Real-Time Single-Stage Vehicle Detector Optimized by Multi-Stage Image-Based Online Hard Example Mining

Che-Tsung Lin, Shu-Ping Chen, Patrisia Sherryl Santoso, Hung-Jin Lin, and Shang-Hong Lai, *Member, IEEE*

***Abstract*—Vehicle detection is a fundamental function required for advanced driver assistance systems. Extensive research has shown that good performance can be obtained on public datasets by various state-of-the-art approaches, especially the deep learning methods. However, those methods are mostly two-stage approaches which inevitably require extensive computing resources and are hard to be deployed on an embedded computing platform with real-time computing performance. We introduce a single-stage vehicle detector which can work in real-time on NVIDIA DrivePX2 platform. The main contributions of this paper are threefold. We propose a detection scheme which includes multi-scale features and multi-anchor boxes to improve the accuracy of a single-stage detector. Secondly, a new data augmentation strategy is proposed to systematically generate a lot of vehicle training images whose appearances are randomly truncated, so our detector is trained to detect partially-seen vehicles better. Thirdly, we present a multi-stage image-based online hard example mining (MSI-OHEM) framework specifically designed for single-stage detectors. MSI-OHEM performs fine-tuning on hard examples and the ones with slightly-insufficient IOU that are considered true positives. Compared to other classical object detectors, the proposed detector achieves very competitive result in terms of average precision (AP) and computational speed. For the newly-defined vehicle class (car+bus) on VOC2007 test, our detector, using MobileNetV2, GoogLeNet, Inception-v2 and ResNet-50 as basenets, achieves 85.35%/85.62%/86.49%/87.81% AP and runs at 64/58/48/28 FPS on NVIDIA DrivePX2, respectively.**

***Index Terms*—Deep Learning, vehicle detection, convolutional neural networks (CNN), bootstrapping, hard negative mining.**

Manuscript received October 4, 2018; revised February 2, 2019, July 7, 2019, and October 19, 2019; accepted December 10, 2019. Date of publication December 23, 2019; date of current version February 12, 2020. This research is partially supported by the Ministry of Science and Technology (MOST), Taiwan, R.O.C., under the Grant MOST 108-2634-F-007-002. The review of this article was coordinated by Dr. A. Chatterjee. *(Corresponding author: Shang-Hong Lai.)*

C.-T. Lin is with the Mechanical and Mechatronics System Research lab, Industrial Technology Research Institute, Hsinchu, Taiwan, and also with the Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan (e-mail: alexofntu@gmail.com).

P. S. Santoso is with Mechanical and Mechatronics System Research lab, Industrial Technology Research Institute, Hsinchu, Taiwan (e-mail: sherrylsantoso@itri.org.tw).

S.-P. Chen and H.-J. Lin are with the Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan (e-mail: scarletclaw24@gmail.com; vtsh.jn@gmail.com).

S.-H. Lai is with the Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan (e-mail: lai@cs.nthu.edu.tw).

This article has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors.

Digital Object Identifier 10.1109/TVT.2019.2961625

I. Introduction

INTELLIGENT Transportation Systems (ITS) aim at improving safety of ground transportation. Vehicles have been evolved such that that they are equipped with better perception for the road condition and better understanding of the driving environment. The on-road detection of vehicles has been a topic of great interest to researchers over the past decade [1]. One expects that on-road vehicles could be detected as accurately as possible by an advanced driver assistance system (ADAS) because the function of such a system is to enhance driving safety especially for the scenarios that the host vehicles and the preceding vehicles are very close.

Millimeter-wave radar [2] and Lidar [3] could directly estimate depth of the environment to assist a host vehicle for avoiding potential collision. However, the vision-based approach could detect objects and differentiate vehicles from non-vehicle objects. Stereo vision could estimate distances to objects with sufficient texture information and the distance information is helpful to eliminate false positives in the background. Recently, there are significant advances in object detection from images, which provides a very competitive and affordable solution.

The traditional approach of combining a feature extractor and a classifier has dominated object detection for a long time. A number of researches applied SVM [4] and Adaboost [5] in vehicle detection. In [6], SVM classification was used to classify extracted Haar features. The integration of HOG features and SVM has shown to detect pedestrians effectively [7] and was also applied to vehicle detection in [8], [9]. Edge features were classified for vehicle detection using SVM in [10]. In [11], vehicles were detected by classifying Legendre moments and Gabor features, using SVM classification. AdaBoost [12] is very fast and effective in face detection largely owing to its integration in a cascade scheme. It was applied to vehicle detection in [13] based on symmetry features and edge features [14], respectively. The combination of Haar-like features and AdaBoost classification has been used to detect rear faces of vehicles in [15] and [16]. An important observation is that the feature extraction-classification paradigm is empirically proved to be quite effective for objects fully visible with expected viewing angle. For example, rear-view vehicle detector cannot detect side vehicle.

Robustly detecting partially occluded vehicles or those captured from arbitrarily viewing angles is a great challenge because

their appearances vary to a large extent. Early works in this area were done by detecting a combination of independent vehicle parts. The general idea is to combine multiple different part-detecting classifiers and apply the spatial constraint of each part to eliminate false alarms. In [17], Haar features combined with AdaBoost classifier were used to detect front body, front wheel, rear wheel, and rear body with four different classifiers. Thus, a side vehicle can be successfully detected under varying lighting conditions, at different vehicle poses, and in the presence of partial occlusions. In [18], combination of Haar features and AdaBoost classification was used to detect front body and rear body of a vehicle at intersections. In [19], Lin et al. combined speeded-up robust features (SURF) and edge features to detect vehicles, with vehicle parts identified by keypoint detection. In [20], vehicles were detected as a combination of parts, using scale invariant feature transform (SIFT) features and hidden conditional random field classification. In [21], Chavez-Aragon et al. proposed to detect fourteen different car parts for vehicle detection and spatially constrain the detected parts to reduce false detection.

The deformable parts-based model (DPM) [22], using HOG features and the latent SVM, has been used for on-road vehicle detection in [23] and [24]. DPM can successfully handle deformable object detection even when the target is partially occluded. However, it leads to high computational costs due to a large number of repeated feature extraction and classification tasks in a sliding window search framework.

Recently, image classification has been significantly improved by deep ConvNets, such as Alexnet [25], GoogLeNet [26], VGG16 [27], Inception-v2 [28] and the powerful ResNets [29]. Using a well pre-trained CNN (in ImageNet classification [30]) as the basenet for a CNN-based object detector is beneficial to train an accurate detector. In other words, the better the basenet, the higher the detection capability of the corresponding detector.

The success of region-based convolutional neural networks (RCNNs) [31], which can detect partially-occluded objects without specifically modifying the training algorithm, is significant because it outperforms DPM to a great extent. Thus, most of the subsequent approaches were all powered by deep learning. Although RCNNs are computationally expensive originally, their computational cost has been drastically reduced by sharing convolutions, inspired by [32], across proposals generated by Selective Search [33]. This idea brought up Fast R-CNN [34] which achieves near real-time efficiency using very deep networks, i.e., VGG16, when ignoring the time spent on region proposal generation. Proposal generation was the computational bottleneck for the phase of object detection and its computation was significantly reduced by region proposal network (RPN) which was proposed in Faster R-CNN [35]. Although Faster R-CNN achieves 5 FPS and 7 FPS using ResNet-101 and VGG16 respectively, it is still far from the requirement of real-time object detection. i.e., 30 FPS. The major reason comes from the two-stage framework: the first stage generates region proposals that are expected to contain all objects with a small amount of negative windows, and the second stage classifies them into different classes of objects or background.

The single-stage YOLO detector [36] made a break-through by realizing an end-to-end framework for object detection. It reframes object detection as a single regression problem as how CNNs are applied for image classification. The last layer is simply the probability of an object, the class it belongs to and its location in the image. This work achieves 45 FPS for PASCAL VOC 2007 [37] test dataset with mAP 63.4%. Its sped-up version, Fast YOLO [36], further reaches 155 FPS with mAP 52.7% on the same dataset. However, poor localization precision was also reported according to the analysis in their work. Then, a multi-scale version of the single-stage detector, SSD [38], demonstrates significant improvements in terms of mAP by predicting objects from lower to top layers. SSD300 and SSD500 could achieve 72.1% and 75.1% mAP with 46 FPS and 19 FPS, respectively. An important difference between YOLO and SSD is that the former would only produce 98 bounding boxes by interpreting the last layer while the latter would generate thousands (SSD300) or even tens of thousands bounding boxes (SSD500). YOLOv2 proposed in YOLO9000 [39] further boosts the mAP to 78.6% with 40 FPS.

Recently, the increasing needs of running deep neural networks on embedded system encourage the study on efficient model design. SqueezeNet [40] reduces parameters and computation significantly while maintaining Alexnet-level accuracy. Both ShuffleNet [41] and MobileNetV1 [42] are designed specifically for mobile or embedded devices with limited computing power and could achieve significantly higher ImageNet classification accuracy over Alexnet. Shufflenet introduces pointwise group convolution and channel shuffle to greatly reduce computation cost. MobileNetV1 uses depthwise separable convolutions to build light weight deep neural networks. MobileNetV2 [43] proposes an inverted residual structure to further improve the performance.

It is worth mentioning that the performance of the detectors introduced previously are all assessed on powerful GPU. For example, SSD300 could achieve 46 FPS on Maxwell Titan X but its FPS would drastically drop to 11 FPS [44] when it is deployed on NVIDIA TX2. However, using MobileNetV1 as the basenet of SSD300, its inference speed would be significantly increased to 73 FPS (TX2) with slightly inferior mAP (68.0%) on PASCAL VOC 2007 test dataset [45]. SSDlite [43] is a variation of SSD by replacing the regular convolutions with separable convolutions (depthwise followed by 1×1 projection) in SSD prediction layers. SSDLite-MobileNetV2 could reach 70.9% mAP and 60 FPS (TX2) [45]. Finally, compared to other basenets, the advantage of using MobileNets is more significant on smartphone because depthwise separable convolutions are not directly supported in GPU firmware (the cuDNN library) [46].

Besides using more efficient basenet, specifically-designed hardware or software package could also directly boost the FPS of object detection models. For example, NVIDIA TensorRT [47] is a C++ library that facilitates high performance inference on NVIDIA GPUs and could deliver up to 3.7x faster inference. Generally speaking, YOLO and YOLOv2 are easier to be further sped up by TensorRT because their inference is exactly the same as a classification CNN.

In order to better accommodate objects of different sizes, Faster R-CNN reported that using 9 anchor boxes is most beneficial in terms of mAP boost while the size and aspect ratio for each prior are manually selected. Recently, the idea of multi-scale feature map is introduced so that receptive fields match objects of different scales. As opposed to Overfeat [48] and YOLO that operate on a single scale feature map, SSD proposed to detect objects at different sizes from lower to higher layers. MS-CNN [49] carefully selects feature maps from early layers to higher ones for generating region proposals from different scales. Hypernet [50] aggregates hierarchical feature maps using pooling and deconvolution layers to form hyper feature maps where RPN generates region proposals from. YOLOv2 adds a passthrough layer to concatenate high-resolution features with the low-resolution ones. MFR-CNN [51] further incorporates multi-scale features and global information to achieve better performance on traffic object detection.

Hard negative mining was a very popular training technique for enhancing the object detection accuracy. The basic procedure is to collect initial training data, freeze this model, run detection on validation data with Ground-Truth, collect false-positive results and re-train the model with the false detection results. Within some iteration cycles, the detection capability could be quantitatively boosted. Online hard example mining (OHEM) [52] introduced an online framework for region-based detectors, such as Fast R-CNN, to perform BP for foreground or background objects with higher loss. OHEM claimed that, in Fast R-CNN, only training with hard examples without keeping the original 1/3 fg-bg (foreground-background) ratio would lead to significant mAP boost. YOLO also suffers from the fg-bg imbalance because most images contain only a few objects so they specifically set the parameters for balancing the fg-bg loss. However, the imbalance makes YOLO sensitive to the weighting parameters in the loss function.

Non-maximum suppression (NMS) is a post-processing step used to obtain the final detection results. It sorts all detection boxes on the basis of their scores and then the detection bounding box with the maximum score is selected. Finally, all others with considerable overlaps with the maximum-score bounding box (using a pre-determined threshold) are suppressed. Soft-NMS [53] was proposed to decrease the detection scores of these overlapped detection bounding boxes with a continuous function of the corresponding overlap size. It is proven effective for boosting the detection accuracies of two-stage object detectors such as Faster RCNN.

Most on-road vehicle datasets are captured with limited viewing angles or at fixed distances. LISA 2010 dataset [54] is entirely composed of rear-viewed front vehicles. Urban Traffic Surveillance (UTS) dataset [55] provides vehicle images captured by surveillance systems. KITTI dataset [56] is specifically designed for autonomous driving and all images are collected in real-driving scenario. Vehicles in this dataset suffer from scale variation, occlusion and truncation problems. Furthermore, cars and buses in PASCAL VOC 2007 and 2012 are more diverse and challenging because their appearances are fully or partially seen at different distances, aspect ratios, sizes and viewing angles. Car simulation software, such as Carsim [57], could easily generate a

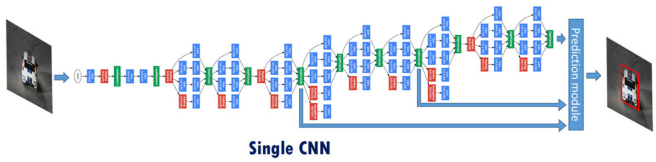

Fig. 1. Our CNN model.

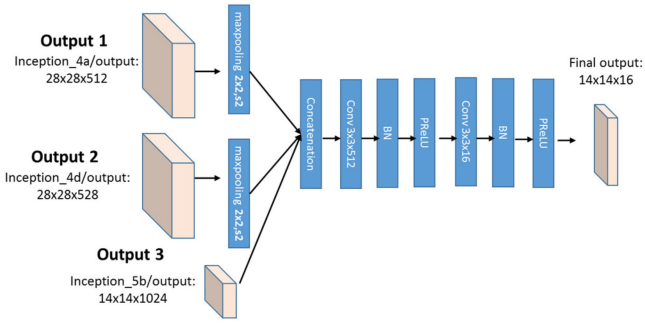

Fig. 2. Our prediction module.

massive amount of data with auto-generated ground truth (GT). In this paper, we also investigate the possibility of using Carsim to generate training/testing data in vehicle detector development. iROADS [58] is a comprehensive vehicle dataset recorded from on-road moving vehicles under various weather conditions as well as challenging lighting conditions, including day light, night light, tunnel light, rainy day, rainy night, snowy, and sun strokes. Our detector is tested on all real-driving video sequences provided by iROADS in order to assess the robustness of the proposed detector in versatile driving scenarios.

Inspired by the advantages and drawbacks of previous works, in this paper, we propose a real-time single-stage vehicle detector which could be deployed on NVIDIA DrivePX2 under real-time constraints. The output of our proposed CNN is two-dimensionally interpreted to detect the vehicles in a given image. The major contributions of this work include (1) a multi-scale prediction module which carefully merges features from low-level features to high-level semantic information across different layers, (2) a heavy data augmentation strategy specifically designed for vehicle detection and (3) a multi-stage image-based OHEM (MSI-OHEM) strategy which significantly boosts the AP by fine-tuning the CNN with hard examples and the ones with slightly-insufficient IOU to be considered true positives. Comparisons with the state-of-the-art methods on some benchmark datasets demonstrate the advantages of the proposed method for vehicle detection.

II. Methodology

This section describes the details of our proposed detector and the associated training methodology. The structure of our model is illustrated in Fig. 1 and is built by stacking a prediction module to the end of a basenet. We take GoogLeNet for illustration and the best detection accuracy is reported by using ResNet-50 as the basenet. As shown in Fig. 2, our prediction module is composed of concatenated multi-scale feature maps subsequently processed by two convolution layers, each followed by Batch

Normalization [59] and PReLU [60]. The last layer is the output to be spatially interpreted as a fixed number of grids responsible for detecting objects whose centers of the object windows are located inside. The last step to produce the final detection bounding boxes of a detector is usually done by applying non-maximum suppression (NMS) to select the ones with the maximum confidence score. We applied Soft-NMS [53] to our model. The whole detector is trained with the proposed data augmentation strategy using standard BP until the network is converged. Then, the proposed Multi-Stage Image-based OHEM (MSI-OHEM) is applied to fine-tune the whole network for further boosting the detection accuracy.

In the following, we describe the details of the proposed detection model and the associated training strategies.

A. *Efficient Basenet*

Most object detectors apply CNNs pretrained in ImageNet classification for the subsequent detector training. Generally speaking, using a better basenet is beneficial for the subsequent detection task. However, detector using very deep networks, such as ResNet-101 or ResNet-152, would be difficult to meet the real-time requirement. Therefore, in this work, we chose MobileNetV2, GoogLeNet, Inception-v2 and ResNet-50 as the basenet because they provide good compromised solutions based on the considerations of computation speed and detection accuracy.

The input image of a CNN-based detector is normally bigger than 224×224. Our detector uses a 448×448 image as our input. However, instead of training higher resolution images from scratch, we simply fine-tune the pretrained CNN with the new resolution images. Such a strategy improves the top-5 accuracy for using MobileNetV2/GoogLeNet/Inception-v2/ResNet-50 as basenet from 0.901/0.891/0.920/0.916 to 0.910/0.911/0.929/0.939.

B. *Data Augmentation*

Data augmentation is usually employed for training a classifier or a detector. Different from the traditional strategy, our design tries to enhance the detection capability even when the vehicles are heavily occluded. Therefore, we resize the training images with 0.1 to 0.9 of the original sizes if the area of the vehicle instance is above a predetermined threshold and the images are horizontally flipped with probability 0.5. In addition, we introduce a random truncation strategy to truncate a vehicle with probability 0.5 by a new window such that the vehicle contains at least 25% of its appearance visible. In the subsequent recall analysis of our data augmentation strategy, it is quantitatively proven useful in detecting occluded vehicles. Although only a few vehicles in the training dataset are partially visible, the ones that are entirely visible were chosen to be randomly resized, rotated, horizontally-flipped, recolored, cropped or put in the boundary to make them truncated to a random extent for training the vehicle detector. Our detector is therefore very robust in learning the vehicle appearance under large variations, even with large partial occlusions. Finally, in order to improve the capability in detecting vehicles with fuzzy appearance, we randomly blur the training images with either mean filter, median filter or, Gaussian blur. Some examples of the augmented images are shown in Fig. 3.

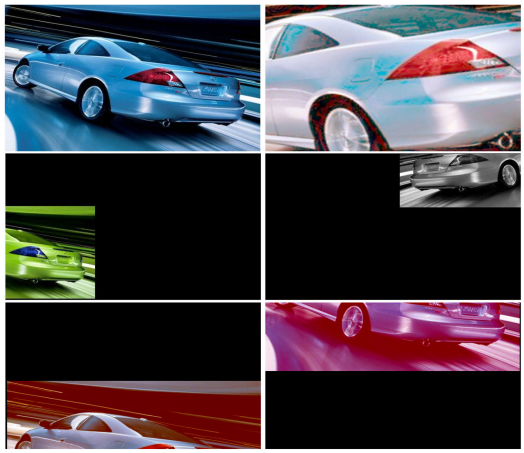

Fig. 3. Examples of our data augmentation strategy: the top-left image is the original training image and the others are all augmented ones.

C. *Multi-Scale Feature Maps and Bounding Box Priors*

Objects can appear in an image within a large range of scales. A single receptive field cannot accommodate large scale variations for object sizes. In the GoogLeNet/Inception-v2 version of our model, i.e., our prediction module is stacked on top of GoogLeNet/Inception-v2, we approach the idea of multi-scale features by concatenating feature maps from inception_4a, inception_4d and inception_5b. Then, the concatenated feature maps are subsequently processed by our prediction module which is composed of two convolution layers, each followed by Batch Normalization and PReLU.

Faster R-CNN and SSD predict bounding boxes using hand-picked priors. Since there is no anchor box strategy in YOLO, its main detection errors as described in their work [36] could be attributed to localization error. However, instead of hand-picked priors, we perform k-means algorithm to explore better priors. In the experiment of testing on PASCAL VOC dataset, we found that using 3 priors (59×46, 198×124, and 414×245) reached the best performance. The final AP achieved by using different numbers of priors will be discussed in the subsequent model analysis section.

D. *Non-Maximal Suppression*

Non-maximum suppression (NMS) is commonly used for eliminating overlapping results to obtain the final detection results. Soft-NMS has been proven effective in two-stage detectors. Our detector has also benefitted from the linear version of Soft-NMS. The performance gain would be shown in the ablation study.

E. *Multi-Stage Image-Based Online Hard Example Mining*

Hard negative mining is a classical training strategy in training an object detector. OHEM introduced a framework for

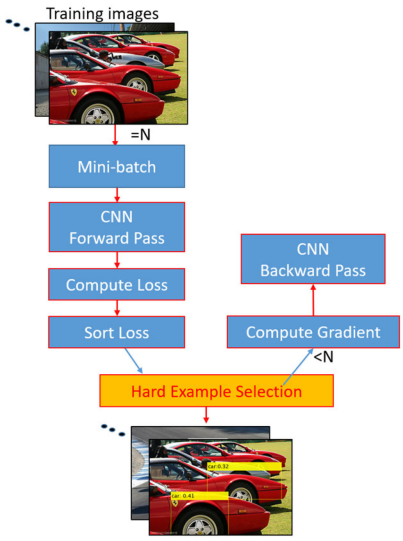

Fig. 4. The framework of OHEM applied to a single-stage detector.

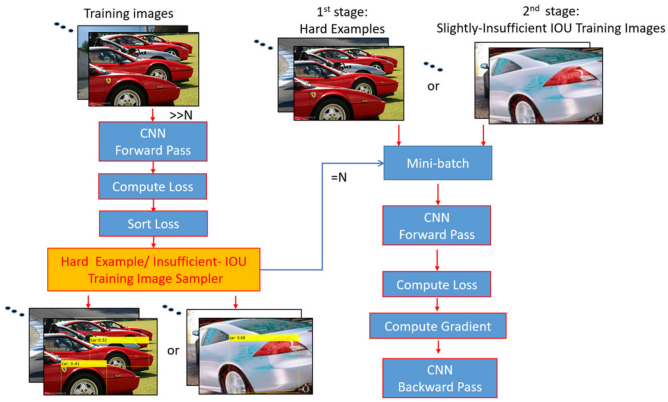

Fig. 5. Our MSI-OHEM in training the proposed single-stage detector.

region-based detectors, such as Fast R-CNN, to perform BP for foreground or background objects with higher loss. However, this framework could only be applied for region-based detectors.

We propose Multi-Stage Image-based OHEM (MSI-OHEM) to fine-tune a converged CNN trained by standard BP. A naïve way to implement OHEM for training single-stage detectors is shown in Fig. 4. In a given mini-batch of N images, the multi-task loss in every training image is sorted. Then, some training images with lower loss values are nullified to do BP. The cost of such BP is nearly the same as before and no additional mechanism is applied. This is similar to the idea of region-based OHEM but the AP boost is marginal in our case because such a framework would only perform BP by selecting the images with higher loss values in a single mini-batch. In fact, after our detector is trained using standard BP, global hard examples are sparsely distributed in training images. Therefore, the training images associated with higher loss values in each mini-batch might not be quantitatively hard enough.

As shown in Fig. 5, our MSI-OHEM is designed to fine-tune a converged CNN with two additional stages, each of which is designed to boost the AP from a different perspective. In the first stage, inference is stopped until the number of hard images associated with loss higher than a predetermined threshold is reached. Typically, it is the number of a mini-batch. An alternative way is to infer every training image and pick the N top losses. However, it is quite time consuming because we have to perform inference for every training image before performing a single BP. Nevertheless, the loss function is the same as the standard training. What we have done so far is to pick the ones with higher loss and accumulate them until the number of mini-batch is reached. Now, our detector is supposed to have learned how to eliminate those false positives in non-object grids and improve the class score, the objectness score and the bounding box precision of the object grids at the same time.

Sometimes, some detected objects in a training image are considered false negatives because of slightly-insufficient IOU. The total loss of the entire image might be low because the objectness of the non-object grids is low. A key observation that inspired us is that by just lowering the required IOU (0.5 from PASCAL) for 0.01, the AP would increase at most 1%. Therefore, for the 2nd stage of MSI-OHEM, the optimization goal is towards picking the training images with higher location loss in the predetermined range (as will be described in the subsequent loss function section) with sufficient objectness for the object grid(s), and low objectness for non-object grids. After training with these images, our detector would learn to better detect the vehicles which were originally detected as false negatives. This is because, in standard BP, these important training images are mostly eliminated due to their less noticeable gradient to guide the optimization. As can be seen in the following experiment, it can boost the AP for object detection.

F. Loss Functions of Training and Fine-tuning

Our training includes one stage of regular CNN training and two stages for fine-tuning. The regular training stage applies object loss and non-object loss as eq. (1) and eq. (2), given below. The first stage of the fine-tuning adopts the same loss functions, but the whole CNN is fine-tuned using the proposed MSI-OHEM. The second stage of the fine-tuning fine-tunes the network with a different loss function designed for IOU boosting in conjunction with our MSI-OHEM strategy.

At the beginning of training, the class scores, the objectness scores and the bounding boxes are learned altogether. For an object whose center of object window located inside grid ij, the loss function is defined by

$$\sum_{n=1}^{k}\left(\sum_{p\in\{w,h\}}\alpha_p\left(\bar{G}_{ij}^{p}(n)-\hat{G}_{ij}^{p}\right)^2+\sum_{q\in\{x,y\}}\alpha_q\left(G_{ij}^{q}(n)-\hat{G}_{ij}^{q}\right)^2+\alpha_{object}\left(G_{ij}^{o}(n)-1\right)^2\right)+\left(G_{ij}^{c}-1\right)^2, \quad (1)$$

where G_{ij}^{c} is the class score of grid ij. $G_{ij}^{o}(n)$, $G_{ij}^{x}(n)$, $G_{ij}^{y}(n)$ are the objectness score and the coordinate of the n-th predicted output. $\bar{G}_{ij}^{w}(n)$ and $\bar{G}_{ij}^{h}(n)$ are the relative width and height with

reference to the n-th bounding box prior, $W(n)$ and $H(n)$, which were pre-estimated by k-means clustering on the validation data. They are related to the predicted width, $G_{ij}^{w}(n)$, and height, $G_{ij}^{h}(n)$, of the n-th bounding box by $\bar{G}_{ij}^{w}(n) = G_{ij}^{w}(n)/W(n)$ and $\bar{G}_{ij}^{h}(n) = G_{ij}^{h}(n)/H(n)$, respectively. $\hat{G}_{ij}^{x}$, $\hat{G}_{ij}^{y}$, $\hat{G}_{ij}^{w}$ and $\hat{G}_{ij}^{h}$ represent the location and size GT of the object bounding box located at grid ij.

For a non-object grid, i.e., no object whose center of object window locates inside grid ij, its loss function is given by

$$\sum_{n=1}^{k} \alpha_{non-object} (G_{ij}^{o}(n))^2 + (G_{ij}^{c})^2. \tag{2}$$

There are 2 terms for the non-object grids. The first one is the objectness score corresponding to each predicted output and the second is the class score of this grid. Although we only focus on detecting one type of object, i.e., vehicle, in this work, we still keep the class score typically used for multi-class object detectors. This is because using both the class score and objectness score would slightly boost the AP compared to that using only the objectness score. In this work, we explored the optimal hyperparamters in the validation dataset and they are $\alpha_x = \alpha_y = 1$, $\alpha_w = 1.2$, $\alpha_h = 1.6$, $\alpha_{\text{object}} = 1$, and $\alpha_{non\text{-}object} = 0.65$. The analysis of these hyperparameters and their impact on the testing results will be discussed in the subsequent model analysis section.

The loss function designed for the second stage of the fine-tuning is to fine-tune the whole CNN by the training images with slightly-insufficient IOU but with strong objectness for object grids and low objectness for non-object grids. Therefore, the so-called hard examples turn out to be the images with the aforementioned conditions but with higher location loss (slightly-lower IOU) in a given range. This loss function is designed to fine-tune the whole network in an attempt to increase the IOU of each detected object whose objectness score is high but its IOU is below 0.5 (PASCAL) and 0.7 (KITTI car), which are the thresholds for determining if an object is successfully detected with a precise bounding box. We define a new loss function which only fine-tunes the detected objects whose IOU is slightly lower than the required value to be considered true positive (TP). For an object grid with IOU higher than β (0.4 for PASCAL VOC, and 0.6 for KITTI car) and objectness score higher than γ (0.6), its loss function is defined by:

$$\sum_{n=1}^{k} \left(\sum_{p\in\{w,h\}} \alpha'_p (\bar{G}_{ij}^{p}(n) - \hat{G}_{ij}^{p})^2 + \sum_{q\in\{x,y\}} \alpha'_q (G_{ij}^{q}(n) - \hat{G}_{ij}^{q})^2 \right). \tag{3}$$

Since this is a fine-tuning procedure, the learning rate is as low as that at the end of regular training. Besides, the parameters, α'_p and α'_q are set to 1/100 of their original values in training by using eq. (1).

III. Experimental Results

We evaluate the proposed detector on PASCAL VOC 2007 dataset, KITTI detection benchmark, (self-collected) Carsim-generated dataset and iROADS dataset, and the latter two are used for testing only. The training in all of our experiments follows the same setting. Each grid in the 14×14 feature maps contains 3 objects with 3 corresponding objectness scores and 1 class score. Additionally, there are 12 values describing the object windows for the three objects as shown in Fig. 2. The starting learning rate is 10^{-3} for 2 k iterations. Then, we continue the training with 10^{-2}, 10^{-3}, and 10^{-4} for 16k iterations, respectively. In both stages of the fine-tuning, the learning rate is 10^{-4}. The weight decay and momentum are set to 0.0005 and 0.9, respectively, in the regular training and fine-tuning.

The total numbers of layers in our models using MobileNetV2/GoogLeNet/Inception-v2/ResNet-50 as the basenet are 56 (54+2), 24 (22+2), 36 (34+2) and 52 (50+2), which occupy 1113, 629, 707 and 1375 MiB (Mebibyte) in GPU during runtime in Caffe [61] framework, respectively.

Our model applying MobileNetV2/GoogLeNet/Inception-v2 (33/30/25FPS on GP106 and 51/48/39 FPS on Titan X) as the basenet achieves 85.35%/85.62%/86.49% AP with real-time computing performance on DrivePX2. The best result in terms of AP is the ResNet-50 version (87.81% AP with 13 FPS on GP106 and 21 FPS on Titan X). When NVIDIA TensorRT is used for speeding up our model, MobileNetV2, GoogLeNet, Inception-v2 and ResNet-50 versions could run at 64/58/48/28 FPS, respectively.

A. PASCAL VOC2007

In VOC 2007 trainval and VOC2012 trainval, there are totally 16551 training images. In VOC2007 test, there are 4952 images. However, our work only focuses on the vehicle class which is the superclass of car-class and bus-class. If we tried to select training images of car (1161+721) or bus (421+186), the AP might not be promising because the appearance of bus is essentially similar to the one of car. Some training or testing photos contain both of them. If both classes are trained separately, the other would be treated negative data and this is harmful to learn the essence of vehicle appearance. Therefore, we defined a new class called vehicle which is composed of car & bus classes for both training (1420) and test images (825).

In Table I, we compare our detectors to other famous detectors in terms of training data and FPS (assessed on Maxwell Titan X). As can be seen from the table, our vehicle detector, MobileNetV2 version, is faster than the others except Fast YOLO. Table II summarizes the Average Precision (AP) for vehicle detection by applying different detectors on PASCAL VOC 2007 test dataset, and the results show that our model achieves state-of-the-art AP in this experiment. Although other detectors are designed for multi-class detection, our work could be easily extended to detect more classes of objects by including additional feature maps with the number the same as the total number of classes in the last layer. In our experiment, the processing time would only increase less than 1 millisecond (from 1-class to 20-class) on average for an image, so such a comparison is

TABLE I
DETECTION FRAMEWORK ON PASCAL VOC 2007
(BASENETS ARE INDICATED IN BRACKETS)

Detection Frameworks	Train	FPS
Fast R-CNN	2007+2012	0.5
Faster R-CNN (VGG-16)	2007+2012	7
Faster R-CNN (ResNet-101)	2007+2012	5
YOLO	2007+2012	45
Fast YOLO	2007+2012	155
SSD300	2007+2012	46
SSD500	2007+2012	19
YOLOv2 544x544	2007+2012	40
MFR-CNN	2007+2012	7
This work (MobileNetV2)	2007+2012	51
This work (GoogLeNet)	2007+2012	48
This work (Inception-v2)	2007+2012	39
This work (ResNet-50)	2007+2012	21

TABLE II
PASCAL VOC DETECTION RESULTS
(BASENETS ARE INDICATED IN BRACKETS)

	Bus	Car	Vehicle (Car+Bus)
Fast R-CNN	81.6	78.6	80.10
Faster R-CNN (VGG-16)	83.1	84.7	83.90
Faster R-CNN (ResNet-101)	85.1	85.3	85.15
YOLO	71.0	65.0	68.00
Fast YOLO	60.0	67.0	63.50
SSD 300	81.1	80.8	80.95
SSD 500	84.9	85.6	85.25
YOLOv2 544x544	84.2	85.7	84.92
MFR-CNN	82.1	87.0	84.55
This work (MobileNetV2)			85.35
This work (GoogLeNet)			85.62
This work (Inception-v2)			86.49
This work (ResNet-50)			87.81

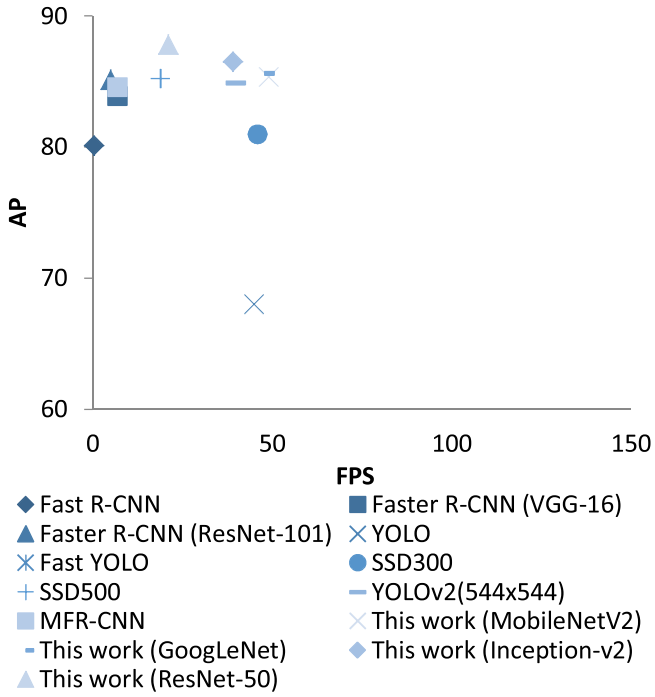

Fig. 6. AP and FPS on VOC07 test dataset.

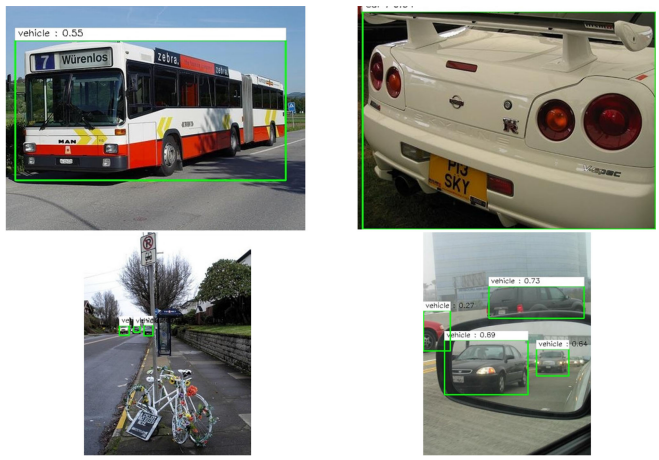

Fig. 7. Four typical scenarios of vehicle(s) expected to be detected. From top-left, top-right, bottom-left to bottom-right: one vehicle fully-seen, one vehicle partially-seen, small and fuzzy vehicles & occluded/truncated vehicles.

fair. Fig. 6 depicts the comparison of all detectors in terms of AP and FPS.

Fig. 7 demonstrates some example detection results by using our detector on PASCAL VOC 2007 test. The top-left image shows the easiest scenarios because only an entirely-seen vehicle is inside. To detect a vehicle partially seen as in the top-right image in Fig. 7 is challenging because the detector needs to be very robust to localize a vehicle from randomly truncated window of a vehicle body. Using bounding box priors is helpful in this kind of scenario because our detector would learn to better accommodate vehicles with wider variations in size. In the bottom-left image, we can see that the detection capability of small objects is enhanced. It is achieved by shrinking the training images containing vehicles and applying multi-scale feature to produce larger responses from those small vehicles. The bottom-right image demonstrates the detection results of truncated or occluded vehicles. Therefore, it is obvious that the proposed data augmentation strategy successfully boosts our detector accuracy, which will be analyzed in the subsequent ablation study and recall analysis sections. In our implementation, we generate 9500 training images by using the proposed data augmentation strategy for the training.

B. KITTI Val Set Evaluation

In this section, we evaluate the proposed vehicle detector on the challenging KITTI object detection benchmark dedicated to autonomous driving. This dataset is composed of 7481 training images and 7518 testing images. Since GT annotations for the testing set are not released, we use train/validation (3712/3769 images) splits, as indicated by [62, 63], from the training set to validate our method.

TABLE III
VEHICLE DETECTION RESULTS ON KITTI VAL SET

		AP(IOU=0.7)		
Method	FPS	Easy	Moderate	Hard
YOLO	45	19.82	15.63	13.17
Faster-RCNN (VGG-16)	7	61.01	54.62	47.51
This work (MobileNetV2)	51	61.58	51.98	46.94
This work (GoogLeNet)	48	62.51	52.45	47.45
This work (Inception-v2)	39	64.80	56.15	47.55
This work (ResNet-50)	21	65.22	57.82	48.12

Fig. 8. KITTI detection result comparison: Left column: detection results of this work (ResNet-50); Right column: detection result of Faster R-CNN (VGG-16).

In Table III, the accuracy for each model with the setting that leads to the best result in PASCAL VOC is shown. Although the accuracies of the MobileNetV2 version of our model are similar to those of Faster R-CNN, it could run nearly 7 times faster. Our Inception-v2 and ResNet-50 versions outperform Faster R-CNN (VGG-16) for all levels and are still 5.6/3 times faster. It is worth mentioning that typical single-stage methods, such as YOLO, fail at achieving very accurate results. The detection results of this work against Faster R-CNN (VGG-16) are shown in Fig. 8. Our detector works better when vehicles are truncated or partially-occluded. Finally, we generate 8000 training images by using the proposed augmentation technique to achieve the best results.

C. CarSim-Generated Data Evaluation

CarSim-generated driving data is not realistic enough to replace real-world driving data for training but it could be used to test a vehicle detector given vehicles viewed at any designated viewing angles without any manual labeling.

In order to assess whether or not our model trained by VOC07+12 trainval would be capable of detecting on-road vehicles, we use CarSim to generate on-road vehicles with auto-generated GT. As shown in Fig. 9, there are totally 12 vehicles of different types, including sedan, truck, SUV, etc. In scenario-1, each vehicle maneuvers with the same pre-set trajectory and the length of each video is the same (1052 frames). Totally, there are 12624 frames containing different types of vehicles in various road environments.

For scenario-2 to -5, vehicle-1, -6 and -8 in Fig. 9 are selected for conducting the following experiments. Each setting, in terms of a fixed distance and tilt angle pairs, ranging from (4.4 m, 0°), (8 m, 0°), (5 m, 25°), (8 m, 20°), creates 360-degree view of the selected vehicle for 241 frames. We compare YOLO, Faster R-CNN (VGG-16) with our work (GoogLeNet) on all of the CarSim-generated data.

Fig. 9. 12 vehicle types in Carsim from top to bottom and left to right.

TABLE IV
AP COMPARISON OF YOLO, FASTER R-CNN AND OUR WORK IN 5 SCENARIOS

	YOLO	Faster R-CNN (VGG-16)	This work (GoogLeNet)
Scenario 1 (12 vehicles)	0.676	0.999	0.996
Scenario 2 (3 vehicles)	0.873	1	1
Scenario 3 (3 vehicles)	0.955	1	1
Scenario 4 (3 vehicles)	0.685	0.963	1
Scenario 5 (3 vehicles)	0.673	0.982	0.999

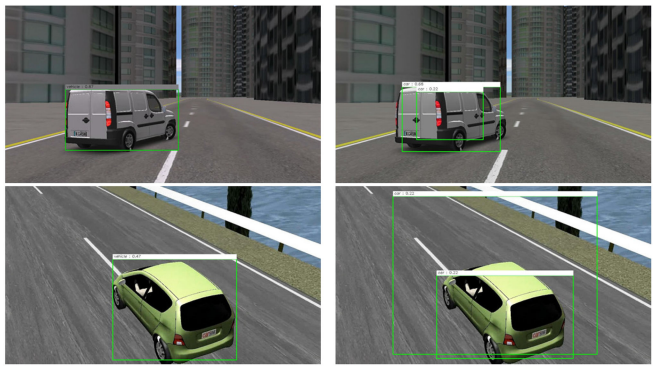

Fig. 10. Carsim detection result comparison; Left column: detection results of this work (ResNet-50); Right column: detection result of YOLO; 1st row: scenario-1 (vehicle-3); 2nd row: scenario-4 (vehicle-1).

As can be seen in Table IV, YOLO underperforms others significantly in scenario-1. This is because the vehicles are relatively small in the maneuvering. Furthermore, YOLO suffers from false positives in the sense that there are sometimes two bounding boxes for a single vehicle, as shown in the right column of Fig. 10. The main reason comes from the poor size estimation of the bounding boxes so even NMS could not eliminate one of them. Our work consistently outperforms YOLO in all scenarios, and it is more accurate than Faster R-CNN (VGG-16). In addition, our detector (GoogLeNet version) is nearly 7 times faster

TABLE V
AP COMPARISON OF YOLO, FASTER R-CNN AND OUR WORK IN 7 SCENARIOS IN iROADS DATASET

Scenario	YOLO	Faster R-CNN (VGG-16)	This work (MobileNetV2/GoogLeNet /Inception-v2/ ResNet-50)
Day light (903 imgs)	38.1	76.7	85.2/86.8/87.8/88.2
Night (1050 imgs)	5.6	33.9	34.0/34.6/38.0/39.0
Rainy day (1049 imgs)	26.5	61.3	67.0/68.6/70.5/71.2
Rainy night (431 imgs)	7.8	32.7	30.2/30.1/31.5/32.5
Snowy (569 imgs)	26.7	66.0	66.2/67.3/68.5/69.5
Sun stroke (347 imgs)	20.9	27.1	32.6/34.3/34.3/35.6
Tunnel (307 imgs)	13.5	42.8	36.3/35.5/38.8/42.0

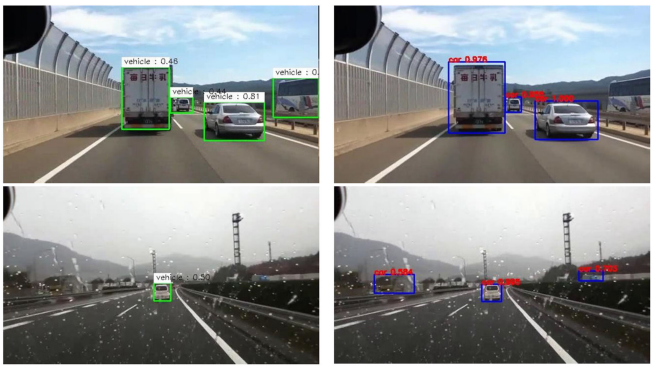

Fig. 11. iROADS selected detection results. Left column, detection results of this work (ResNet-50); Right column: detection result of Faster R-CNN (VGG-16); 1st row: Daylight; 2nd row: RainyDay.

than Faster R-CNN. The demonstration videos which show the detection results in Table IV are provided in the supplementary material.

D. iROADS Dataset Evaluation

iROADS dataset is composed of 4656 image frames in 7 categories including daylight, night, rainy day, rainy night, snowy, sun strokes, and tunnel. In this experiment, we train each detector with VOC 07+12 trainval (car+bus) and KITTI-train (from the aforementioned KITTI train/val split) datasets. The proposed detectors outperform YOLO and Faster R-CNN (VGG-16) in almost every scenario, as shown in Table V. Fig. 11 depicts some examples of detection results by using the proposed detector (ResNet-50) and Faster R-CNN.

IV. MODEL ANALYSIS

In order to quantitatively improve our detector, we have carried out several controlled experiments to examine how each component affects the detection accuracy. For all of the following experiments, the dataset is PASCAL VOC with the same training iteration and learning rate.

A. Ablation Study

We made some improvements based on the original version of our work to achieve the final result in PASCAL VOC. A summary of the ablation study could be found in Table VI.

B. The Impact of Hyperparameters

In our pursuit of better quantitative results in PASCAL VOC dataset, the impact of different values of hyperparamters is also analyzed. In Table VII, the hyperparameters that lead us to the highest detection accuracy on VOC07 val dataset are used to verify if they could also work well on the unseen VOC07 test dataset. Finally, these hyperparameters are used to train our model using both training and validation datasets and the highest average precision is thus obtained, which is common practice in other works, such as Faster R-CNN. It is worth mentioning that these hyperparameters are not carefully chosen for a particular dataset. Our model in other experiments (datasets) of this work has also achieved results better than other detectors using the same hyperparameters.

C. Basenet Comparison

ResNet family is highly possible to boost the detection capability of a detector because ResNet-101 and ResNet-152 have achieved very high top-1 and top-5 accuracies. Using them as the basenet is beneficial to boost the detection accuracy. However, in consideration of the possibility to deploy our detector on an embedded system, only ResNet-18, ResNet-32 and ResNet-50 are considered in our experiments. VGG-16 is also involved in the basenet analysis but we found it is more efficient to remove its fully-connected layers in conducting the same analysis so the FPS is much faster than its original version.

To fairly analyze the performance of basenets, we simply add two more convolution layers on top of each of them. The input image size is 448×448 and the output grid setting is 7×7, as the basic version of our work in the ablation study. In addition, each basenet is fine-tuned with 448×448 ImageNet data. Overall, ResNet-50 leads to the highest AP in our initial evaluation, as shown in Table VIII. However, since real-time computation (on an embedded platform such as DrivePX2) is the other important consideration to ADAS and autonomous vehicle, MobileNetV2, GoogLeNet or Inception-v2 is also a good choice due to its high computational efficiency.

D. Recall Analysis of the Proposed Data Augmentation Strategy

At the beginning, our detector encountered difficulties in detecting small vehicles and the vehicles that occupy most areas in the images. As previously-mentioned, our detector would be trained with randomly-resized and -blurred vehicles so as to better accommodate vehicles with extreme sizes and learn their appearance when they are not clearly-seen or look blurry. In

TABLE VI
DESIGN DECISIONS THAT LEAD TO BETTER PERFORMANCE IN TERMS OF AP FOR THIS WORK USING GOOGLENET AS BASENET

	Car	Bus	Vehicle (Car+Bus)									
448×448 Fine-tuning				√	√	√	√	√	√	√	√	√
Batch normalization + PReLU					√	√	√	√	√	√	√	√
14×14 prediction						√	√	√	√	√	√	√
Data augmentation							√	√	√	√	√	√
Multi-scale feature								√	√	√	√	√
Bounding box priors									√	√	√	√
Soft-NMS										√	√	√
MSI-OHEM 1st stage											√	√
MSI-OHEM 2nd stage												√
AP	68.2	69.1	68.8	70.1	74.9	77.5	81.2	81.8	82.4	83.1	84.5	85.6

TABLE VII
VEHICLE DETECTION RESULTS OF OUR MODEL (GOOGLENET) USING DIFFERENT HYPERPARAMETER SETTINGS ON VALIDATION AND/OR TESTING DATASETS

Hyperparameters	Training data	Testing data	
		VOC07 val	VOC07 test
α_x=α_y=1, α_w=α_h=1, α_{object}=1, $\alpha_{non\text{-}object}$=0.8	07+12 train	77.8	79.5
α_x=α_y=1, α_w=α_h=1.4, α_{object}=1, $\alpha_{non\text{-}object}$=0.6	07+12 train	76.4	79.8
α_x=α_y=1, α_w=1.2, α_h=1.6, α_{object}=1, $\alpha_{non\text{-}object}$=0.65	07+12 train	78.0	81.5
α_x=α_y=1, α_w=1.2, α_h=1.6, α_{object}=1, $\alpha_{non\text{-}object}$=0.65	07+12 trainval	-	85.6

TABLE VIII
AP ON VOC07 TEST WITH TRAINING DATA VOC 07+12 TRAINVAL OF OUR INITIAL MODEL

Basenet	FPS(TitanX)	FPS (GP106)	AP
MobileNetV2	58	39	70.0
GoogLeNet	56	36	70.1
Inception-v2	46	30	71.1
ResNet-18	58	33	68.5
ResNet-32	34	22	69.9
ResNet-50	24	16	72.1
VGG-16	42	27	69.7

order to quantitatively assess the capability of our detector with and without using the proposed data augmentation strategy, we conducted the following analysis. The vehicle objects in VOC07 test data can be categorized into the following categories: (1) one vehicle entirely seen, (2) one vehicle partially seen, (3) small and/or fuzzy vehicles, and (4) others.

TABLE IX
RECALL COMPARISON OF OUR MODEL BEFORE AND AFTER USING OUR TRAINING STRATEGIES

	One vehicle entirely seen	One vehicle partially seen	Small and/or fuzzy vehicle(s)	Others
Frames	383	51	27	364
Without Augmentation	0.831	0.705	0.148	0.612
With Augmentation	0.941	0.902	0.568	0.701

Fig. 12. Detection results of partially-seen vehicle (left) and fuzzy vehicle (right) after applying data augmentation strategy.

As can be seen in Table IX, the recall rate of the testing result in VOC07 test after applying the proposed data augmentation strategy to VOC07+12 trainval is significantly improved. In the left image of Fig. 12, due to the augmented strategy that truncates the entirely-seen vehicle in the training images, our detector has learned the essence of a vehicle body even though it is partially visible in the image.

For the case of small and/or fuzzy vehicles, due to the randomly-resized training images, the size of detectable vehicle becomes smaller even though multi-scale feature module is not applied yet. For the right image of Fig. 12, the vehicle inside cannot be detected originally because its appearance is fuzzy or

Fig. 13. Detection results of extremely-small vehicles after the multi-scale feature model is applied.

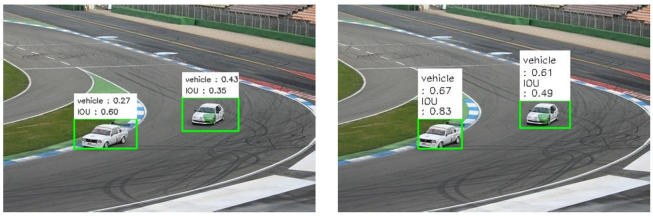

Fig. 14. Detection results before and after using bounding box priors. Left image: without priors. Right image: with priors.

not visually-prominent. However, it could be successfully detected after applying our data augmentation strategy, including random blurring and resizing. This is the reason why the recall rate is increased.

E. The Benefit of Multi-Scale Feature and Bounding Box Prior

A simple approach to directly enhance the capability of a model in detecting small vehicles is to enlarge the input image size at the cost of more computational resource. In order not to compromise between detection performance and FPS, mixing feature maps from different layers enable small objects to produce large responses. Although the AP boost (0.6) by using multi-scale features in Table VI seems marginal, it could bring more potential for detecting objects with large scale variations. For the images in Fig. 13, our original model fails to detect both vehicles. After the multi-scale feature module is used, both of them can be detected by our detector. However, only the vehicle in the right image is considered TP because the corresponding IOU passed the required threshold, i.e., 0.5. As to the slightly-IOU-insufficient vehicle in the left image, it becomes TP after the subsequent fine-tuning stage because of its strong objectness score higher than the required 0.6. It is worth mentioning that the confidence shown in Fig. 13 is the product of class score and objectness score.

To better detect objects with a wider range of scales and aspect ratios, we introduce bounding box priors. In the left image of Fig. 14, both vehicles are detected without using priors. It is obvious that insufficient IOU is observed. After bounding box priors are applied, the IOUs of both vehicles are significantly boosted. As to the right vehicle in the right image, its IOU is on the verge of becoming TP and is highly possible to be further promoted in the subsequent fine-tuning stage (MSI-OHEM) proposed in this work. To find an appropriate number of priors, we have also analyzed the final AP in our model using different numbers of priors. The prior analysis is done by using k-means algorithm on VOC07 val with training data VOC07+12 train. In Table X, we reached the best results in our GoogLeNet version when 3 priors are applied.

TABLE X
AP ON VOC07 VAL WITH TRAINING DATA VOC 07+12 TRAIN OF OUR MODEL (GOOGLENET) USING DIFFERENT NUMBERS OF PRIORS

Number of priors	Prior size (w×h)	Vehicle
2	86×61, 380×225	77.2
3	59×46, 198×124, 414×245	78.0
4	59×45, 192×124, 404×332, 414×203	75.8
5	50×38, 138×98, 264×161, 401×326, 437×203	76.2
6	50×38, 135×98, 256×139, 313×311, 421×181, 453×284	75.9
7	41×30, 88×71, 164×109, 270×130, 290×253, 428×336, 439×205	77.5
8	49×37, 127×93, 195×247, 244×117, 348×239, 366×437, 437×173, 458×286	77.6
9	39×29, 88×61, 94×167, 166×96, 260×134, 313×398, 330×237, 433×174, 456×286	77.4

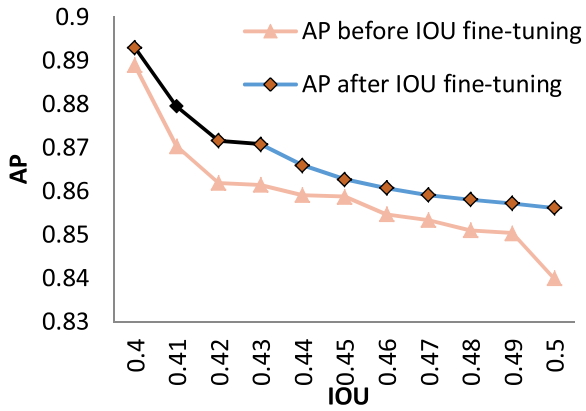

Fig. 15. AP corresponding to different IOU thresholds of our model (GoogLeNet) in VOC07 test.

F. IOU Fine-Tuning Analysis

Insufficient IOU is a main reason to miss a detection. As can be seen in Fig. 15, there is approximately 5% difference in AP between 0.4 IOU and 0.5 IOU in the GoogLeNet version of our model. After our MSI-OHEM is applied to fine-tune the correct detection results with IOU>0.4 in VOC07+12 trainval, the AP in VOC07 test is boosted for nearly 1.1%.

G. Multi-Class or Single-Class Training

Although our vehicle detector is proved to achieve very competitive results and it could be extended to a multi-class detector

TABLE XI
MULTI-CLASS AND SINGLE CLASS DETECTION RESULTS OF FASTER R-CNN AND OUR INITIAL MODEL

	Car	Bus	Vehicle(Car+Bus)
This work(initial model): 20-class version	69.2	68.6	68.9
This work(initial model): single-class version	68.2	69.1	68.8
Faster R-CNN(VGG-16): original 20-class version	84.7	83.1	83.9
Faster R-CNN(VGG-16): single-class version	84.2	83.2	84.0
YOLO original 20-class version	65.0	71.0	68.0
YOLO single-class version	64.2	70.3	65.5

with unnoticeable additional computation cost, one may ask what if other general detectors are trained using only single-class data. That is to say, will the AP of a specific class for a multi-class detector outperform its counterpart in its single-class version? We explore this question by training: (1) a 20-class version of our initial model (GoogLeNet), (2) a single-class version Faster R-CNN (VGG-16), and (3) a single-class version YOLO. These experiments were done in VOC07+12 trainval and VOC07 test while every hyper parameter follows the original setting. However, the training in CNN is non-deterministic in the sense that the resulted AP would be slightly different every time. Therefore, we simply perform training for five times in (1), (2), and (3), respectively.

In Table XI, the vehicle-class detection results in the 1st, the 3rd, and the 5th row are obtained by averaging the AP of car and bus from the 20-class result. The APs in the 2nd, 4th, and 6th rows are the results of single-class training for the car, bus and vehicle classes, respectively.

In the first experiment, as can be seen in the 1st and 2nd rows, there is no significant difference between the 20-class version and single-class version of our initial model in the AP of car and bus classes. Also, the vehicle class detection result of our (single-class) model is almost the same as those for car and bus classes in the 20-class training. As to the second experiment, the resulted AP of car and bus in a single-class Faster R-CNN is quantitatively close to the ones obtained by its original 20-class version. For the YOLO case, its single-class version underperforms its 20-class version in car, bus and vehicle class, respectively.

The above experiments indicate that the detection performance of a class in a multi-class detector is on average slightly better than that in the corresponding single-class detector.

V. CONCLUSION

In this work, a real-time single-stage vehicle detector is presented and it achieved very competitive results in PASCAL VOC, KITTI, iROADS and our Carsim datasets. By introducing the idea of multi-scale feature maps, bounding box priors in the prediction module, the data augmentation strategy, and the MSI-OHEM training framework, we have significantly improved the accuracy of the proposed real-time vehicle detector.

The accuracy improvement is largely due to the proposed multi-stage fine-tuning framework which fine-tunes our detector with hard examples and the ones with slightly-insufficient IOU to be considered as true positives. Our models using MobileNetV2, GoogLeNet, Inception-v2 and ResNet-50 could achieve 51/48/39/21 FPS on Titan X and 33/30/25/13 FPS on GP106 (DrivePX2), respectively. With the help of NVIDIA TensorRT package, which only supports single-stage detector, our detector could be sped up to 64 FPS (MobileNetV2), 58 FPS (GoogLeNet), 48FPS (Inception-v2) and 28 FPS (ResNet-50) on DrivePX2, which fulfills the real-time performance for ADAS or autonomous vehicle.

In the future, we will try to apply powerful 3D engine, such as Unity3D, to generate more realistic vehicle training data to further improve our detector.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their constructive and insightful comments to this paper.

REFERENCES

[1] Z. Sun, G. Bebis, and R. Miller, "On-road vehicle detection: A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 5, pp. 694–711, May 2006.
[2] X. Mao, D. Inoue, S. Kato, and M. Kagami, "Amplitude-modulated laser radar for range and speed measurement in car applications," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 1, pp. 408–413, Mar. 2012.
[3] J. Levinson *et al.*, "Towards fully autonomous driving: Systems and algorithms," in *Proc. IEEE IV Symp.*, Jun. 2011, pp. 163–168.
[4] C. Cortes and V. Vapnik, "Support vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995.
[5] Y. Freund and R. Schapire, "A short introduction to boosting," *J. Jpn. Soc. Artif. Intell.*, vol. 14, no. 5, pp. 771–780, Sep. 1999.
[6] W. Liu, X.Wen, B. Duan, H. Yuan, and N. Wang, "Rear vehicle detection and tracking for lane change assist," in *Proc. IEEE IV Symp.*, Jun. 2007, pp. 252–257.
[7] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. Comput. Vision Pattern Recognit. (CVPR)*, Jun. 2005, pp. 886–893.
[8] T. Machida and T. Naito, "GPU and CPU cooperative accelerated pedestrian and vehicle detection," in *Proc. IEEE Int. Conf. Comput. Vision. Workshops (ICCV Workshops)*, Nov. 2011, pp. 506–513.
[9] Q. Yuan, A. Thangali, V. Ablavsky, and S. Sclaroff, "Learning a family of detectors via multiplicative kernels," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 3, pp. 514–530, Mar. 2011.
[10] N. Blanc, B. Steux, and T. Hinz, "LaRASideCam: A fast and robust vision-based blindspot detection system," in *Proc. IEEE IV Symp.*, Jun. 2007, pp. 480–485.
[11] Y. Zhang, S. Kiselewich, and W. Bauson, "Legendre and Gabor moments for vehicle recognition in forward collision warning," in *Proc. IEEE Intell. Transp. Syst. Conf.*, Sep. 2006, pp. 1185–1190.
[12] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. Comput. Vision Pattern Recognit. (CVPR)*, Dec. 2001, pp. I-511–I-518.
[13] T. Liu, N. Zheng, L. Zhao, and H. Cheng, "Learning based symmetric features selection for vehicle detection," in *Proc. IEEE IV Symp.*, Jun. 2005, pp. 124–129.
[14] A. Khammari, F. Nashashibi, Y. Abramson, and C. Laurgeau, "Vehicle detection combining gradient analysis and AdaBoost classification," in *Proc. IEEE Intell. Transp. Syst. Conf.*, Sep. 2005, pp. 66–71.

[15] I. Kallenbach, R. Schweiger, G. Palm, and O. Lohlein, "Multi-class object detection in vision systems using a hierarchy of cascaded classifiers," in *Proc. IEEE IV Symp.*, Jun. 2006, pp. 383–387.

[16] D. Acunzo, Y. Zhu, B. Xie, and G. Baratoff, "Context-adaptive approach for vehicle detection under varying lighting conditions," in *Proc. IEEE Intell. Transp. Syst. Conf.*, Sep. 2007, pp. 654–660.

[17] W. C. Chang and C. W. Cho, "Real-time side vehicle tracking using parts-based boosting," in *Proc. IEEE Int. Conf. Syst, Man Cybern.*, Oct. 2008, pp. 3370–3375.

[18] S. Sivaraman and M. M. Trivedi, "Real-time vehicle detection using parts at intersections," in *Proc. IEEE Intell. Transp. Syst. Conf.*, Sep. 2012, pp. 1519–1524.

[19] B. F. Lin *et al.*, "Integrating appearance and edge features for sedan vehicle detection in the blind-spot area," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 2, pp. 737–747, Jun. 2012.

[20] X. Zhang, N. Zheng, Y. He, and F. Wang, "Vehicle detection using an extended hidden random field model," in *Proc. IEEE Intell. Transp. Syst. Conf.*, Oct. 2011, pp. 1555–1559.

[21] A. Chavez-Aragon, R. Laganiere, and P. Payeur, "Vision-based detection and labeling of multiple vehicle parts," in *Proc. IEEE Intell. Transp. Syst. Conf.*, Oct. 2011, pp. 1273–1278.

[22] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32. no. 9, pp. 1627–1645, Sep. 2010.

[23] A. Takeuchi, S. Mita, and D. McAllester, "On-road vehicle tracking using deformable object model and particle filter with integrated likelihoods," in *Proc. IEEE IV Symp.*, Jun. 2010, pp. 1014–1021.

[24] H. Niknejad, S. Mita, D. McAllester, and T. Naito, "Vision-based vehicle detection for nighttime with discriminately trained mixture of weighted deformable part models," in *Proc. IEEE Intell. Transp. Syst. Conf.*, Oct. 2011, pp. 1560–1565.

[25] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neu. Inf. Proc. Syst. (NIPS)*, Dec. 2012. pp. 1097–1105.

[26] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, and D. Anguelov, A. Rabinovich, "Going deeper with convolutions," in *Proc. Comput. Vision Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Repr. (ICLR)*, May. 2015, pp. 730–734.

[28] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision." in *Proc. Comput. Vision Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. Comput. Vision Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[30] O. Russakovsky *et al.*, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, Dec. 2015.

[31] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. Comput. Vision Pattern Recognit. (CVPR)*, Jun. 2014, pp. 580–587.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *Proc. Eur. Conf. Comput. Vision (ECCV)*, Sep. 2014, pp. 346–361.

[33] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders, "Selective search for object recognition," *Int. J. Comput. Vision (IJCV)*, vol. 104, no. 2, pp. 154–171, Sep. 2013.

[34] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vision (ICCV)*, Dec. 2015, pp. 1440–1448.

[35] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neu. Inf. Proc. Syst. (NIPS)*, Dec. 2015. pp. 91–99.

[36] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.

[37] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *Int. J. Comput. Vision (IJCV)*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[38] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.Y Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vision (ECCV)*, Oct. 2016, pp. 21–37.

[39] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. Comput. Vision Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6517–6525.

[40] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and $<$ 0.5 MB model size," 2016, *arXiv: 1602.07360*.

[41] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proc. Comput. Vision Pattern Recognit. (CVPR)*, 2018, pp. 6848–6856.

[42] A. G. Howard *et al.*, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv: 1704.04861*.

[43] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. Comput. Vision Pattern Recognit. (CVPR)*, Jun. 2018, pp. 4510–4520.

[44] S. Hossain and D. J. Lee, "Deep learning-based real-time multiple-object detection and tracking from aerial imagery via a flying robot with GPU-based embedded devices," *Sensors*, vol. 19, no. 15, Jan. 2019, Art. no. 3371.

[45] R. J. Wang, X. Li, and C. X. Ling, "Pelee: A real-time object detection system on mobile devices," in *Proc. Adv. Neu. Inf. Proc. Syst. (NIPS)*, Dec. 2018, pp. 1963–1972.

[46] M. Orsic, I. Kreso, P. Bevandic, and S. Segvic, "In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images," in *Proc. Comput. Vision Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12607–12616.

[47] NVIDIA, TensorRT. [Online]. Available: https://developer.nvidia.com/tensorrt

[48] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," in *Proc. Int. Conf. Learn. Repr. (ICLR)*, Apr. 2014, pp. 1–16.

[49] Z. Cai, Q. Fan, R. S Feris, and N. Vasconcelos, "A unified multi-scale deep convolutional neural network for fast object detection," in *Proc. Eur. Conf. Comput. Vision (ECCV)*, Oct. 2016, pp. 354–370.

[50] T. Kong, A. Yao, Y. Chen, and F. Sun, "HyperNet: Towards accurate region proposal generation and joint object detection," in *Proc. Comput. Vision Pattern Recognit. (CVPR)*, Jun. 2016, pp. 845–853.

[51] H. Zhang, K. Wang, Y. Tian, C. Gou, and F-Y Wang, "MFR-CNN: Incorporating multi-scale features and global information for traffic object detection," *IEEE Trans. Veh. Technol.*, vol. 67. no. 9, pp. 8019–8030, Jun. 2018.

[52] A. Shrivastava, A. Gupta, and R. Girshick, "Training region based object detectors with online hard example," in *Proc. Comput. Vision Pattern Recognit. (CVPR)*, Jun. 2016, pp. 761–769.

[53] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis, "Improving object detection with one line of code," in *Proc. IEEE Int. Conf. Comput. Vision (ICCV)*, Oct. 2017, pp. 5562–5570.

[54] S. Sivaraman and M.M Trivedi, "A general active-learning framework for on-road vehicle recognition and tracking," *IEEE Trans. Intell. Transp.*, vol. 11. no. 2, pp. 267–276, Feb. 2010.

[55] Y. Zhou, L. Liu, L. Shao, and M. Mellor, "DAVE: A unified framework for fast vehicle detection and annotation," in *Proc. Eur. Conf. Comput. Vision (ECCV)*, Oct. 2016, pp. 278–293.

[56] A. Geiger, P. Lenz, and R Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *Proc. Comput. Vision Pattern Recognit. (CVPR)*, Jun. 2012, pp. 3354–3361.

[57] Mechanical Simulation Corporation. CarSim mechanical simulation, https://www.carsim.com/products/carsim/

[58] M. Rezaei and M. Terauchi, "Vehicle detection based on multi-feature clues and dempster-shafer fusion theory," in *Proc. Pacific-Rim Symp. Image Video Technol. (PSIVT'13)*, Nov. 2013, pp. 60–72.

[59] S. loffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Jun. 2015, pp. 448–456.

[60] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-Level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vision (ICCV)*, Dec. 2015, pp. 1026–1034.

[61] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. ACM Int, Conf. Multimedia*, Nov 2014, pp. 675–678.

[62] Y. Xiang, W. Choi, Y. Lin, and S. Savarese, "Data-driven 3d voxel patterns for object category recognition," in *Proc. Comput. Vision Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1903–1911.

[63] Y. Xiang, W. Choi, Y. Lin, and S. Savarese, "Subcategory aware convolutional neural networks for object proposals and detection," in *Proc. Win. Conf. Appl. Comput. Vision (WACV)*, Mar. 2017. pp. 924–933.

Che-Tsung Lin received the B.S. degree in mechanical engineering from the National Taiwan University of Science and Technology, Taipei, Taiwan, in 2003, the M.S. degree from the Institute of Applied Mechanics in National Taiwan University, Taipei, Taiwan, in 2005. He is currently working toward the Ph.D. degree with the Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan. He was a Visiting Scholar with the Computer Science Department, University of California, Santa Barbara, CA, USA, in 2013. He has been working with Safety Sensing and Control Department, Intelligent Vehicle Division, Mechanical and Mechatronics System Research lab, Industrial Technology Research Institute, Hsinchu, Taiwan, since 2006, where he is currently a Researcher in the field of intelligent transportation system, intelligent vehicle, and advanced driver assistance system. His research interests include computer vision, machine learning, deep learning and their application in on-road object detection.

Shu-Ping Chen received the B.S. degree from the College of Electronic Engineering and Computer Science and the M.S degree in computer science from National Tsing Hua University, Hsinchu, Taiwan, in 2016 and 2018, respectively. His research interests include object detection and deep learning.

Patrisia Sherryl Santoso received the B.E. degree in industrial engineering from the Sepuluh Nopember Institute of Technology, Indonesia, in 2014, and the M.S. degree in EECS from National Chiao Tung University, Hsinchu, Taiwan, in 2016. She is currently working as an Associate Researcher with Safety Sensing and Control Department, Intelligent Vehicle Division, Mechanical and Mechatronics System Research lab, Industrial Technology Research Institute, Hsinchu, Taiwan, since 2016. Her research topics range from pedestrian detection to road mark detection using various mixed technologies of image processing, computer vision, machine learning and deep learning.

Hung-Jin Lin received the B.S. and M.S. degrees in computer science from National Tsing Hua University, Hsinchu, Taiwan, in 2016 and 2018, respectively. His research interests include computer vision include 3D reconstruction, pose estimation and registration in scene understanding.

Shang-Hong Lai (M'95) received the Ph.D. degree in electrical and computer engineering from the University of Florida, Gainesville, in 1995. Then, he joined Siemens Corporate Research in Princeton, New Jersey, as a member of technical staff. Since 1999, he has been a Faculty Member with the Department of Computer Science, National Tsing Hua University (NTHU), Taiwan. He is currently a Professor with the same department. Since 2018, he has been on leave from NTHU to join Microsoft AI R&D Center in Taipei as a Principal Researcher. Dr. Lai's research interests include computer vision, image processing and machine learning. He has authored more than 200 papers published in the related international journals and conferences. Dr. Lai has been a member of program committee of several international conferences, including CVPR, ICCV, ECCV, ACCV, ICPR, PSIVT and ICME. Furthermore, he has been an Associate Editor or Guest Editor for several international journals, including *Pattern Recognition, Journal of Signal Processing Systems, Journal of Visual Communication and Image Representation, IPSJ Transactions on Computer Vision and Applications*, etc.