin[1]

[1]Centre for Unmanned Technologies (CUTe), Kulliyyah of Engineering, International Islamic University Malaysia (IIUM), 53100 Kuala Lumpur, Malaysia
[2]Delloyd R&D (M) Sdn. Bhd., Jln. Kebun, Kampung Jawa, 41000 Klang, Selangor, Malaysia

Corresponding authors: [*]haqqnol@gmail.com.my; [**]hasanzaki@iium.edu.my

ORIGINAL ARTICLE **Open Access**

Article History:

Received
14 Sep 2019

Received in revised form
1 Dec 2019

Accepted
2 Dec 2019

Available online
1 Jan 2020

Abstract – *Vision-based Forward Collision Warning System (FCWS) is a promising assist feature in a car to alleviate road accidents and make roads safer. In practice, it is exceptionally hard to accurately and efficiently develop an algorithm for FCWS application due to the complexity of steps involved in FCWS. For FCWS application, multiple steps are involved namely vehicle detection, target vehicle verification and time-to-collision (TTC). These involve an elaborated FCWS pipeline using classical computer vision methods which limits the robustness of the overall system and limits the scalability of the algorithm. Deep neural network (DNN) has shown unprecedented performance for the task of vision-based object detection which opens the possibility to be explored as an effective perceptive tool for automotive application. In this paper, a DNN based single-shot vehicle detection and ego-lane estimation architecture is presented. This architecture allows simultaneous detection of vehicles and estimation of ego-lanes in a single-shot. SSD-MobileNetv2 architecture was used as a backbone network to achieve this. Traffic ego-lanes in this paper were defined as semantic regression points. We collected and labelled 59,068 images of ego-lane datasets and trained the feature extractor architecture MobileNetv2 to estimate where the ego-lanes are in an image. Once the feature extractor is trained for ego-lane estimation the meta-architecture single-shot detector (SSD) was then trained to detect vehicles. Our experimental results show that this method achieves real-time performance with test results of 88% total precision on the CULane dataset and 91% on our dataset for ego-lane estimation. Moreover, we achieve a 63.7% mAP for vehicle detection on our dataset. The proposed architecture shows that an elaborate pipeline of multiple steps to develop an algorithm for the FCWS application is eliminated. The proposed method achieves real-time at 60 fps performance on standard PC running on Nvidia GTX1080 proving its potential to run on an embedded device for FCWS.*

Keywords: Deep learning, Forward Collision Warning System (FCWS), ego-lane estimation, fine-tuning, feature extractor architecture, meta-architecture

Journal homepage: www.jsaem.saemalaysia.org.my

1.0 INTRODUCTION

Lack of attention by drivers is identified as the cause for 80% of driver-related accidents (Cui et al., 2010). With recent advances in technology, many applications in Advanced Driver Assistance Systems (ADAS) are implemented in cars to assist drivers to ensure safety. With the advance in low-cost camera and the surmounting pressure from the new car assessment safety rating program to push car manufacturers to continuously improve the safety ratings of their cars (Dabral et al., 2014), manufacturers look towards the potentials of low-cost cameras to achieve what high-end sensors can achieve. Therefore, many manufacturers opt for vision-based ADAS for its cost-effectiveness.

For object detection, most existing systems still use traditional computer vision approaches such as Cascaded Haar-like feature detection (Viola & Jones, 2001), HOG-SVM (Dalal & Triggs, 2005) and Hough transform-based line detection (Ballard, 1981). However, such models are susceptible to environmental noises such as adverse lighting conditions, viewpoint changes, and partial occlusion. This is due to the fact these traditional methods depend on lower level features that are based mostly but not strictly on edges, corners, symmetries and histograms gradients.

Deep neural networks (DNN) based methods have achieved unprecedented performance in solving several computer vision problems involving image classification and object detection which could become a key enabler for highly accurate and robust ADAS applications. The key benefit of DNN, as opposed to traditional methods for classification and detection, is because they can extract appropriate features for the tasks. Following the breakthrough of AlexNet (Krizhevsky et al., 2017) architecture that led to the popularity that shows DNN performs better than their traditional approach; the increasing number of researches focused on using DNN for object detection tasks. This led to numerous successful DNN object detection architecture with high performances.

DNN based object detection architectures such as R-CNN (Girshick et al., 2014), SSD (Liu et al., 2016), and YOLO (Redmon et al., 2016) are an extension of classification based DNN. The end goal is concerned with giving object localization that involves drawing a bounding box around one or multiple objects of multiple classes. The extension architecture for object detection is called meta-architecture. Meta-architecture is usually built on top of the pre-trained classification model, which is sometimes called feature extractor architecture. This brings an end-to-end solution for object detection.

However, developing Forward Collision Warning System (FCWS) requires a vehicle detection model and ego-lane estimation model which are two separate tasks that are not practical for real applications in embedded advanced driver assistance systems (ADAS). The core bottleneck in such methods is that deep neural network (DNN) was trained for each task and combined in a late fusion manner. Moreover, although DNN constitutes the highest performing models for vehicle detection and ego-lane estimation, it is also known for data hungriness and computational complexity. Therefore, we propose a unified DNN based single-shot vehicle detection and ego-lane estimation architecture which allows both tasks to be performed simultaneously in a single shot.

Breakthrough in DNN brought opportunities to bring detection models to be implemented in vision-based ADAS application for secondary safety or crash protection technologies to deliver large life savings. Among the potential benefits of high accuracy object

detections in ADAS are collision avoidance systems such as FCWS, reverse collision warning system, adaptive cruise control and emergency brake assist. Although many of these systems have been developed using high-end sensors, a breakthrough on vision-based detection using DNN with high accuracy detection has opened up opportunities for cheaper replacement possibilities without compromising performance.

2.0 METHODOLOGY

2.1 Related Works

2.1.1 Ego-lane detection

The lane that the host-vehicle is driving on is called the ego-lane. Identifying the ego-lane allows the FCW system to understand where the other vehicles detected on a frame is positioned relative to the host-car. This is important because the vehicle that is directly in front of the host-car is the vehicle that is most likely to be in a course to a collision. Moreover, understanding the characteristic of the ego-lane allows the estimation of the trajectory of the host-car is headed to and therefore a prediction model can be performed as to which target vehicle the host-car is likely to collide in consecutive time steps.

To define the host car's ego-lane, literature has shown there is a lot of methods to achieve that. Using the classical method of image processing, it is possible to segment out the lane lines using edge features and colour thresholding method (Li et al., 2018). Lane-lines are then created by grouping them based on their regions (Chen et al., 2018). The line fitting method can be used to group regions and used to generate and define a lane line.

Recent works have proposed deep learning-based methods to solve lane line segmentation. Lane-line detection is made possible using a semantic segmentation architecture which works by using a base Convolutional Neural Network (CNN) feature extractor (also known as an encoder) and then connecting the encoder to an up-sampling neural network (also known as a decoder) to output the segmentation mask (Lee et al., 2017).

Aside from using DNN based model to output a segmentation mask, a CNN feature extractor is also known to be able to extract micro features in an image. Micro-feature such as the features on a human face can be extracted to perform human age and emotion predictions (Levi & Hassncer, 2015). John et al. (2015) showed that it is possible to use CNN to extract lane features and train a decision tree model to make predictions to where the current lane is.

2.1.2 Vehicle detection

Sivaraman & Trivedi (2013) summarized some early works in traditional vision-based vehicle detection. In general, it is possible to achieve vehicle detection by both the monocular vision method and stereo vision method. This paper will focus on the study of monocular vision methods. Studies for vehicle detection using monocular vision has shown that this was generally achieved by either evaluating the characteristics of the frames' appearances (Niknejad et al., 2012; Sun et al., 2006; Lin et al., 2012) or the frames' motions (Alonso et al., 2008; Jazayeri et al., 2011).

To achieve detections by appearances means to use general image features like edges, corners, and symmetry that best represent the vehicle. Some study also uses robust feature sets to allow classification and detection of vehicles. Robust feature sets are extracted and are trained with a machine learning model that analyses these features used for classification (Joh et al., 2016).

2.1.3 Deep Neural Network (DNN)

Using the traditional method for vision-based detection images goes through two phases. The first phase is the feature extraction method. During this phase, hand-engineered algorithms are applied such as HOG and Harr-like features. to quantify the contents of an image based on a component of the image we want to encode (i.e., shape, color, texture). At the second phase, given these features extracted, these are then used to train our classifier and evaluate it. However, when building the Convolutional Neural Network, we can skip handcrafting a feature extracting step. The reason for this is because CNN models are end-to-end trainable models.

We construct the convolutional layers and then present the raw input data (pixels) to the network. A learning algorithm will then teach these networks through backpropagation to learn suitable filters inside its layers that can be used to discriminate and classify object classes. This yield learned representations of data with multiple layers of abstraction. The output of the network is then a probability distribution over class labels (Krizhevsky et al., 2017). One of the notable aspects of using CNNs is that instead of carefully designing highly tuned hand-crafted features, the network learns the discriminative features by modelling the data distribution of the input in an end-to-end manner.

2.2 System Design

In this sub-section, the overall architecture and training of the architecture of the single-shot vehicle detection and ego-lane estimation architecture will be discussed. Furthermore, dataset collection and processing to be used for the training, validation, and testing will be discussed as well.

2.2.1 Data collection and annotation

For the ego-lane estimation architecture training dataset, we first define individual ego-lane as a regression problem. Concretely, given an image as an input matrix, the DNN needs to predict the *x*-coordinate for a given *y*-coordinate. As depicted in Figure 1, we define a fixed step along the *y*-coordinate and the last layer of DNN will be trained to estimate a corresponding *x*-coordinate.

To achieve a DNN architecture that outputs both vehicle detections and ego-lane estimations in a single shot, suitable datasets need to be prepared for training. For those reasons we can resort to the following options to collect datasets to achieve our desired outcome:

i. Open-source dataset: CULane Dataset (Sandler et al., 2018)
ii. Hand-labelled dataset: Malaysian own dataset

For vehicle detection, we hand-labelled images collected from a dashcam. We use the augmentation approach to increase the number of data and add variations in the dataset. Types of augmentation include flipping, raining effect, glaring warping, and random box occlusion.

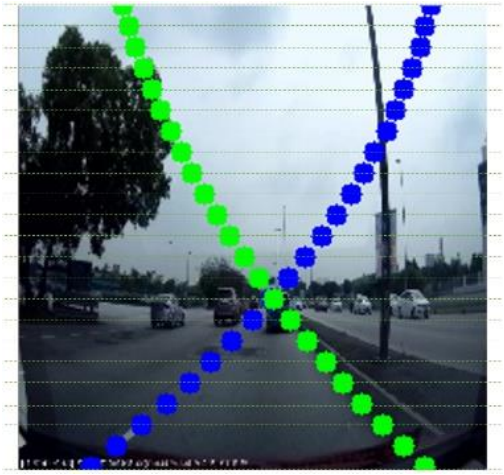

Figure 1: Ego-lane defined as a point regression problem

2.2.2 Dataset statistics

In this section, we provide the statistics of our proposed dataset for single-shot detection and ego-lane estimation tasks (Table 1 and 2). In summary, we manually collected and labelled ~4k frames for the ego-lane dataset and a total of ~1.6k vehicle detection dataset.

Table 1: Ego-lane dataset stats

Dataset	Ego-lane Dataset Stats		
	Type	Training Set (Number of Frames)	Test Set (Number of Frames)
CULane	Original training dataset	21,819	8,934
CULane	Augmented original training set	128,329	14,258
Malaysian Dataset	Original hand-labelled set	4,000	800
Malaysian Dataset	Augmented of original	59,068	14,767

Table 2: Vehicle detection dataset stats

Dataset	Vehicle Detection Dataset Stats		
	Type	Training Set	Test Set
Malaysian Dataset	Dashcam recording - original	1,607	402
Malaysian Dataset	Dashcam recording - augmented	2,894	724

2.3 Neural Network

2.3.1 Architecture

To achieve constructing single-shot vehicle detection and ego-lane estimation architecture we must choose a meta-architecture and a corresponding feature extractor architecture (Figure 2). In this paper, we use a pre-trained MobileNetv2 (Sandler et al., 2018) as our backbone model for meta-architecture due to its trade-off performance between accuracy and efficiency. Development of the single-shot vehicle detection and ego-lane estimation architecture will proceed with the following steps sequentially:

1. We train a feature extractor architecture to predict where the ego-lanes are on an image. This is a supervised learning approach that can be achieved by training a pre-trained DNN feature extractor architecture based on thousands of lane labelled datasets.
2. We then train a detection meta-architecture utilizing the already ego-lane trained feature extractor architecture to train for vehicle car detection. This again is a supervised learning approach based on thousands of vehicles labelled datasets.

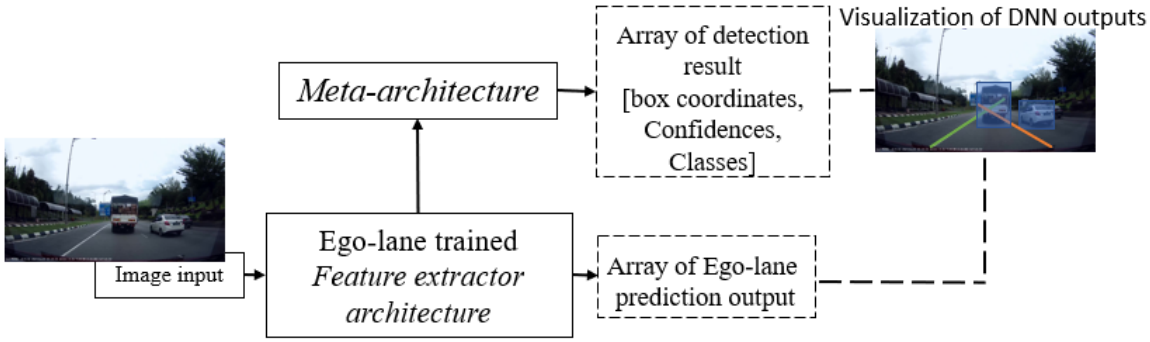

Figure 2: Overall single-shot vehicle detection and ego-lane estimation pipeline

2.3.2 Ego-lane estimation architecture

Feature extractor architectures as discussed are the DNN models that are trained with thousands of images to classify multiple objects, usually hundreds of objects. These feature extractor models can be used as a standalone architecture to classify objects in an image. Since these architectures had been trained on thousands of datasets to classify hundreds of classes, they are known to have learned several discriminative features critical to identifying the trained object.

Here we treat ego-lane estimation as a point regression problem. The aim here is that given fixed *y*-coordinate values in an image the network will be trained to estimate the *x*-coordinate values of where the lane is. The overall training of point regression architecture is depicted in Figure 3.

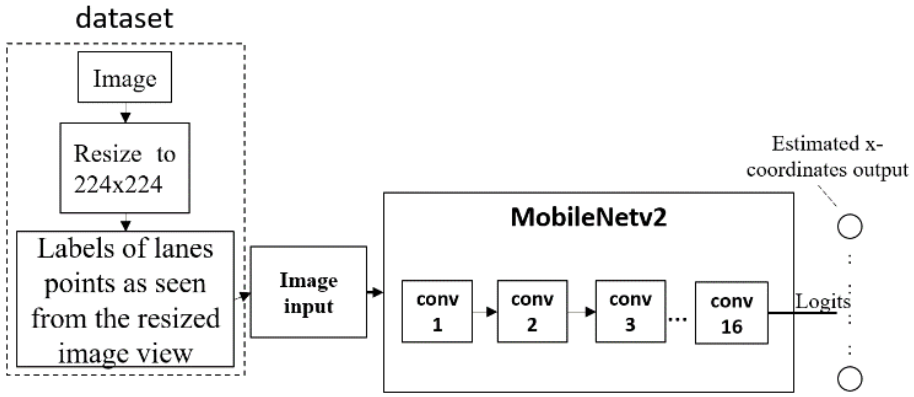

Figure 3: Training of point regression architecture will require dataset labelled based on the resized image

To achieve ego-lane estimation as point regression problem we pre-process ego-lane dataset such that the labels are the *x*-coordinates of a given fixed *y*-coordinate. Since MobileNetv2 by design is a 224x224 input architecture we first pre-process dataset to a size of 224x224. For fixed *y*-coordinates, we choose a *y*-coordinate at every tenth-pixel step and then label where the *x*-coordinate at that fixed *y*-coordinate lies on. Referring to Figure 4, dotted horizontal lines show the tenth pixels steps from along the *y*-axis. At each step, the output is the x-axis coordinate value that the ego-lane DNN model must predict. Therefore, the dataset for this purpose should be labelled as such that the dataset should first be resized to 224x224 and then the labels are the *x*-axis of every step along the *y*-axis.

2.3.3 Vehicle detection

We chose the Single Shot Detector (SSD) meta-architecture for objection. We utilize the lightweight architecture SSD-Mobilenetv2 for the development of DNN architecture for which SSD meta-architecture will be trained on top of the already trained ego-lane feature extractor architecture for vehicle detections.

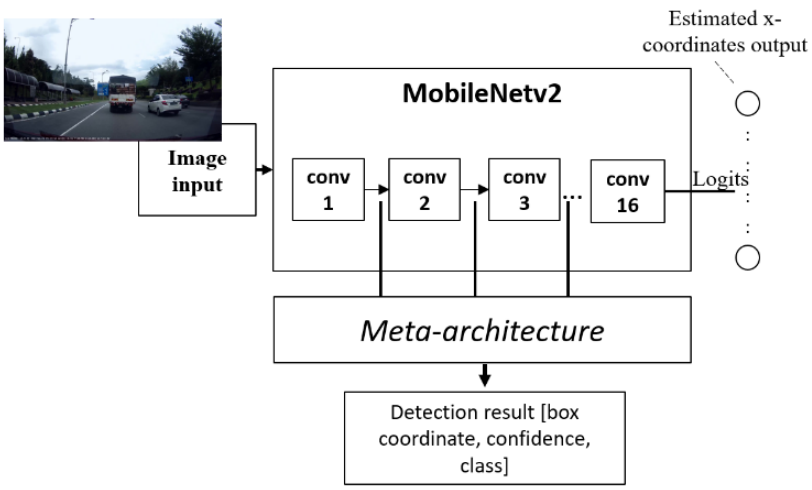

Figure 4: Overall single-shot vehicle detection and ego-lane estimation using MobileNetv2

3.0 RESULTS

3.1 Experimental setup

The implementation and training of our proposed algorithm were done on an Alienware R6 whereas speed performances were tested on multiple platforms, namely Alienware R6, HP ENVY 13-AH0042TX and Lenovo Y510P where the specifications are shown in Table 3.

Table 3: Devices specifications to be used to run real-time test

No.	Platform	Hardware Used
1	Alienware R6 • i7 7th gen Intel CPU • Nvidia GTX 1080 8GB • RAM 16GB	GPU
2	HP ENVY 13-AH0042TX • I7 8th gen Intel CPU • Nvidia GeForce MX150 2GB • RAM 8GB	GPU
3	Lenovo Y510P • Intel i7 4th gen • RAM 8GB	CPU

3.2 Evaluation Metrics

For ego-lane estimations, the output estimation of the ego-lane estimation model will be evaluated against the ground truth. A script was written to compute the estimation performance by calculating the average precisions of the ego-lane estimation as compared with the ground truth.

$$\text{Total Average precision} = \sum_{k=0}^{n} \frac{\frac{TPRk}{TPRk+FPRk} + \frac{TPLk}{TPLk+FPLk}}{2} \quad (1)$$

For vehicle detections, output detection of the vehicle detection model will be evaluated against the ground truth dataset. An evaluation script was used to compute the estimation performance by calculating the mAP of the detections.

3.3 Ego-lane Estimation Results

In order to truly evaluate the performance of DNN models, evaluating the models' result against the test dataset are necessary. This is because the evaluation done on a test dataset is done on a dataset that the model has not seen during training. This will enable us to understand how well the training of the model performs to generalize its intended tasks. For ego-lane estimation, we evaluate based on the CULane test dataset and compare results based on the results presented by (Pan et al., 2018). Moreover, we evaluate the ego-lane estimation model trained on the Malaysian dataset against our Malaysian test dataset (Table 4).

Table 4: Total accuracy comparison based on CULane test dataset

Architecture	Total Accuracy
Baseline+SCNN	90.6%
ResNet-101	90.2%
Mobilenetv2 (CULane dataset) – point regression (ours)	**88.39%**
ResNet-50	87.4%
MRFNet	86.3%
DenseCRF	81.3%
ReNet	83.3%

3.4 Vehicle Detection Results

The vehicle detection DNN model will be evaluated against the test dataset prepared from our Malaysian test dataset. Firstly, we evaluated the SSD+MobileNetV2 model trained on the raw Malaysian vehicle detection dataset. Secondly, we evaluated the SSD+MobileNetV2 model trained on the augmented Malaysian vehicle detection dataset. We will discuss the effects of data augmentation have on the performance of the DNN model. Finally, we train a better variant of the SSD based model known as the SSD+InceptionV2 model for object detection with our vehicle detection dataset. This will present to us how far off our SSD+MobileNetV2 vehicle detection model is from the better model.

Table 5 summarizes the results of training DNN architecture for vehicle detection. It is noticed varying outcome happens when dataset increased by augmentation. Augmentation of dataset adds more dataset with for training with little to no labelling cost. Therefore, augmentation is always a preferred pre-processing method. SSD+Inceptionv2, on the other hand, was evaluated which gives a ~10% more accuracy which is expected since the feature extractor architecture allows more features to be captured and to further utilized for the meta-architecture. Given that SSD+MobileNetv2 is still very lightweight in comparison to SSD+InceptionV2.

Table 5: Performance comparison of vehicle detection model

Architecture	Dataset	Car AP@0.5 IOU	Bike AP@0.5 IOU	Truck AP@0.5 IOU	mAP@0.5 IOU
SSD+MobileNetv2	Vehicle detection dataset without augmentation	0.79885	0.36566	0.543969	0.569495
SSD+MobileNetv2	Vehicle detection dataset with augmentation	0.84352	0.35022	0.719649	0.637800
SSD+Inceptionv2	Vehicle detection dataset with augmentation	0.86426	0.57328	0.810133	0.749228

3.5 Speed Performance

Finally, we test the real-time performance of the single-shot vehicle detection and ego-lane estimation architecture on CPU and GPU computer devices. The results of the processing speed are as in Table 6.

Table 6: Processing speed of our model SSD+MobileNetv2 and SSD+Inceptionv2 on various platforms

Model	SSD+MobileNetV2 FPS	SSD+InceptionV2 FPS
Alienware R6	60	48
HP ENVY 13-AH0042TX	25	20
Lenovo Y510P	18	10

4.0 CONCLUSION AND FUTURE WORKS

This research investigates the potential use of deep learning architecture that can simultaneously do vehicle detection and ego-lane estimation in a single-shot. Pre-trained feature extractor architecture trained on thousands of images to classify several categories can be fine-tuned to make predictions of where traffic lanes are located on an image. A meta-architecture to do object detection can then be trained based on the feature extractor architecture trained on the ego-lane dataset. These methods had been proved to work for a single-shot vehicle detection and ego-lane estimation architecture application by utilizing SSD+MobileNetv2 architecture.

To sum up, it was found that the DNN model was able to be trained to estimate the location of ego-lanes from a given image frame. To achieve this, we trained feature extractor architecture MobileNetv2 to estimate ego-lane location by treating ego-lanes as a regression problem. We've seen that our architecture and method of training the DNN architecture had allowed us to achieve a single-shot output for both vehicle detection and ego-lane estimation. This was achieved by first training the ego-lane architecture to successfully output a suitable ego-lane estimation. Once that is achieved a vehicle detection meta-architecture is trained based on the frozen weightage of the ego-lane trained feature extractor architecture. Consequently, that results in allowing us to get simultaneous vehicle detection and ego-lane estimation outputs.

Finally, results had shown that this method achieved real-time performance without compromising the speed of the original SSD+MobileNetv2 making this architecture more valuable for FCWS. This allows you to have two advantages for a single deep learning architecture. While ego-lane estimation here still does not beat a more robust semantic segmentation architecture as in the literature, this method serves as satisfactory use for FCWS and can be extended for lane departure warning system. Some limitations and recommendation of this project:

1. Use of the single-shot vehicle detection and ego-lane estimation architecture was limited on a personal computer. Personal computer by default is equipped with high specification hardware which is contrary to the reality of deploying it on an actual car. Deploying it on an actual car makes sense if it is on an embedded hardware with constrained resources. Automotive standards often have the highest standard to follow so challenges are still plenty to realize deploying this project's deep learning algorithm to an automotive-grade embedded system.

2. Ego-lane estimation tested here are devoid of an influence created from any feedback estimation. So often time it is seen that the ego-lane estimation displays on the output screen to be jerky especially when scenes of the traffic lane changes. This can be further improved by integrating the tracking mechanism or simply use the Kalman filter to improve.

3. Ego-lane estimation as observed is very good when inferencing an image taken from the same camera the dataset trained with was taken from. Although we have done a lot of dataset augmentation further investigation can be made to further improve the model to be able to learn variance in camera specification.

ACKNOWLEDGMENTS

This research was sponsored by CREST and Delloyd R&D Sdn. Bhd., under grant CREST ID: P11C2-17 (SMART DRIVER ASSISTANCE SYSTEM). We thank NVIDIA Corporation for donating the Titan XP used for this research.

REFERENCES

Alonso, J.D., Vidal, E.R., Rotter, A., & Muhlenberg, M. (2008). Lane change decision aid system based on motion-driven vehicle tracking. *IEEE Transactions on Vehicular Technology*, *57*(5), 2736-2746, sept. 2008.

Ballard, D. (1981). Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, *13*(2), 111-122. doi:10.1016/0031-3203(81)90009-1

Chen, P., Lo, S., Hang, H., Chan, S., & Lin, J. (2018). *Efficient road lane marking detection with deep learning*. Retrieved from https://arxiv.org/abs/1809.03994

Chinese University of Hong Kong (2018). *CULane Dataset by Multimedia Laboratory*. Retrieved from https://xingangpan.github.io/projects/CULane.html

Cui, J., Liu, F., Li, Z., & Jia, Z. (2010). *Vehicle localisation using a single camera*. Paper presented at 2010 IEEE Intelligent Vehicles Symposium, California, USA.

Dabral, S., Kamath, S., Appia, V., Mody, M., Zhang, B., & Batur, U. (2014). *Trends in camera based Automotive Driver Assistance Systems (ADAS)*. Paper presented at 2014 IEEE 57th International Midwest Symposium on Circuits and Systems (MWSCAS), Texas, USA. doi:10.1109/mwscas.2014.6908613

Dalal, N., & Triggs, B. (2005). *Histograms of oriented gradients for human detection*. Paper presented at 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR05), San Diego, CA, USA. doi:10.1109/cvpr.2005.177

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). *Rich feature hierarchies for accurate object detection and semantic segmentation*. Paper presented at 2014 IEEE Conference on Computer Vision and Pattern Recognition, Washington, DC, USA. doi:10.1109/cvpr.2014.81

Jazayeri, A., Cai, H., Zheng, J.Y., & Tuceryan, M. (2011). Vehicle detection and tracking in car video based on motion model. *IEEE Transactions on Intelligent Transportation Systems*, *12*(2), 583-595.

John, V., Liu, Z., Guo, C., Mita, S., & Kidono, K. (2015). Real-time lane estimation using deep features and extra trees regression. *Image and Video Technology*, *9431*, 721-733.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84-90. doi:10.1145/3065386

Lee, S., Kim, J., Yoon, J.S., Shin, S., Bailo, O., Kim, N., Lee, T.-H., Hong, H.S., Han, S.-H., & Kweon, I.S. (2017). *VPGNet: Vanishing Point Guided Network for lane and road marking detection and recognition*. Paper presented at 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy. doi:10.1109/iccv.2017.215

Levi, G., & Hassncer, T. (2015). *Age and gender classification using convolutional neural networks*. Paper presented at 2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Boston, MA, USA. doi:10.1109/cvprw.2015.7301352Y.

Li, M., Li, Y., & Jiang, M. (2018). Lane detection based on connection of various feature extraction methods. *Advances in Multimedia, 2018*, 1-13. doi:10.1155/2018/8320207

Lin, B.-F., Chan, Y.-M., Fu, L.-C., Hsiao, P.-Y., Chuang, L.-A., Huang, S.-S., & Lo, M.-F. (2012). Integrating appearance and edge features for sedan vehicle detection in the blind-spot area. *IEEE Transactions on Intelligent Transportation Systems*, *13*(2), 737-747.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. *Computer Vision – ECCV 2016 Lecture Notes in Computer Science*, 21-37. doi:10.1007/978-3-319-46448-0_2

Niknejad, H.T., Takeuchi, A., Mita, S., & McAllester, D. (2012). On-road multivehicle tracking using deformable object model and particle filter with improved likelihood estimation. *IEEE Transactions on Intelligent Transportation Systems*, *13*(2), 748-758.

Pan, X., Shi, J., Luo, P., Wang, X. and Tang, X. (2018). Spatial as Deep: Spatial CNN for Traffic Scene Understanding. Paper presented at The Thirty-Second AAAI Conference on Artificial Intteligence, New Orleans, LA, USA.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779-788. doi:10.1109/cvpr.2016.91

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4510-4520. doi:10.1109/cvpr.2018.00474

Sivaraman, S., & Trivedi, M. M. (2013). A review of recent developments in vision-based vehicle detection. *Proceedings of the 2013 IEEE Intelligent Vehicles Symposium (IV)*, 310-315.

Sun, Z., Bebis, G., & Miller, R. (2006). On-road vehicle detection: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(5), 694-711.

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001)*, 1-9. doi:10.1109/cvpr.2001.990517