Fast Vehicle Detector for Autonomous Driving

Che-Tsung Lin[1,2], Patrisia Sherryl Santoso[2], Shu-Ping Chen[1], Hung-Jin Lin[1], Shang-Hong Lai[1]
[1]Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan
[2]Intelligent Mobility Division, Mechanical and Systems Research Laboratories, Industrial Technology Research Institute, Hsinchu, Taiwan
alexofntu@gmail.com, sherrylsantoso@itri.org.tw, scarletclaw24@gmail.com, salas@gapp.nthu.edu.tw, lai@cs.nthu.edu.tw

Abstract

This paper presents a fast vehicle detector which can be deployed on NVIDIA DrivePX2 under real-time constraints. The network predicts bounding boxes with different aspect ratio and scale priors from the specifically-designed prediction module given concatenated multi-scale feature map. A new data augmentation strategy is proposed to systematically generate a lot of vehicle training images whose appearance is randomly truncated so our detector could detect occluded vehicles better. Besides, we propose a non-region-based online hard example mining framework which performs fine-tuning by picking (1) hard examples and (2) detection results with insufficient IOU. Compared to other classical object detectors, this work 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 achieves 85.32 AP and runs at 48 FPS and 30 FPS on NVIDIA Titan X & GP106 (DrivePX2), respectively.

1. Introduction

Vehicle detection is a fundamental problem required for both Advanced Driver Assistance Systems (ADAS) and autonomous vehicle. One expects that vehicles could be detected as accurately as possible by an ADAS because the function of such a system is to enhance driving safety especially for the scenario that the host vehicle and the preceding vehicle is very close. That is to say, either false-positive or false-negative should be eliminated under critical conditions. Before the era of deep learning, traditional vehicle detection methods were mostly developed under a Hypothesis Generation (HG) + Hypothesis Verification (HV) framework [1] that the former is to generate region proposals and the latter applies a pair of feature extractor and classifier to eliminate false positives. To achieve high detection performance, one has to employ a star-structured architecture consisting of root and parts filters with associated deformation models for object detection. DPM [2] can successfully handle deformable object detection even when the target is partially occluded. However, it leads to heavy 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 [3], GoogLeNet [4], VGG16 [5] and the powerful ResNets [6].

Object detection is a challenging task and its recent advances are driven by the success of region-based convolutional neural networks (RCNNs) [7]. Although RCNNs were computationally expensive originally, their computational cost has been drastically reduced by sharing convolutions, inspired by [8], across proposals generated by Selective Search [9]. This idea brought up Fast R-CNN [10] which achieves near real-time efficiency using very deep networks-VGG16, when ignoring the time spent on region proposal generation. Proposals were the test-time computational bottleneck in the object detection systems and its computation was significantly reduced by region proposal network (RPN) which was proposed in Faster R-CNN [11]. Although Faster R-CNN achieves 4-5 FPS, it is still far from the requirement of real-time object detection. i.e., 30 FPS. Recently, YOLO [12] made a break-through by realizing an end-to-end framework. It reframes object detection as a single regression problem. The last layer is simply the probability of an object, the class it belongs to and the location it is in the image. This work achieves 45 FPS in PASCAL VOC 2007 [13] test dataset with mAP of 63.4. Its sped-up version, Fast YOLO, further reaches 155 FPS with mAP of 52.7% on the same dataset. However, poor localization precision was also reported according to the analysis in their work.

In order to better accommodate objects of different sizes, Faster R-CNN reported that using 9 anchor boxes brings the benefit of mAP boost while the size and aspect ratio of 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 [14] and YOLO that operate on a single scale feature map, SSD [15] expects object at different size to be detected from lower to higher layers. MS-CNN [16] carefully selects feature maps from early layers to higher

ones for generating region proposals from different scales. Hypernet [17] aggregates hierarchical feature maps using pooling and deconvolution layers to form hyper feature maps where RPN generates region proposal from.

Hard negative mining was a very popular technique for enhancing the object detection accuracy. However, it needs to freeze the model for selecting the hard negative results and then put them into negative training results. In Fast R-CNN, the fg-bg (foreground-background) ratio is 1/3. OHEM [18] claimed that changing this ratio in training Fast R-CNN would decrease mAP so they proposed to train only hard examples without keeping this heuristic parameter. Only those region proposals with higher loss would contribute to the back propagation (BP) and the mAP is significantly boosted for this reason. YOLO also suffers from such inconvenience because most images contain only a few objects so they specifically set the parameters for balancing fg-bg loss. The imbalance makes YOLO sensitive to the weighting parameters in the loss function.

Most on-road vehicle datasets are captured with limited viewing angles or at fixed distances. LISA 2010 dataset [19] is entirely composed of rear-viewed front vehicles. Urban Traffic Surveillance (UTS) dataset [20] provides vehicle images captured by surveillance systems. KITTI dataset [21] 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 than previous ones. However, 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 [22], could easily generate a massive amount of data with auto-generated ground truth (GT). In this paper, we also investigate the possibility of using Carsim in vehicle detector development.

Inspired by the advantages and drawbacks of previous works, in this paper, we propose a fast vehicle detector which two-dimensionally relates the outputs of a CNN to a given image. The improvements include (1) an efficient basenet, GoogLeNet, combined with a multi-scale-based prediction module which carefully merges spatially-rich information from low-level features with high-level semantic information encoded in upper layers, (2) a heavy data augmentation strategy specifically designed for vehicle detection and (3) a Non-region-based OHEM (NOHEM) strategy which significantly boosts the AP. Comparisons with the state-of-the-art methods on these benchmarks demonstrate the advantages of the proposed method for vehicle detection.

2. Fast Vehicle Detector

This section describes the proposed framework and the associated training methodology. As can be seen in Figure 1, if an image is fed into our CNN with a prediction module on top, the vehicle inside is expected to be detected. Our prediction module is based on the concatenated multi-scale feature maps together with 2 more convolution layers on top. As shown in Figure 2, the last layer is the output and it is spatially interpreted as a fixed number of grids responsible for detecting objects whose centers of the object windows are located inside. Our design has the following key features:

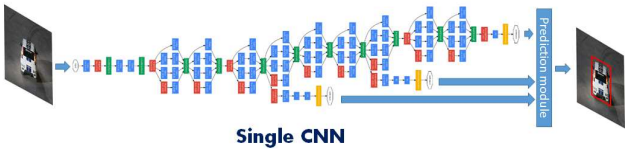

Figure 1: Our CNN model.

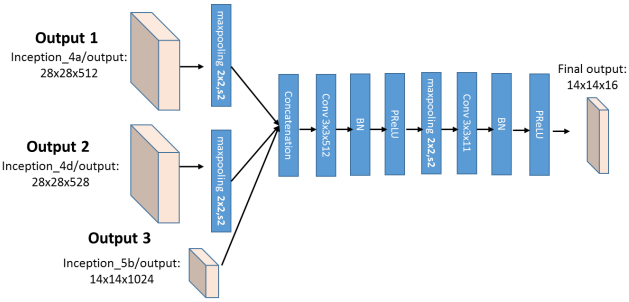

Figure 2: Our prediction module.

Efficient Basenet: Most object detectors applied pre-trained CNNs. SSD is with VGG16. YOLO proposed a CNN whose performance is close to GoogLeNet. Faster R-CNN is with ZF-net [23] for ablation studies and achieved better performance using VGG16 and ResNet-101. Generally speaking, using better basenet is beneficial to the subsequent detection task. VGG16 and ResNet-101 achieve 0.912 & 0.937 for single crop, top-5 accuracy at 22 FPS & 19 FPS respectively on Titan X for 224×224 images. [1] However, vehicle detection is mostly expected to run in real-time. Although there are alternative choices, such as ResNets with fewer layers, we applied GoogLeNet because of its simplicity and fast inference speed (56 FPS with our two extra layers and 65 FPS without on Titan X for 448×448 images). Since we use a 448×448 image as our input, we pre-trained the whole GoogLeNet on 1000-way ImageNet classification [24] with 448×448 images from scratch and finally reached top-5 accuracy of 0.895 and top-1 accuracy of 0.6995 at 2.4 million iterations. Besides, we also tried to fine-tune the original GoogLeNet with 448×448 images for 0.24 million iterations. The fine-tuning followed the recommended GoogLeNet pre-training parameters with 0.9 momentum, 0.0002 weight decay but the learning rate is lowered to

[1] https://github.com/jcjohnson/cnn-benchmarks

0.0001.This attempt reached top-5 accuracy at 0.9109 and top-1 accuracy with 0.715. The quantitative results had proved that fine-tuning with higher resolution training images is better than training from scratch. Most importantly, the improved accuracy did boost the subsequent detection.

Data Augmentation: This skill is crucial to train either a classifier or a detector. Fast and Faster R-CNN horizontally flip the input image with probability of 0.5. YOLO introduces random scaling and translation of up to 20% of the original image size and adjusts the exposure and saturation of the image by up to a factor of 1.5 in the HSV color space. SSD's strategy is more diverse in that they randomly sample a patch that overlaps the object with certain size and horizontally flipping. Our strategy is similar in that we want to enhance the detection capability even when it is heavily occluded so we also resize the vehicles in the training images with 0.1 to 0.9 of the original size if their area is above a predetermined threshold and this image is horizontally flipped with 0.5 probabilities. And there is also a probability of 0.5 for this vehicle to be truncated by a new window that only contains at least 25% of its appearance visible. We found this strategy very useful in dealing with vehicles occluded by each other. While only a few of them are partially seen, the ones that are entirely seen were chosen to be randomly resized, rotated, horizontally-flipped, recolored, cropped or put in the boundary to make them truncated to a random extent. Our detector is therefore doing better in learning the essence of vehicle's appearance. Finally, in order to improve the detector capability in detecting vehicles with fuzzy appearance, we randomly blur the training image with either mean filter, median filter or, Gaussian blur.

Multi-Scale Feature Maps and Bounding Box Priors: Objects can appear at a variety of scales. A single receptive field cannot match this variability. MS-CNN selected four different feature maps to generate region proposals. Hypernet directly generates region proposals from a hyper feature map concatenated by three feature maps coming from lower to higher layers. We approach the idea of multi-scale features by concatenating feature maps from inception_4a, inception_4d and inception_5b in GooLeNet. Then, two more additional convolutional layers are stacked on top. Besides, each layer is followed by Batch Normalization [25] and PReLU[26].

Faster R-CNN predicts bounding boxes using hand-picked priors. The input image of single-shot detector like YOLO is normalized to 448×448 so it is equivalent to a single anchor box. Most detection errors by YOLO as described in their work could be attributed to localization error. SSD did not suffer from this problem comparatively because they use multi-scale feature maps together with three default boxes to internalize the idea of multi-anchor box. However, instead of hand-picked priors, we perform k-means algorithm in exploring better priors. In our study, we found that using 3 priors (318x313, 212x176, 63x56) reached best performance in PASCAL VOC07 test and the overhead to add more priors is not significant because only several more feature layers are necessary to be stacked on the top layer.

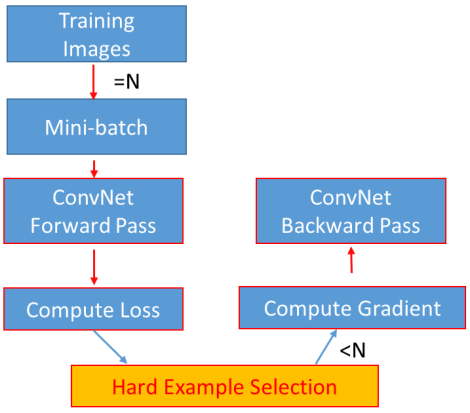

Figure 3: Flowchart of Non-region-based Hard Example Mining- initial version.

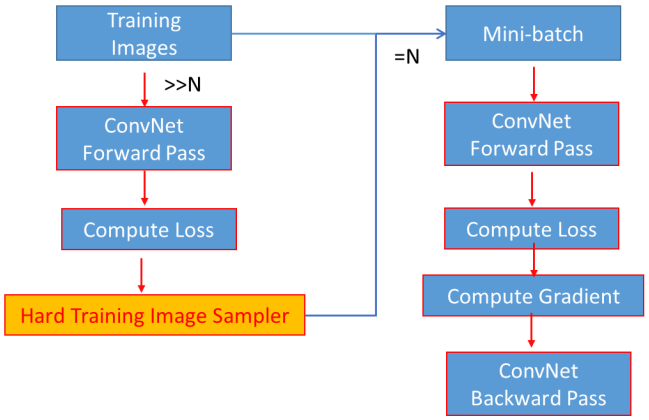

Figure 4: Flowchart of Non-region-based Hard Example Mining- final version.

Non-region-based Online Hard Example Mining: Hard negative mining is a classical training skill in training object detector. The basic procedure is to collect initial training data, freeze this model, run detection on test 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. With the advent of GPU, it is inefficient to freeze the model every time we found negative results. OHEM introduced a framework for region-based detectors, such as Fast R-CNN, to do BP for foreground or background objects with higher loss.

However, this framework could only be applied for region-based detectors.

A naïve way to implement this idea in non-region-based detectors is shown in Figure 3. In a given mini-batch of N images, the loss sum in every training image is sorted. Then, some training images with lower loss are nullified to do BP. The cost of such BP is nearly the same as before and no additional mechanism should be re-designed. 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 in a single mini-batch. In fact, after our detector is trained using standard BP, global hard examples are distributed in sparse training images based on the current snapshot and the training images possessing higher loss might not be quantitatively hard enough.

We proposed Non-region-based OHEM (NOHEM) to train a CNN with images possessing loss higher than a threshold instead of higher loss. Our framework is shown in Figure 4. As can be seen, the left module would keep performing inference until the number of hard images possessing 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 ones, equal to N, with top losses. However, it is quite time consuming because we have to perform inference for every training image before doing a single BP. Last but not least, this framework is only designed for fine-tuning after standard training is completed. NOHEM is carried out for two times. The 1st one is performed after standard training is finished and the loss function is the same. The 2nd one uses the loss function designed for IOU boosting.

Non-Maximal Suppression: Non-maximum suppression (NMS) is usually a post-processing step 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 and all others with a significant overlap (using a pre-determined threshold) are suppressed. Soft-NMS, an algorithm [27] which decays the detection scores of all other objects as a continuous function of their overlap with the detection box that possesses maximum score, is applied to our detector. Using the linear version of Soft-NMS is proved beneficial in our experiments.

Loss function of Training and Fine-tuning: Our training includes 3 stages with different loss functions. The 1st stage applies standard training using object loss and non-object loss as (1) and (2). The 2nd one adopts the same loss functions, but the whole CNN is fine-tuned using the proposed NOHEM. The 3rd one fine-tunes the network using (3) which is designated for IOU boosting and still applies our NOHEM. Since the latter two only fine-tune the CNN, their learning rate is relatively low.

At the beginning, the class, objectness and the bounding boxes are learned altogether. For an object whose center of object window locating in grid ij, i.e., $G_{ij} \in (Object)$, its loss function is

$$\sum_{n\in\{prior_1,\ldots,prior_k\}}\Big(\sum_{m\in\{w(n),h(n),x,y\}}\alpha_m(\mathrm{G}_{ij}^{m}-\hat{G}_{ij}^{m})^2+\alpha_{object}(\mathrm{G}_{ij}^{o}-1)^2\Big)+\alpha_{class}(\mathrm{G}_{ij}^{c}-1)^2 \quad (1)$$

where G_{ij}^{o} denotes the objectness score; G_{ij}^{c} the class score; (G_{ij}^{x}, G_{ij}^{y}) the coordinate of object. $\mathrm{G}_{\mathrm{ij}}^{w(prior_k)}$ and $\mathrm{G}_{\mathrm{ij}}^{h(prior_k)}$ stand for the regressed width and height in terms of the kth prior estimated by k-means clustering based on the width and height statistics. $\hat{G}_{ij}^{x}$, $\hat{G}_{ij}^{y}, \hat{G}_{ij}^{w}$ and $\hat{G}_{ij}^{h}$ represent the location and size GT of the object locating at grid ij. For $G_{ij} \notin (Object)$, i.e., no object whose center of object window locates in grid ij, its loss function is

$$\sum \alpha_{non-object}\left(G_{ij}^{o}\right)^2+\alpha_{class}\left(G_{ij}^{c}\right)^2. \quad (2)$$

There are 2 terms for non-object grid. The first one is the objectness score and the 2nd is the class score. Although in this work, our goal is only to detect 1 type of object-vehicle, we still keep class score that typical multi-class object detectors possess because determining if there is an object locating in a grid by considering the class score and objectness score rather than the latter only would slightly boost the AP.

The 2nd loss function is designed to boost the IOU using hard examples because detection results with insufficient IOU would be treated as the localization error. 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 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 a grid $G_{ij} \in (Object,\ IOU(G_{ij}) > \beta,\ G_{ij}^{o} > \gamma)$, i.e., object belonging to G_{ij} with IOU higher than β(0.4 for PASCAL VOC, and 0.6 for KITTI car) and objectness higher than γ (0.6), its loss function is

$$\sum_{n\in\{prior_1,\ldots,prior_k\}}\Big(\sum_{m\in\{w(n),h(n),x,y\}}\alpha_n(G_{ij}^{m}-\hat{G}_{ij}^{m})^2\Big). \quad (3)$$

Since this is a fine-tuning procedure, the learning rate is set to 0.0001 and the parameter α_m is also set to 1/100 of its original value in training by using eq. (1).

3. Experimental Results

We evaluate our detector on PASCAL VOC2007 dataset, KITTI detection benchmark, and Carsim-generated dataset

which is self-collected and is only used for testing. Our training all follows the same setting. Each grid in 14×14 feature maps contains 3 objects with corresponding 3 objectness scores and 1 class score. Therefore, there will be 16 values describing objects whose center of object window is located in a grid as shown in Figure 2. The starting learning rate is 10^{-3} and it is divided by 10 for every 16k iterations and the total iterations are 48k. The weight decay and momentum are set to 0.0005 and 0.9, respectively.

3.1. PASCAL VOC2007

In VOC 2007 trainval and VOC2012 trainval, there are totally 16551 training images. In VOC2007 test, there are 4 952 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 appearances of bus are essentially similar to those 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's appearance. Therefore, we defined a new class called vehicle which is composed of car & bus classes for both training (1420) and test images (825).

Detection Frameworks	Train	FPS
Fast R-CNN	2007+2012	0.5
Faster R-CNN VGG-16	2007+2012	7
Faster R-CNN ResNet	2007+2012	5
YOLO	2007+2012	45
Fast YOLO	2007+2012	155
SSD300	2007+2012	46
SSD500	2007+2012	19
This work	2007+2012	48

Table 1: Detection frameworks on PASCAL VOC 2007.

	Bus	Car	Vehicle (Car+Bus)
Fast R-CNN	81.6	78.6	80.1
Faster R-CNN VGG-16	83.1	84.7	83.9
Faster R-CNN ResNet	85.1	85.3	85.15
SSD 300	81.1	80.8	80.95
SSD 500	84.9	85.6	85.25
YOLO	71	65	68
Fast YOLO	60	67	63.5
This work			85.32

Table 2: PASCAL VOC 2007 test detection results.

In Table 1, we compare our work to other famous detectors in terms of training data and FPS (assessed on Maxwell Titan X). As can be seen, our vehicle detector is only slower than Fast YOLO. As shown in Table 2, our model achieves state-of-the-art AP in the newly-defined vehicle class in VOC07 test. Although other detectors are designed for multi-class detection, our work could be easily extended to detect 20 classes, the original number of class in PASCAL VOC, because only 19 more feature maps are required in the last layer. In our experiment, the processing time would only increase a few milliseconds for an image, so such a comparison is fair. Figure 5 visualizes the comparison of this work and other detectors in terms of AP and FPS. Last but not least, our detector could run 48 FPS on Titan X and 30FPS on GP106 (DrivePX2).

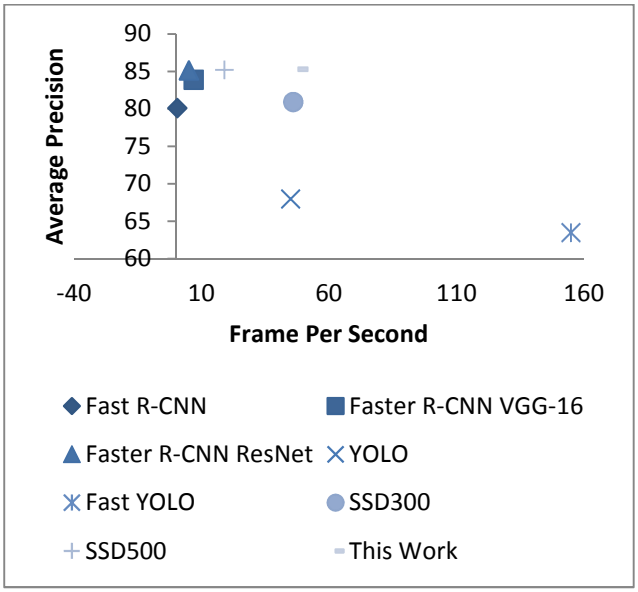

Figure 5: AP and FPS on VOC07 test dataset.

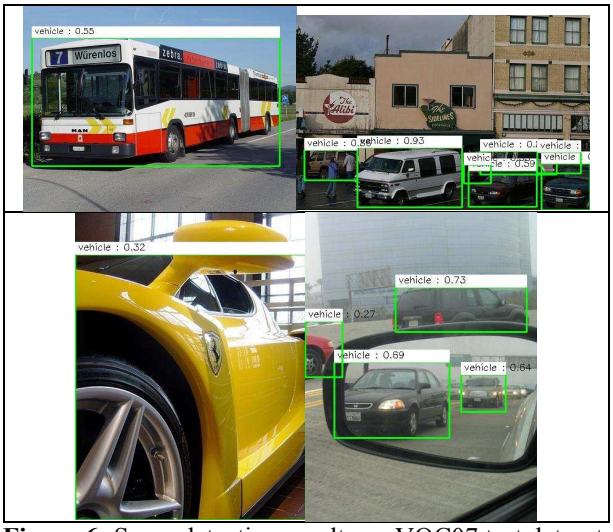

Figure 6: Some detection results on VOC07 test dataset.

Figure 6 demonstrates some typical detection results including non-, slightly-, heavily-occluded or truncated vehicles.

3.2. KITTI Val Set Evaluation

In this section, we evaluate the proposed approach 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 splits from the training set to validate our method. To compare our approach with other state-of-the-art methods, we use train/val splits as indicated by [28, 29]. We follow every setting that leads us to the best results in PASCAL VOC. As can be seen in Table 3, since the required IOU is higher than PASCAL VOC, our results seems not better. However, if we follow the same IOU required by PASCAL VOC, our results are satisfactory in that most of the closest vehicles in path could be successfully detected.

		AP(IOU=0.7)		
Method	Time	Easy	Moderate	Hard
3DVP	40s	80.48	68.05	57.20
Faster-RCNN	2s	82.91	77.83	66.25
Ours	0.02	62.41	52.41	47.15
		AP(IOU=0.5)		
Ours	0.02	83.75	78.48	74.92

Table 3: Vehicle detection results on KITTI val sets.

	YOLO	Faster R-CNN VGG-16	This work
Scenario 1 (12 vehicles)	0.676	0.999	0.992
Scenario 2 (3 vehicles)	0.873	1	1
Scenario 3 (3 vehicles)	0.955	1	1
Scenario 4 (3 vehicles)	0.685	0.963	0.999
Scenario 5 (3 vehicles)	0.673	0.982	0.999

Table 4: AP comparison of YOLO, Faster R-CNN and our work in 5 scenarios.

3.3. 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 and no manual labeling is necessary.

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 Figure 7, there are totally 12 different vehicle types including sedan, truck, SUV, etc. And, none of them is or looks like bus because we only want to quantitatively evaluate our detector and other famous detector's "car" detection capability. In scenario-1, each vehicle maneuvers with the same pre-set trajectory and the frames of each video is the same (1052 frames). Totally, there are 12624 frames. The purpose of this experiment is to assess a vehicle detector's capability in detecting all kinds of vehicles in a controlled environment. Therefore, the background of each video is the same. For scenario-2 to -5, vehicle-1, -6 and -8 in Figure 7 are selected for conducting the following experiments. Each setting, in terms of a fixed distance and tilt angle pairs, ranging from (4.4m, 0°), (8m, 0°), (5m, 25°), (8m, 20°), creates 360-degree view of the selected vehicle for 241 frames. So, there are 723 frames for each scenario other than the 1st one. We compare YOLO, Faster R-CNN with our work on all of the CarSim-generated data.

Figure 7: 12 vehicle types in Carsim from top to bottom and left to right.

As can be seen in Table 4, in scenario-1, YOLO underperforms others when vehicles are relatively small in the maneuvering. In the rest scenarios, YOLO is slightly better because vehicle size is bigger. However, Faster R-CNN (VGG-16) and our method achieve very promising result for almost all scenarios. It is worth mentioning that the only imperfect detection results of Faster R-CNN is due to false negative results instead of detecting a vehicle as a bus but this is possible because we directly use the downloaded version so originally it is a 20-class detector.

4. Model Analysis

In order to quantitatively improve our detector, we have carried out several controlled experiments to examine how each component affects the final performance. For all of the following experiments, the training data are all VOC 07+12 with the same training iteration and learning rate.

4.1. Basenet Comparison

ResNet family is highly possible to boost the detection capability of a detector because ResNet-101, ResNet-152 has achieved promising Top-1 and Top-5 accuracy. Using them as basenet is beneficial to directly boost the detection capability. However, in consideration of the possibility to deploy our detector on an embedded system, only ResNet-18 and ResNet-32 are compared in our experiments. ResNet-18 and ResNet-32 obtain 0.861 & 0.881 for single crop, top-5 accuracy on Titan X for 224×224 images. Both

	Car	Bus	Vehicle (Car+Bus)									
448×448 Fine-tuning				√	√	√	√	√	√	√	√	√
Batch normalization + PReLU					√	√	√	√	√	√	√	√
14×14 prediction						√	√	√	√	√	√	√
Data augmentation							√	√	√	√	√	√
Multi-scale feature								√	√	√	√	√
Bounding box priors									√	√	√	√
NOHEM										√	√	√
NOHEM (IOU)											√	√
Soft-NMS												√
AP	68.2	69.1	68.8	70.1	74.9	77.5	81.2	81.8	82.4	83.9	84.8	85.3

Table 6 Each design decisions that lead to better performance in terms of AP.

of them slightly underperform GoogLeNet. In order to unbiasedly analyze the capability of basenets, we simply add two more convolution layers on top of each of them. The output size is all with 448×448 input images and 7×7 output grid setting, as the basic version of our work in the ablation study.

Basenet	FPS(TitanX)	FPS (GP106)	AP
GoogLeNet	56	36	68.8
ResNet-18	58	33	68.1
ResNet-32	34	15	69.6

Table 5: Vehicle detection results for our initial version.

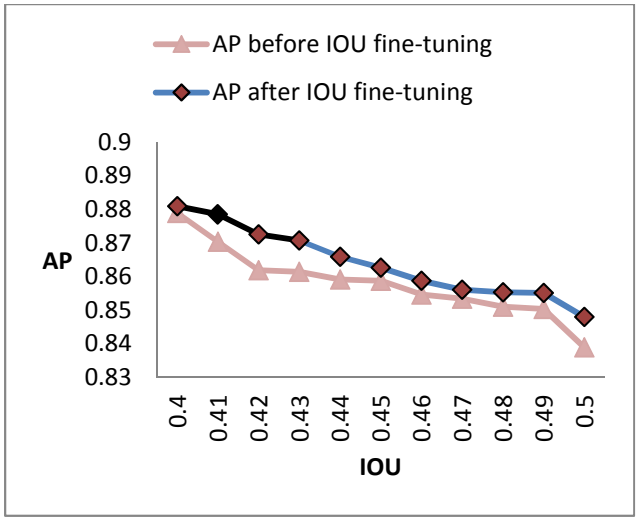

Figure 8: AP corresponding to different IOU thresholds.

As shown in Table 5, the FPS of ResNet-18 and GoogLeNet are not significantly different but the AP of the latter is slightly higher. ResNet-32 is significantly slower but it could achieve better result. Since real-time applicability is the major concern to ADAS and autonomous vehicle, we choose computationally-economical GoogLeNet as our basenet.

4.2. Ablation study

We made some improvements based on the original version of our work to achieve the final result. A summary result could be found in Table 6.

4.3. IOU Fine-tuning Analysis

Insufficient IOU is a major reason to consider a detection result as TP. As can be seen in Figure 8, there is an approximate AP 4% difference between 0.4 IOU and 0.5 IOU. After our NOHEM is applied to fine-tune the correct detection results with IOU>0.4 in PASCAL, the AP is boosted for nearly 0.5%.

5. Conclusion

We developed a fast vehicle detector which achieves very competitive results in vehicle detection. Major improvements have been made by introducing the idea of multi-scale feature maps, size priors and most importantly the NOHEM. We found that the AP could still be boosted by only training with hard examples after standard training stops improving AP. Besides, insufficient IOU for objects to be counted as TP is also alleviated by fine-tuning only the weightings describing the location and the size of the detected objects. Our method achieves 48 FPS on Titan X and 30 FPS on GP106 (DrivePX2) and it is therefore able to function in real-time for ADAS or autonomous vehicle.

In the future, we will try to use more realistic vehicle training data generated by car simulators using more powerful 3D engine, such as Unity3D, to learn the essence of vehicle appearances.

References

[1] Z. Sun, G. Bebis, and R. Miller, "On-road vehicle detection: a review" IEEE Transactions On Pattern Analysis And Machine Intelligence, vol. 28, no.5, pp. 694-711, May 2006.

[2] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32. no.9, pp.1627–1645, Sep 2010.

[3] A. Krizhevsky, I. Sutskever and G. Hinton, "ImageNet classification with deep convolutional neural networks," in Advances in Neural Information Processing Systems (NIPS), 2012.

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed and D. Anguelov, A. Rabinovich, "Going deeper with convolutions," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in International Conference on Learning Representations (ICLR), 2015.

[6] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in European Conference on Computer Vision (ECCV), 2014.

[9] J. Uijlings, K. van de Sande, T. Gevers and A. Smeulders, "Selective search for object recognition," International Journal of Computer Vision (IJCV), 2013.

[10] R. Girshick, "Fast R-CNN," in International Conference on Computer Vision (ICCV), 2015.

[11] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in Advances in Neural Information Processing Systems (NIPS), 2015.

[12] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You only look once: Unified, real-time object detection," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[13] M. Everingham, L. Van Gool, C. K. Williams, J. Winn and A. Zisserman, "The pascal visual object classes (voc) challenge," International Journal of Computer Vision (IJCV), 2010.

[14] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," in International Conference on Learning Representations (ICLR), 2014

[15] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.Y Fu and A. C. Berg, "SSD: Single shot multibox detector," in European Conference on Computer Vision (ECCV), 2016

[16] Z. Cai, Q. Fan, R.S Feris and N. Vasconcelos, "A unified multi-scale deep convolutional neural network for fast object detection," in European Conference on Computer Vision (ECCV), 2016.

[17] T. Kong, A. Yao, Y. Chen and F. Sun, "HyperNet: Towards accurate region proposal generation and joint object detection," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[18] A. Shrivastava, A. Gupta and R. Girshick, "Training region based object detectors with online hard example," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[19] S. Sivaraman and M.M Trivedi, "A general active-learning framework for on-road vehicle recognition and tracking," IEEE Transactions on Intelligent Transport System, vol. 11. no. 2, pp. 267-276, 2010.

[20] Y. Zhou, L. Liu, L. Shao and M. Mellor, "DAVE: A unified framework for fast vehicle detection and annotation," in European Conference on Computer Vision (ECCV), 2016.

[21] A. Geiger, P. Lenz and R Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012

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

[23] M.D Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in European Conference on Computer Vision (ECCV), 2014.

[24] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg and L. Fei-Fei, "ImageNet large scale visual recognition challenge," International Journal of Computer Vision (IJCV), 2015.

[25] S. loffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in International Conference on Machine Learning (ICML), 2013.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: surpassing human-Level performance on ImageNet classification," in International Conference on Computer Vision (ICCV), 2015.

[27] N. Bodla, B. Singh, R. Chellappa, and L.S. Davis, "Improving object detection with one line of code," in International Conference on Computer Vision (ICCV), 2017.

[28] Y. Xiang, W. Choi, Y. Lin, and S. Savarese, "Data-driven 3d voxel patterns for object category recognition," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[29] Y. Xiang, W. Choi, Y. Lin, and S. Savarese, "Subcategory aware convolutional neural networks for object proposals and detection," in IEEE Winter Conference on Applications of Computer Vision (WACV), 2017.