Deep Neural Network Based Vehicle and Pedestrian Detection for Autonomous Driving: A Survey

Long Chen, *Senior Member, IEEE*, Shaobo Lin, Xiankai Lu, *Member, IEEE*, Dongpu Cao, *Member, IEEE*, Hangbin Wu, Chi Guo, Chun Liu, *Member, IEEE*, and Fei-Yue Wang, *Fellow, IEEE*

***Abstract*— Vehicle and pedestrian detection is one of the critical tasks in autonomous driving. Since heterogeneous techniques have been proposed, the selection of a detection system with an appropriate balance among detection accuracy, speed and memory consumption for a specific task has become very challenging. To deal with this issue and to provide guidance for model selection, this paper analyzes several mainstream object detection architectures, including Faster R-CNN, R-FCN, and SSD, along with several typical feature extractors, such as ResNet50, ResNet101, MobileNet_V1, MobileNet_V2, Inception_V2 and Inception_ResNet_V2. By conducting extensive experiments using the KITTI benchmark, which is a commonly used street dataset, we demonstrate that Faster R-CNN ResNet50 obtains the best average precision (AP) (58%) for vehicle and pedestrian detection, with a speed of 8.6 FPS. Faster R-CNN Inception_V2 performs best for detecting cars and detecting pedestrians respectively (74.5% and 47.3%). ResNet101 consumes the highest memory (9907 MB) and has the largest number of parameters (64.42 millions), and Inception_ResNet_V2 is the slowest model (3.05 FPS). SSD MobileNet_V2 is the fastest model (70 FPS), and SSD MobileNet_V1 is the lightest model in terms of memory usage (875 MB), both of which are suitable for applications on mobile and embedded devices.**

***Index Terms*— Deep neural networks, autonomous driving, vehicle detection, pedestrian detection, survey.**

I. Introduction

CURRENTLY, vehicle and pedestrian detection technologies are commonly employed in various applications for public safety purposes. For instance, vehicle detection is the premise of the planning and control of self-driving vehicles [1]–[3]. Only by coordinating the positional relationship with the surrounding vehicles, can the safety of the ego-vehicle be guaranteed [4]. Pedestrian detection is used to help protect pedestrians by avoiding probable collisions with them. Another essential application of pedestrian detection is identification and tracking of criminals in crowds. Because identifying vehicles and pedestrians is the key to self-driving cars and intelligent infrastructures, both industry and academia have made great efforts on this topic recently.

Thanks to the application of convolutional neural networks (CNNs), there has been significant progress in object detection. Most of the mainstream object detection approaches, such as Faster R-CNN [5], R-FCN [6], SSD [7] and YOLO [8], which are all CNN-based, exhibit good performance in the detection of various object classes, such as cars and people, etc. However, there is still no clear guidance for researchers or practitioners regarding the selection of which model is appropriate for specific tasks. Depending on the application, the requirements differ in terms of accuracy, processing time and memory usage. Self-driving cars usually require both high detection precision and fast processing speed. However, for identification of criminals, a security system mainly requires high accuracy. In contrast, applications on mobile devices focus more on a lightweight model with real-time performance. As the foremost mainstream models, which are trained on the data sets of PASCAL VOC [9], ImageNet [10] and Microsoft COCO [11], they are mainly optimized for accuracy, while less considering other factors.

Therefore, two problems persist in current studies. First, comparison of model performance in specific applications is rare. In particular, the full view concerning the trade-off among accuracy, speed and memory usage for vehicle and pedestrian detection has not been explored. Such trade-off normally depends on a multitude of factors, such as meta-architectures, feature extractors, target sizes, etc. Second, it should be more reasonable and convincing to compare the models on the same platform in terms of their running time, memory usage and even accuracy value.

Although the number of recently proposed detection algorithms is huge, most of them are descendants from the families of several popular state-of-the-art approaches. This inspires us to only compare those base methodologies, because they impose a great influence on the performance of their descendants. In this paper, we particularly implement and evaluate meta-architectures based on Faster R-CNN, R-FCN, and SSD. Additionally, this paper combines the above meta-architectures

Manuscript received August 4, 2019; revised November 28, 2019 and February 8, 2020; accepted April 21, 2020. Date of publication May 25, 2021; date of current version June 2, 2021. This work was supported by the National Key Research and Development Program of China under Grant 2018YFB1305002. The Associate Editor for this article was N. Zheng. *(Corresponding author: Dongpu Cao.)*

Long Chen and Shaobo Lin are with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510275, China (e-mail: chenl46@mail.sysu.edu.cn; linshb9@mail2.sysu.edu.cn).

Xiankai Lu is with the Inception Institute of Artificial Intelligence, Abu Dhabi, United Arab Emirates (e-mail: carrierlxk@gmail.com).

Dongpu Cao is with the Waterloo Cognitive Autonomous Driving (CogDrive) Lab, University of Waterloo, Waterloo, ON N2L 3G1, Canada (e-mail: dongpu@uwaterloo.ca).

Hangbin Wu and Chun Liu are with College of Surveying and Geo-Informatics, Tongji University, Shanghai 200092, China (e-mail: hb@tongji.edu.cn; liuchun@tongji.edu.cn).

Chi Guo is with the National Satellite Positioning System Engineering Technology Research Center, Wuhan University, Wuhan 430072, China (e-mail: guochi@whu.edu.cn).

Fei-Yue Wang is with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China (e-mail: feiyue.wang@ia.ac.cn).

Digital Object Identifier 10.1109/TITS.2020.2993926

with various selective feature extractors (such as ResNet50, ResNet101, MobileNet_V1, MobileNet_V2, Inception_V2 and Inception_ResNet_V2) and thus obtain eight models. These models are pretrained on the MS COCO database. To better reveal their performance in vehicle and pedestrian detection, we fine-tune such models on the KITTI data set using a transfer learning method. This is a rare comparison of model performance in specific applications. The evaluations of such models are conducted with standard metrics, such as the average precision, execution time, memory allocation, number of network parameters and floating point operations in a network. To the best of our knowledge, our work is the first to analyze mainstream object detection algorithms transferred to the domain of vehicle and pedestrian detection problems, while considering important factors, including accuracy, inference time, memory usage, etc.

Several recent surveys focusing on object detection [12]–[16] have presented analysis results. However, there are three major differences between our work and that of others. First, we fully evaluate the combination of mainstream frameworks and networks, including frameworks such as Faster R-CNN, R-FCN and SSD and networks such as ResNet50, ResNet101, InceptionNet, Inception_ResNet, MobileNet, etc. Second, we focus on the two important targets on the street, vehicles and pedestrians. Third, we adopt more comprehensive metrics, such as memory usage, inference time, model size and number of FLOPS, not only the basic accuracy in [14].

To summarize, our contributions are mainly as follows:

- We give a detailed survey regarding mainstream object detection approaches based on three meta-architectures (Faster R-CNN, R-FCN, and SSD) along with selective feature extractors (ResNet50, ResNet101, MobileNet_V1, MobileNet_V2, Inception_V2 and Inception_ ResNet_V2).
- We provide analysis and evaluation regarding selective object detectors trained specifically for vehicle and pedestrian detection in terms of various metrics, including accuracy, memory usage, inference time, model size and number of FLOPS.
- We demonstrate that Faster R-CNN ResNet50 provides the best trade-off between accuracy and speed in detection, while Faster R-CNN Inception_V2 achieves the highest accuracy in detecting one class of objects. SSD is weak at detecting small objects and SSD MobileNet is the most suitable choice for embedded and mobile devices.
- Our experimental results can provide guidance for researchers and practitioners in selecting an appropriate detection model for their specific tasks.

II. Related Works

In this section, we review related studies regarding two aspects: traditional object detection approaches and deep learning methods based on CNNs.

A. Traditional Methods

Object detection is a method for identifying and localizing objects in an image via the use of object detectors. Early studies on traditional object detection were mainly based on template matching and part-based models [17]–[19]. Before 1990, geometric representations [20], [21] were the mainstream methods of object detection. After that, the focus of object detection made notable changes, moving towards statistical classifiers (such as SVM [22] and Neural Networks [23]) based on appearance features [24], [25].

By moving from global information [24], [26]–[28]to local representations, the appearance features become invariant to changes in rotation, illumination, scale, translation, etc. In [29], the authors used a Harris corner detector to identify feature locations for epipolar alignment of images taken from differing viewpoints. In [28], the experimental results show that robust object recognition can be achieved in cluttered partially occluded images. Therefore, the appearance features are also invariant to changes in viewpoint and occlusion. Handcrafted features gained tremendous popularity, such as the Scale Invariant Feature Transform (SIFT) [28], Shape Contexts [30], Haar-like features [31], Histogram of Gradients (HOG) [32] and Local Binary Patterns (LBP) [33]. By using support vector machines (SVM), Papageorgiou *et al.* [16] introduced a detector with a sliding window. Viola and Jones [44] utilized AdaBoost for automatic feature selection and fast computation. A cascade network was also introduced for effective detection. Larger progress was made by using features based on gradients. Based on the idea of SIFT [34], Triggs and Dalal [32] popularized HOG by utilizing intensity-based features. Based on integral histograms [35], Zhu *et al.* [36] successfully sped up the computation of HOG features.

B. Deep Learning Methods

For years, handcrafted descriptors and discriminative classifiers played an important role in computer vision, including object detection. However, Convolutional Neural Networks (CNN) [37] achieved breakthrough results in the challenge of image classification in 2012. The success of CNNs in image classification tasks was adapted to the domain of object detection, leading to the invention of Region-based CNN (R-CNN) [38]. Since then, object detection methods have gradually evolved, and many relevant approaches have been invented. Thanks to the availability of GPU computing resources, many large-scale datasets and public challenges, such as ImageNet [10], [39] and MS COCO [11], have had deep learning-based methods successfully applied to them. In a word, the rise of deep learning has led to a profound change in the method of implementing computer vision models [37]. In this subsection, we will review various typical approaches using deep learning for object detection.

CNNs have become the basis of most current mainstream object detection approaches due to their learning ability of hierarchical feature interpretation. Two representatives are OverFeat [40] and R-CNN [41]. The former uses a CNN via a sliding window to obtain bounding boxes and scores, while the latter combines a CNN with regional proposals. R-CNN is actually a multistage method, in which Selective Search [42] is used to generate region proposals from the input image and

the CNN is used to compute corresponding feature maps for localization and classification. Compared to the methods on the basis of handcrafted features, the R-CNN obtains much better performance. However, R-CNN is expensive due to its time consumption and memory usage, which are caused by the separated forward-pass and lack of shared computation for each region proposal.

SPPnet [43] is used to address the slowness of R-CNN by sharing computation. However, this model is still time-consuming because it is trained by a multistage method. In addition, it also causes the problem that its parameters below the SPP layer are difficult to be be optimized.

Fast R-CNN [44] is proposed to help fix the problems of both above models by sharing computation and using a multitask loss. This model replaces the multiple SPP layers in SSPnet by the RoI Pooling layer, which shares the computation with feature extractors. Furthermore, at the end of the network, the multitask loss is calculated by regression and classification heads. Notably, during training, all layers in Fast R-CNN are updated, and thereby the problem of SPPnet is solved. Although Fast R-CNN reduces the running time, there is still a bottleneck caused by the slow region proposal method.

To overcome this bottleneck, Faster R-CNN [5] was thereafter introduced. Faster R-CNN first uses a Region Proposal Network (RPN) to process an input image by a feature extractor, and predicts class-agnostic box proposals based on the features at some selected intermediate levels. Second, these box proposals are processed by the remaining network to refine the box predictions. However, each region is computed separately, and the running time is related to the number of interested regions generated by the RPN. In an improved work, R-FCN [6] adopts location-sensitive score maps to replace the traditional ROI pooling operations. In R-FCN, crops are generated from the last feature layer, which can minimize the time of computation. Therefore, the R-FCN model achieves accuracy comparable to that of Faster R-CNN but with faster execution speed.

Aside from the above models which require a multistage pipeline, single-stage object detectors such as SSD [7] and YOLO [8] have also been proposed. These approaches only apply a single convolutional network to predict both class labels and bounding boxes, achieving a much faster running speed.

In addition to the above mainstream approaches, several 2D and 3D detection methods [45]–[47] are proposed for vehicle or pedestrian detection. [45] uses RGB and LiDAR data for 3D vehicle detection and [47] applies an autoregressive network for pedestrian detection.

As we all know, with the advance of deep learning-based methods, we do not need to design the artificial features that traditional methods require. Besides, deep learning methods are much better than traditional methods in accuracy, so we omit traditional methods due to their lack of practicability and instead only focus on deep learning methods.

III. Frameworks

In this section, we will demonstrate two mainstream deep learning-based object detection frameworks that include a two-stage framework with two separate networks for region proposal and object detection, respectively, as well as a one-stage framework, which is a single neural network that does not apply the region proposal. The structures of the one and two-stage frameworks are presented in Fig. 1.

A. Two-Stage Framework

In a two-stage framework, the input image is first processed to extract region proposals. Second, CNN [37] features are generated from such interested regions, and classifiers are utilized to determine the classes of these proposals. R-CNN, SPPNet, Fast R-CNN, Faster R-CNN and R-FCN all belong to this category and include a region proposal module and an object detection module.

1) R-CNN: Inspired by the breakthrough results obtained by CNN models and the selective search for extracting region proposals [42], Girshick *et al.* applied CNN to object detection and developed R-CNN [38], [48], which integrates selective search [42] with AlexNet [37]. R-CNN was also the first detector with a two-stage framework. By using region proposal methods to process the image, the region proposal step is responsible for outputting interested regions. This region proposal step uses selective search [42] and generates approximately 2,000 region proposals while taking 27 seconds. A region proposal is run through the object detector to generate a feature map, and then two fully connected layers are applied to output a linear vector for classification and regression based on two separate SVM networks [49]. The cross-entropy and the L2 loss are used for classification and regression, respectively. Because every region proposal needs to run through the convolutional base, it takes a very long time to process a single image. However, R-CNN provides a good basis for object detection. Because of its limits in time and space, a more effective model was needed.

2) SPPNet: CNN feature extraction and many others from a number of region proposals for every image are the main bottleneck of R-CNN. To overcome such problems, He *et al.* [50] applied spatial pyramid pooling (SPP) [51], [52] on top of the last convolutional (CONV) layer to generate fixed-length features required by the fully connected layers. With such a SPP module, R-CNN only needs to process the test image once, which can yield a significant speedup without sacrificing any detection accuracy. However, such speedup is not comparable to the training pipeline of the object detector. The CONV layers before the SPP layer cannot be optimized due to the limits on accuracy of SPPnet.

3) Fast R-CNN: Girshick [44] introduced Fast R-CNN to address some of the drawbacks of SPPnet and R-CNN. Fast R-CNN is also a two-stage model. Just like SPPNet, Fast R-CNN aims to reduce the overhead of R-CNN by processing a test image just once. The major innovation of Fast R-CNN is the idea of sharing computation between a region proposal step and an object detector, as well as adding the Region of Interest (RoI) pooling layer after the last CONV layer to get a fixed-length feature vector for each region. By developing a streamlined training process that updates all network layers, Fast R-CNN realizes an end-to-end object detector that, using

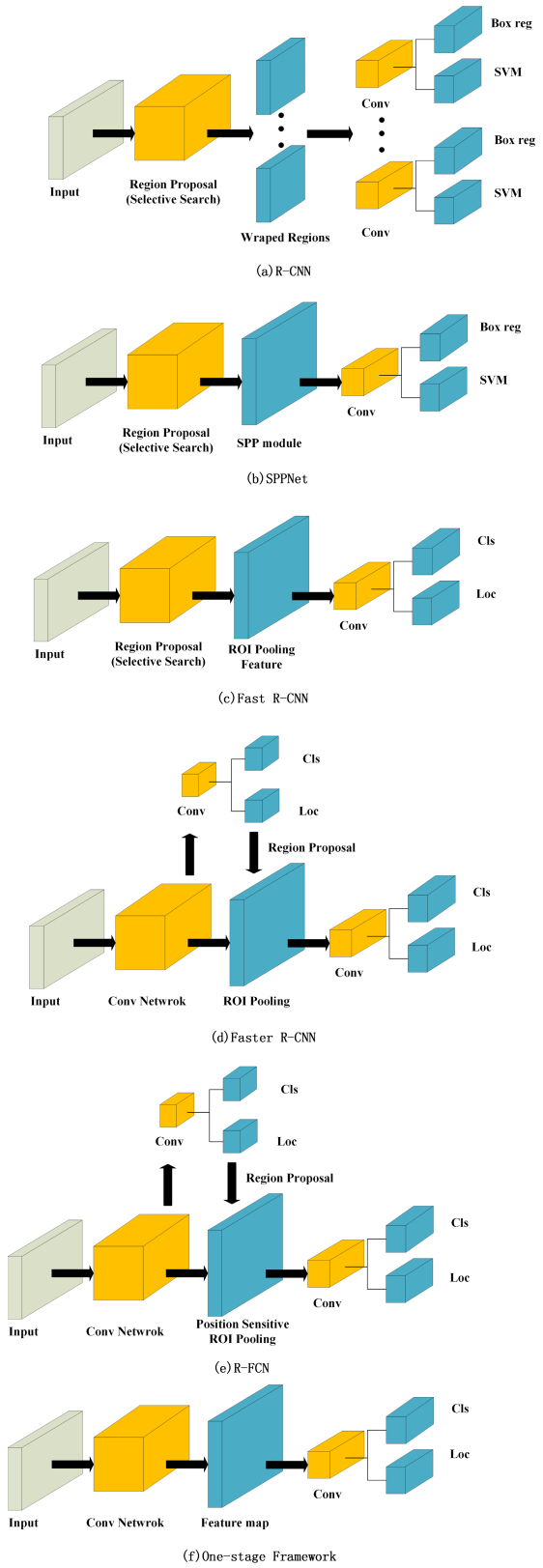


Fig. 1. The structures of the one and two-stage frameworks. (a) - (e) show the structures of two-stage frameworks, including R-CNN, SPPNet, Fast R-CNN, Faster R-CNN, and R-FCN. (f) exhibits the structure of one-stage framework. These frameworks are introduced in this Section.

a multitask loss, simultaneously updates a regression and classification head, rather than the SVMs used in R-CNN and SPPnet. Such changes lead to less inference time and memory consumption. However, the selective search step is exposed as a bottleneck, and a more efficient solution is needed for a real-time system.

4) Faster R-CNN: Although Fast R-CNN speeds up the detection pipeline, it still requires an external region proposal module, which has been proved to be less inefficient than using CNNs directly to localize objects [53]–[57]. Therefore, the selective search used by Fast R-CNN can be replaced by the pure CONV network to extract high-quality interested regions, which leads to the invention of Region Proposal Network (RPN) and Faster R-CNN [5]. RPN, which is a type of fully convolutional network [58], makes Faster R-CNN an end-to-end CNN-based pipeline without using any handcrafted features. The RPN is able to predict regions with various sizes and aspect ratios in the image by using a new concept named anchors. RPN first initializes k n$\times$n anchors of multiple sizes and aspect ratios at each location in the feature map. By mapping the anchors to a low-dimensional feature, we can use each anchor for classification and regression, which are trained with log loss and Smooth L1 loss, respectively.

5) Region-Based Fully Convolutional Network (R-FCN): Dai *et al.* [6] proposed R-FCN, which is a fully convolutional model sharing all computation over the entire image. The RoI subnetwork in R-FCN differs from that in Faster R-CNN. Faster R-CNN [6] uses all convolutional layers to build an RoI subnetwork and extracts RoI regions from the last layer to predict results. By using specialized CONV layers as the output, R-FCN builds the position sensitive maps and applies position-sensitive RoI pooling to get results. In R-FCN, ROI pooling is performed on different feature maps to create an ROI map, which differs from performing ROI pooling on each bin of one feature map in Faster R-CNN. Based on the maximum or average value of an ROI region, a softmax function is used to get the final label probabilities on the classification head. In summary, R-FCN improves the ROI Pooling and the method of sharing features during inference. R-FCN with ResNet101 [59] has comparable accuracy and faster speed compared to Faster R-CNN.

B. One-Stage Frameworks

Two-stage approaches are computationally expensive. Therefore, instead of optimizing the components of a two-stage framework, the research community has started to develop one-stage detection frameworks. One-stage pipelines predict results by using a single feed-forward CONV network that does not involve a region proposal module. The approach is simple because it removes the region proposal step and merges all computation into just one stage, which causes the model to be optimized end-to-end directly. YOLO and SSD both belong to this category.

1) YOLO: YOLO, proposed by Redmon *et al.* [8], is a unified object detection method that predicts class probabilities and bounding box offsets using a single network. Because the region proposal step is removed, YOLO uses the original images to predict results directly. In detail, an input image is separated by YOLO into an S$\times$S grid, and C label probabilities, B bounding box information and corresponding

confidence scores for each grid are finally obtained. These results are represented as a tensor with a size of $S\times S\times (5B + C)$. YOLO is a real-time model that runs at 45 FPS, and Fast YOLO [8] even achieves 155 FPS. As discussed in [8], it is easy for YOLO to make more mistakes in the localization of small objects, because the division of the grid is rough, and each cell only contains a few objects.

YOLOv2, YOLO9000 and YOLOv3: Redmon and Farhadi [60] introduced YOLOv2 to improve YOLO by using multiple strategies, such as replacing the GoogLeNet [61] with a simpler network called DarkNet19, using a batch normalization technique [62], throwing out the fully connected layers, and applying anchor boxes learned from k-means and multiscale strategy. At the same time, Redmon and Farhadi [60] also proposed YOLO9000. By adopting a joint optimization approach to train and a WordTree to merge data from multiple input sources, YOLO9000 is able to classify a large number of object categories in real-time. Following YOLOv2, YOLOv3 [63] was proposed by using an improved network called DarkNet53.

2) SSD: By combining the ideas from Faster R-CNN [5], YOLO [8] and multiscale CONV features [54], Liu *et al.* [7] proposed SSD, which is more effective than YOLO [8] and has comparable accuracy to Faster R-CNN [5]. SSD also adopts the anchors proposed in Faster R-CNN to define the regions in an image. SSD is a fully convolutional model, in which the size of feature maps from input to output progressively decreases and multiple features are linked to the end of their network. Utilizing a high-resolution feature map, SSD can detect small objects. Just like YOLO, SSD can predict the bounding boxes and corresponding scores, and it employs the NMS operation to yeid the final detection.

IV. Network Backbones

CNNs are used as the network backbones in the detection approaches. Representative networks include AlexNet [37], VGGNet [64], GoogLeNet [61], Inception series [65]–[67], ResNet [59], DenseNet [68] and MobileNets [69], [70]. We present some important structures in several networks in Fig. 2 and Fig. 3.

Briefly, a CNN consists of several layers, such as convolutional and pooling layers. For example, the first AlexNet [37] has 5 CONV layers, and 2 FC layers are applied last. In general, the CNN models extract features from low-level information, such as edges to high-level information, such as objects [71]–[74].

During the development of CNN models, networks are becoming deeper. For example, AlexNet is composed of 8 layers, and VGGNet consists of 16 layers. ResNet and DenseNet have more than 100 layers. It is important to notice that some networks, including AlexNet and VGGNet, are only a few layers deep but have an enormous number of parameters because FC layers provide the main portion of parameters. However, networks such as the Inception network, ResNet, and DenseNet have fewer parameters by avoiding the use of FC layers, despite having very large network depth.

With the use of Inception modules, the parameters of GoogLeNet are dramatically reduced. Inception_V2 and

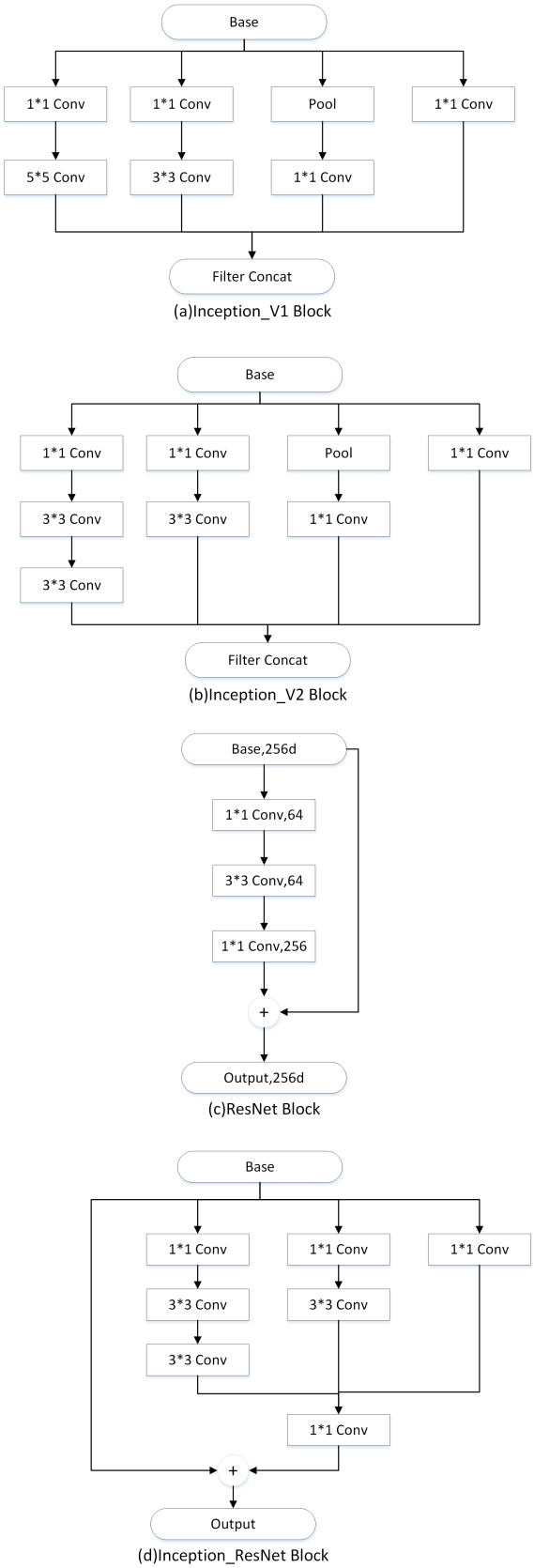


Fig. 2. Representative blocks of InceptionNets (a,b), ResNet (c), and Inception_ResNet (d).

V3 [66] are proposed to factorize large convolutions to smaller convolutions and asymmetric convolutions. For example, they have factorized the traditional 7×7 convolution into

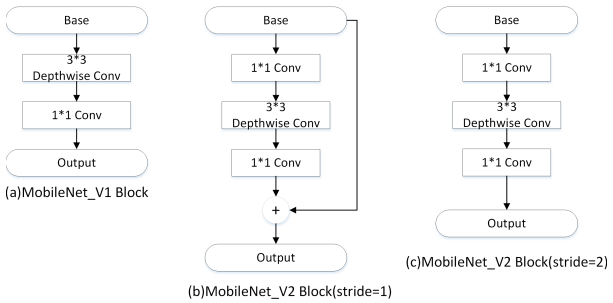


Fig. 3. Representative blocks of MobileNet _V1 (a) and MobileNet _V2 (b,c) with different strides.

three 3×3 convolutions. Combining a 3×1 convolution with a 1×3 convolution is equivalent to using two 3×3 convolutions. By applying factorization, the parameters of networks are reduced further. ResNet demonstrates the importance of shortcut connections for learning the parameters in very deep networks. Inspired by Inception and ResNet [59], Inception_ResNets [67] combine the Inception modules with skip connections, proving that skip connections are able to speed up the training of Inception networks. Based on ResNets, Huang *et al.* [68] introduced DenseNets, which construct dense blocks by connecting layers using a feed-forward method. This type of model has several considerable benefits, including high-efficiency parameters, learning supervision and feature reuse.

MobileNets [69], [70] are introduced for mobile and embedded vision applications. MobileNet_V1 [69] is a light-weight network that utilizes depth-wise separable convolutions (DWs). With two proposed global hyperparameters, the model is able to achieve an effective trade-off between speed and accuracy. MobileNet_V2 [70] adds skip connections which are proposed in ResNets to MobileNet_V1 and employ a linear activation at the last layer of DWs.

V. Experimental Setup

In this section, we mainly study three meta-architectures (i.e., Faster R-CNN, R-FCN, and SSD) and six feature extractors (i.e., ResNet50, ResNet101, MobileNet_V1, MobileNet_V2, Inception_V2 and Inception_ResNet_V2), which are able to obtain high-level features in image classification tasks. All of the above presented models are pretrained on the COCO dataset.[1] We fine-tune these models by using transfer learning [76] on the KITTI dataset to detect and classify vehicles and pedestrians in street scenes. Details regarding combinations of the three meta-architectures and six feature extractors selected in our experiments are shown in Table I. Note that, instead of all possible combinations, here we only investigate those that are mostly employed in current mainstream methods. The hyperparameters of the models chosen in our work are all the same, while the training strategies are different to achieve the best performance of each model.

[1]All pre-trained networks are available in the Google Object Detection API [75]

TABLE I

Combinations of Meta-Architectures With Feature Extractors Investigated, Where the Check Mark Denotes That the Combination Is Employed

Models	Faster R-CNN	R-FCN	SSD
ResNet50	✓		
ResNet101	✓	✓	
MobileNet_V1			✓
MobileNet_V2			✓
Inception_V2	✓		✓
Inception_ResNet_V2	✓		

Experiments are mainly conducted on the KITTI benchmark. KITTI is a street scene dataset captured by a vehicle. Each image contains up to 15 cars and 30 pedestrians. For the detection task, the KITTI dataset provides 7481 training images. Here we set aside the first 500 images as a validation set and train models to detect cars and pedestrians. To detect small objects, we also resize the height of the original KITTI images to 600 pixels while maintaining the aspect ratio for all meta-architectures except SSD.

A. Meta-Architectures

In this section, we experiment on three object detection meta-architectures including Faster R-CNN, R-FCN, and SSD.

1) Faster R-CNN: This approach introduces an RPN to make the model trainable in an end-to-end fashion. Here, the authors use k to denote the number of region proposals at each sliding-window location. These regions are parameterized with anchors with several specific aspect and scale ratios. In this paper, we use 4 scales (0.25, 0.5, 1, 2) and 3 aspect ratios (1:2, 1:1, 2:1), and thus we get 12 anchors at each location. However, there are RPN proposals that highly overlap with others. Therefore, we first use the non-maximum suppression (NMS) method to remove corresponding proposals based on their objectness scores. We set the IoU threshold of the first stage to 0.7. Only the top-N ($N \leq 300$) ranked proposals are required by a detection step. Another NMS is applied for post-processing by setting the IoU threshold to 0.6 to get the final detection results. We set the maximum detections per class to 100, and the maximum of the total detections is 300. The weights of localization and classification loss are respectively set to 2 and 1 in both stages. Moreover, each feature extractor is optimized by an SGD along with a batch size of 1.

2) R-FCN: The architecture and the training of R-FCN are the same as those of Faster R-CNN, while R-FCN adopts a fully convolutional RPN. The position-sensitive score map is introduced to solve the problems of translation-invariance in classification tasks and translation-variance in detection tasks.

3) SSD: In contrast to the framework of Faster R-CNN, the SSD model completely eliminates the RPN and merges all computation into a single CNN. SSD also uses anchor boxes, which are based on the size of the detected objects. At each location of the feature map, SSD can predict the offsets relative to the default boxes, as well as the category scores. Moreover, the network combines results from multiple scales to naturally process objects with various sizes.

TABLE II
TRAINING STRATEGY (ITERATION NUMBER) OF DIFFERENT MODELS

	Models	$stage\_1(K)$	$stage\_2(K)$	$stage\_total(K)$
	Faster R-CNN			
1	ResNet50_all	500	700	800
2	ResNet50_car	500	700	800
3	ResNet50_ped	500	700	800
4	ResNet101_all	500	700	800
5	ResNet101_car	500	700	800
6	ResNet101_ped	300	500	600
7	Inception_V2_all	500	700	800
8	Inception_V2_car	500	700	800
9	Inception_V2_ped	500	700	800
10	Incep_ResNet_all	200	300	400
11	Incep_ResNet_car	200	300	400
12	Incep_ResNet_ped	200	300	400
	R-FCN			
13	ResNet101_all	300	400	500
14	ResNet101_car	300	400	500
15	ResNet101_ped	300	400	500

For experimentation, the NMS is applied for postprocessing with an IoU threshold of 0.6, and the maximum detections per class and the maximum of total detections are both set to 100. The weight of localization and classification loss are both set to 1. The optimizer used by SSD is the RMSprop and the batch size is 24. The initial learning rate is 0.004 which is decayed by a factor of 0.95 for each 800K iterations, except when using MobileNet_V2 for pedestrian detection (315K iterations).

The training strategies of all Faster R-CNN and R-FCN methods are depicted in Table II, in which three training stages are presented. The initial learning rates of models are 0.001 in the first stage and are manually reduced by $100\times$ in the second stage and by another $10\times$ in the third stage. Taking the first model as an example, the learning rate is 0.001 during the first 500K iterations, decays to 0.00001 from 500K to 700K iterations, and is 0.000001 from 700K to the total training iterations (800K). The first 12 models all use the Faster R-CNN meta-architecture. In Table II, We use *car*, *ped* and *all* to respectively represent three detection tasks which include the detection of cars, pedestrians and both of them.

In the beginning of experiment, we explore the learning rates for all models. For training models of Faster R-CNN with ResNet50 and Faster R-CNN with ResNet101, the initial learning rate is 0.0001, and it then increases progressively to 0.0002, 0.0003, 0.0005, 0.0008 and 0.01. We find that increasing the learning rate from 0.0002 to 0.0008 can greatly improve the model performance, but using 0.001 only obtains a tiny improvement. Therefore, we select 0.001 as the initial learning rate for all models for the purpose of a unified training strategy. Moreover, by observing the training curves, we also find that it is able to ensure the convergence of all models. Notably, some models require fewer training iterations due to overfitting. Thus, we further adjust their training iterations for the best performance.

B. Feature Extractors

We use several mainstream CNNs to extract features from the input images, which are ResNet50, ResNet101, MobileNet_V1, MobileNet_V2, Inception_V2 and Inception_ResNet_V2.

1) ResNet50 and ResNet101: Both are deep residual networks [59] and are employed as the feature extractors for Faster R-CNN and R-FCN. ResNets are separated into two stages, i.e., generation of region proposals and extraction of classification features. Here, we get the features from the $conv4\_x$ layer and resize them to 14×14, and then we use max-pooling to get 7×7 pixels. The stride we use is 16 pixels.

2) MobileNet_V1 and MobileNet_V2: MobileNet V1 [69] was designed for mobile applications by using the depth-wise separable convolution modules, which can reduce the cost of computation and the number of parameters. MobileNet V2 [77] applies the inverted residual and linear bottleneck modules to update the V1 version. In our experiments, these two feature extractors are only combined with the SSD meta-architecture. Because the network structure is not separated, auxiliary multiple-scale feature maps are also required. Compared to combining Inception_V2 with SSD, we adopt layers of $conv_11$ and $conv_13$, and append four additional convolutional layers of decayed resolution with depths of 512, 256, 256 and 128, respectively.

3) Inception_V2: Inception units in this architecture are capable of improving the width and depth of a CNN model without increasing its computational cost. In our experiments, when combining Faster R-CNN with Inception_V2, we obtain the features of the layer $Mixed\_4e$ with a 16-pixel stride size. Feature maps are then resized to 14×14 pixels.

When SDD is the meta-architecture utilized, selection of an entire Inception V2 as the feature extractor is not required. However, additional multiple-scale feature maps are needed. Here, we use the layers of $Mixed\_4c$ and $Mixed\_5c$, and there are four auxiliary convolutional layers of decayed resolution with depths of 512, 256, 256 and 128, respectively.

4) Inception_ResNet_V2: Inception_ResNet_V2 combines the benefits of Inception units with skip connections. In our experiments, this network is only used in combination with Faster R-CNN. We obtain features of the layer $Mixed\_6a$ and its associated residual layers with the stride size of 16 pixels. Like ResNet, feature maps need to be resized to 14×14, leading to a final 7×7 by using max-pooling.

VI. RESULTS

In this section, we demonstrate the performance of the car and pedestrian detectors by combing meta-architectures and feature extractors as described above. Object detection models are evaluated with the COCO-style AP (Average Precision), AP_{50} (over IoU threshold of 50%) and AP_{75}. In addition, we analyze the COCO-style AP on three types of objects with small, medium, and large sizes (AP_S, AP_M, and AP_L) following the definitions in [11]. Moreover, we evaluate the models by several additional metrics, such as execution time, memory allocation, number of FLOPS and parameter size. We train and evaluate all models on a computer with an Intel Core i7-7820X CPU and a NVIDIA GeForce GTX GPU equipped with 11GB of GPU memory and CUDA 9.0. We use the TensorFlow Object Detection API as the development tool.

Evaluation metrics, such as the memory consumption, parameter size, and FLOPS of the model, can be measured independently of the platform. Nevertheless, the execution

TABLE III

COCO-STYLE METRICS OF DIFFERENT MODELS ON CAR AND PEDESTRIAN DETECTION

Models	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster R-CNN ResNet50	58	84.2	63.9	49.1	57.9	67.7
Faster R-CNN ResNet101	57.9	83.7	63.7	50.2	57.6	67.5
Faster R-CNN Inception_ResNet_V2	57.2	81.9	63.5	47.2	58.2	66
R-FCN ResNet101	54.9	79.8	61.7	45.7	55	64.5
Faster R-CNN Inception_V2	53	81	58.9	40.4	54	63.7
SSD MobileNet_V2	27.9	52.2	25.9	10.8	24.2	51.1
SSD Inception_V2	27.5	54.7	23.8	11.2	25.2	49.2
SSD MobileNet_V1	25.2	49	23.4	6.2	22.2	46.9

TABLE IV

COCO-STYLE METRICS OF DIFFERENT MODELS ON CAR DETECTION

Models	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster R-CNN Inception_V2	74.5	91.8	85.4	65.7	73.8	80.9
Faster R-CNN ResNet50	73.6	90.5	85	65.7	71.7	80.9
Faster R-CNN ResNet101	72.6	89.8	83.8	66	71.7	78.9
Faster R-CNN Inception_ResNet_V2	72.1	89.9	83	65.1	71	78.7
R-FCN ResNet101	69.3	86.4	81.3	63.6	67.6	76.2
SSD Inception_V2	47.8	78.9	49.8	20.9	43.4	70.8
SSD MobileNet_V2	46.2	76.7	47.4	14.3	40.4	71.1
SSD MobileNet_V1	42.7	73.9	42.8	12.7	38.2	69.2

TABLE V

COCO-STYLE METRICS OF DIFFERENT MODELS ON PEDESTRIAN DETECTION

Models	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster R-CNN Inception_V2	47.3	82.3	48.1	39.5	48.4	57.6
Faster R-CNN ResNet50	46.2	79.4	50.2	35.9	48.1	56.7
Faster R-CNN Inception_ResNet_V2	44.7	78.3	45	32.3	46	57.8
Faster R-CNN ResNet101	44.5	77.7	48.1	34.2	45.9	56.9
R-FCN ResNet101	42.7	75.5	45.6	30.4	44.8	55.3
SSD Inception_V2	18	43	12.1	4.9	17.3	43.8
SSD MobileNet_V2	12.9	31.1	10	4	13.7	33.6
SSD MobileNet_V1	9.6	24.8	4.2	4.5	7.7	30.2

time is influenced by the configuration of the computers, which may differ in software drivers, hardware, frameworks, etc. Both execution time and memory occupation are averaged over 500 images (of the testing set).

A. Analyses

Detailed accuracy results on car and pedestrian detection are presented in Tables III-V along with the COCO-style AP attained by each detector. On car and pedestrian detection, Faster R-CNN ResNet50 achieves the best performance (58%), while Faster R-CNN Inception_V2 is 0.9% better than Faster R-CNN ResNet50 on car detection and 1.1% better than Faster R-CNN ResNet50 on pedestrian detection.

Obviously, a positive correlation between AP and AP_{50} or AP_{75} can be seen in Fig. 4. Thus, for simplification purposes, we only use AP to illustrate the performance of tested models.

Other metrics, such as FPS, Memory, GigaFLOPS and Parameters, are shown in Table VI and ordered by AP. Based on the evaluation results, detailed analysis is given as follows:

On the task of car and pedestrian detection, Faster R-CNN ResNet50 achieves the best performance, followed by ResNet101 and Inception ResNet. As we all know, ResNet is suitable for difficult tasks, such as datasets with a large number of images or objects, which is the task of car and pedestrian detection in this paper. ResNet adopts skip connections to avoid gradient vanishing and achieves excellent performance. ResNet101 and Inception ResNet, which are much deeper and more complicated than ResNet50, are not easy to train with simple datasets. That is why the performance of ResNet50 is better. In two other experiments, Faster R-CNN Inception_V2 performs better than ResNet because, for the simple task, only one object needs to be detected. Due to the differences between two-stage and one-stage frameworks, we can also notice the huge gap between Faster R-CNN and SSD which are both equipped with Inception_V2. In particular, the MobileNets cases have the lowest AP because they use simple network structures and depth-wise separable convolution modules, but the other metrics are excellent, such as the FPS, Memory, etc. In contrast, the ResNets have high memory consumption and low execution speed.

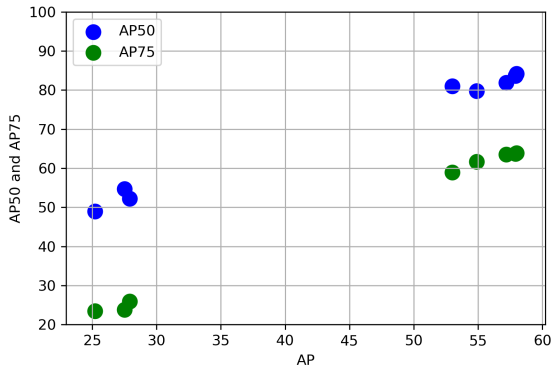


Fig. 4. Overall COCO-style AP for all experiments plotted against corresponding AP_{50} and AP_{75}.

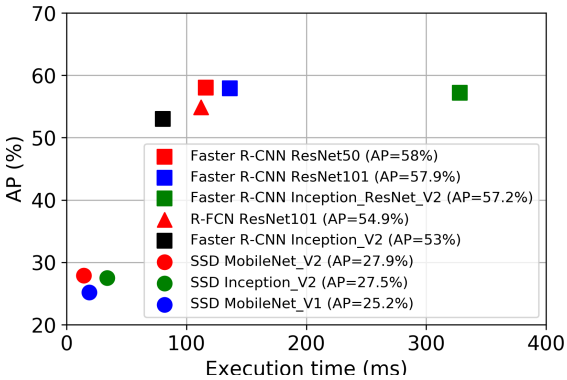


Fig. 5. Accuracy vs execution time.

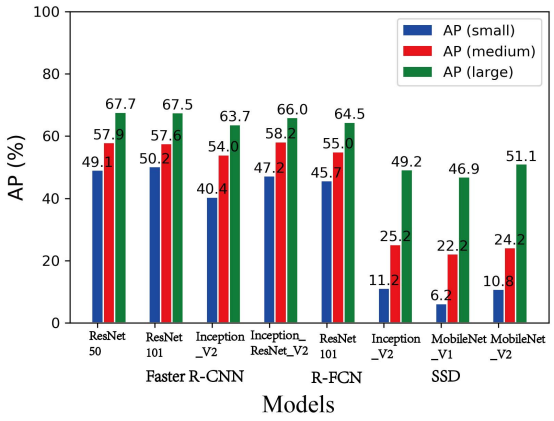


Fig. 6. Accuracy observed by modifing meta-architecture, feature extractor and object size.

TABLE VI
THE RESULTS OF SEVERAL CRITICAL METRICS OF TESTED MODELS

Models	AP (%)	Time (ms)	FPS	Memory (MB)	GFLOPS	Parameters (m)
Faster R-CNN ResNet50	58	116.17	8.61	7910	439.61	43.25
Faster R-CNN ResNet101	57.9	136.13	7.35	9907	439.66	62.24
Faster R-CNN Inception_ResNet_V2	57.2	328.07	3.05	8686	184.28	59.34
R-FCN ResNet101	54.9	112.04	8.93	7029	136.2	64.42
Faster R-CNN Inception_V2	53	80.2	12.47	3712	69.18	12.87
SSD MobileNet_V2	27.9	14.29	70	901	1.31	4.63
SSD Inception_V2	27.5	33.84	29.6	1180	7.57	13.32
SSD MobileNet_V1	25.2	18.89	52.9	875	2.28	5.53

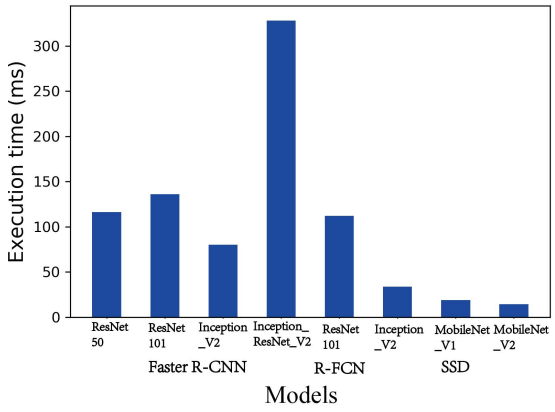


Fig. 7. Execution time for each model.

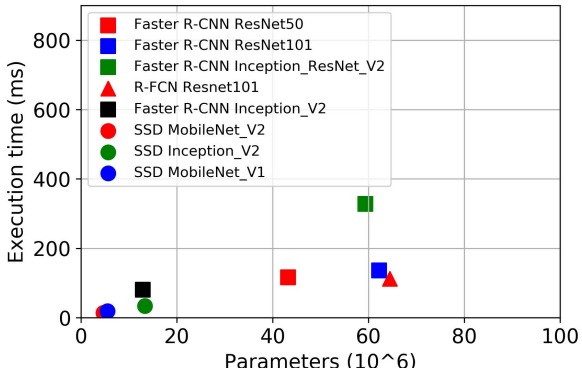


Fig. 8. Execution time vs parameters.

We know the best model on the KITTI leaderboard has higher accuracy than our presented models, the reason is that we just combine the mainstream structures and networks, and we do not change or optimize the components. However, other metrics used in our work are not mentioned on the leaderboard.

1) Accuracy vs. Time: The inference time is an important factor for real-time detection models. Fig. 5 visualizes the AP of each of our model configurations together with their processing time.

In this scatterplot, three group models are demonstrated. The first group comprises the models having faster inference speed, which are based on SSD. SSD MobileNet_V2 is the fastest (14.29 ms per image and 70 FPS) with 0.4% and 2.7% better accuracy than SSD with Inception_V2 and MobileNet_V1. However, SSD-based models still have poor performance in accuracy. The second group consists of Faster R-CNN and R-FN with ResNets or Inception networks. The models in this group have higher accuracy and require approximately 100 ms per image. Faster R-CNN Inception_ResNet_V2 in the third group attains an AP of 57.2%. However, this model is the slowest, requiring 300 ms per image. Consequently, Faster

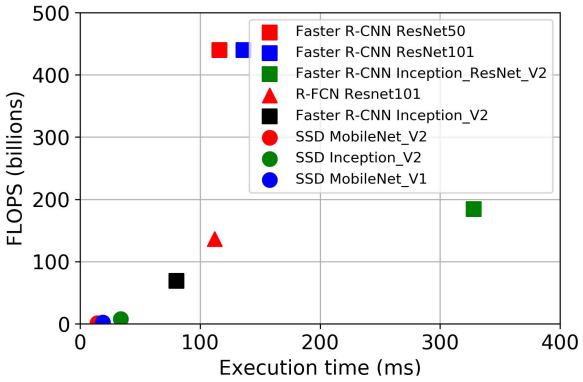


Fig. 9. FLOP vs execution time.

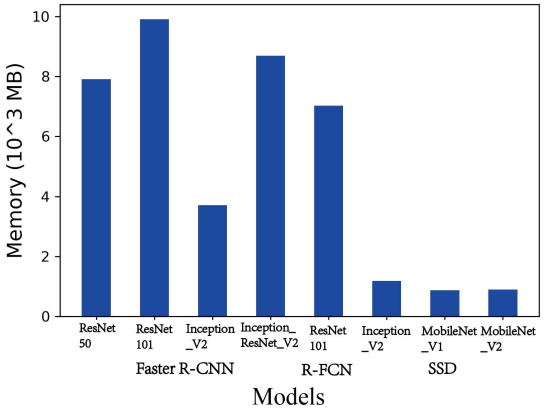


Fig. 10. Memory usage for each model.

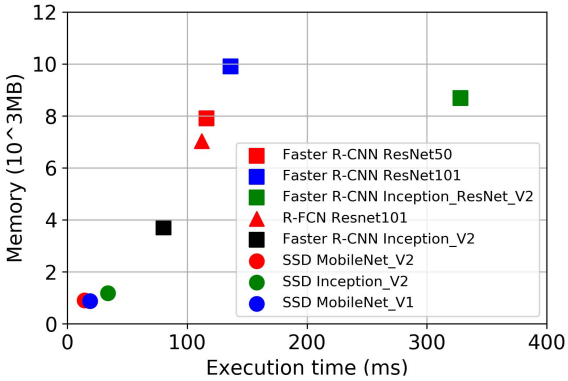


Fig. 11. Memory vs execution time.

R-CNN with ResNet50 achieves an AP of 58% and takes 116 ms per image (8.61 FPS), leading to the best trade-off between speed and accuracy.

2) Effect of Object Size: Fig. 6 shows the model performance with different object sizes. All detectors perform better on large objects. One possible reason is that the initial models

Fig. 12. Qualitative comparison of different models on the KITTI dataset.

are pretrained on the COCO database, whose samples often contain large objects in the center area.

We also conclude that SSD models are not good at detecting small objects (10.8%, 11.2%,6.2%) but are competitive in detecting large objects (51.1%, 49.2%,46.9%). Because pedestrians are usually small and vehicles are larger, SSD models perform unsatisfactorily on pedestrian detection but reach higher accuracy scores than Faster R-CNN and R-FCN on car detection.

3) Running Time Analysis: Fig. 7 shows the execution times for the studied models. This factor is platform-dependent. However, we conclude that SSD models are faster than other models and that Faster R-CNN and R-FCN tend to be slower, especially Faster R-CNN Inception_ResNet_V2, which requires at least 300 ms per image. Moreover, the running time is not directly related to the size of learned parameters (weights and bias), as observed in Fig. 8. We can conclude that models containing ResNet have relatively more parameters and long execution time, while the detectors are more lightweight and faster when equipped with Inception_V2 or MobileNet.

4) FLOPS Analysis: Fig. 9 shows the FLOP number against execution time. Counting FLOPS gives us a platform-independent metric for measuring computation. The dense blocks used in ResNets cause higher FLOPS numbers and more execution time than the Faster R-CNN and R-FCN models. However, SSD MobileNet_V2 has the lowest FLOPS numbers and fastest running speed. Faster R-CNN ResNet101 and ResNet50 have high FLOPS with moderate execution time. Faster R-CNN Inception_ResNet_V2 is exactly the opposite. The FLOPS number, which is not linear with execution time is caused by multiple factors, such as memory I/O, hardware optimization, etc.

5) Memory Analysis: The consumption of memory is also an important factor to decide the hardware configurations required, such as a single GPU or more. It also decides whether the trained models can be applied on our mobile devices.

Fig. 10 observes the memory usage for each model combination and shows that the memory usage of MobileNet is the lightest, which is less than 1GB. Notably, the residual models require the highest memory usage, while the SSD

models with Inception_V2 and MobileNets are cheaper as they require memory usages of 875 MB, 901 MB and 1180MB, respectively.

Fig. 11 plots the memory consumption against the execution time of the studied models. Overall, we observe that more powerful feature extractors have higher running time and require more memory.

6) Qualitative Results on the KITTI Dataset: In Fig. 12, the qualitative comparison is presented for the car and pedestrian detection in images from the KITTI dataset by using previously discussed models. Detection results of three common scenarios are represented, which only contains cars ((a), (b)), both cars and pedestrians ((c), (d)) and only pedestrians ((e), (f)). Overall, we conclude that all of the models perform well in detecting cars in images (a) and (b). In images (c) and (d) of Fig. 12, almost all models are able to detect the existence of cars and pedestrians in a simple scene. The only exception is SSD MobileNet, which does not detect the person in (c). However, in image (d), we notice that Faster R-CNN ResNet101, R-FCN ResNet101, SSD MobileNet_V1 and SSD Inception_V2 all ignore the person on the right side of that image. Because the scene displayed in (d) is more complex, the models are unlikely to detect partially obscured objects. Image (e) and (f) show the results of pedestrian detection. Obviously, all models present difficulties in detecting all persons in images (e), especially SSD MobileNet_V1. In image (f), Faster R-CNN and SSD combined with Inception_V2 mislabel multiple irrelevant objects as pedestrians, and SSD MobileNet_V1 and SSD Inception_V2 both ignore the people on the right side of the image. Other models obtain almost the same results, in which most of the visible persons are detected.

VII. Conclusion

Cars and pedestrians are the most important participants on the road. In this paper, we perform an experimental comparison of several mainstream detectors by combining various meta-architectures and alternative feature extractors. All methods are pretrained on the COCO database and fine-tuned on the KITTI database to recognize cars and pedestrians. We report the extensive evaluation results of these detectors in terms of the accuracy, execution time, memory consumption, number of FLOPS, and parameter size. We conclude that Faster R-CNN ResNet 50 achieves the highest AP and the best balance between processing speed and accuracy. Faster R-CNN Inception_V2 performs best in detecting cars or detecting pedestrians. SSD MobileNet V2 is the fastest model, and SSD MobileNet V1 is the lightest model, both of which are suitable for applications on mobile and embedded devices. Though the research area is dynamic, the results illustrated in our work, such as the accuracy, processing time and memory consumption, are still useful for selecting a suitable model, especially for industrial applications.

References

[1] L. Chen, X. Hu, T. Xu, H. Kuang, and Q. Li, "Turn signal detection during nighttime by CNN detector and perceptual hashing tracking," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 12, pp. 3303–3314, Dec. 2017.

[2] L. Chen, L. Fan, G. Xie, K. Huang, and A. Nuchter, "Moving-object detection from consecutive stereo pairs using slanted plane smoothing," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 11, pp. 3093–3102, Nov. 2017.

[3] L. Chen *et al.*, "Surrounding vehicle detection using an FPGA panoramic camera and deep CNNs," *IEEE Trans. Intell. Transp. Syst.*, to be published.

[4] Q. Li, L. Chen, M. Li, S.-L. Shaw, and A. Nüchter, "A sensor-fusion drivable-region and lane-detection system for autonomous vehicle navigation in challenging road scenarios," *IEEE Trans. Veh. Technol.*, vol. 63, no. 2, pp. 540–555, Feb. 2014.

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

[6] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 379–387.

[7] W. Liu *et al.*, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2016, pp. 21–37.

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

[9] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[11] T.-Y. Lin *et al.*, "Microsoft coco: Common objects in context," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 740–755.

[12] A. Brunetti, D. Buongiorno, G. F. Trotta, and V. Bevilacqua, "Computer vision and deep learning techniques for pedestrian detection and tracking: A survey," *Neurocomputing*, vol. 300, pp. 17–33, Jul. 2018.

[13] E. Arnold, O. Y. Al-Jarrah, M. Dianati, S. Fallah, D. Oxtoby, and A. Mouzakitis, "A survey on 3D object detection methods for autonomous driving applications," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 10, pp. 3782–3795, Oct. 2019.

[14] L. Jiao *et al.*, "A survey of deep learning-based object detection," 2019, *arXiv:1907.09408*. [Online]. Available: http://arxiv.org/abs/1907.09408

[15] L. Chen, Q. Wang, X. Lu, D. Cao, and F.-Y. Wang, "Learning driving models from parallel end-to-end driving data set," *Proc. IEEE*, vol. 108, no. 2, pp. 262–273, Feb. 2020.

[16] L. Chen, W. Zhan, W. Tian, Y. He, and Q. Zou, "Deep integration: A multi-label architecture for road scene recognition," *IEEE Trans. Image Process.*, vol. 28, no. 10, pp. 4883–4898, Oct. 2019.

[17] M. A. Fischler and R. A. Elschlager, "The representation and matching of pictorial structures," *IEEE Trans. Comput.*, vol. C-22, no. 1, pp. 67–92, Jan. 1973.

[18] F. Göngör and Ö. Tutsoy, "Design and implementation of a facial character analysis algorithm for humanoid robots," *Robotica*, vol. 37, no. 11, pp. 1850–1866, Nov. 2019.

[19] F. Gongor and O. Tutsoy, "Humanoid robots learn and recognize seven facial emotions with ANN," in *Proc. Int. Sci. Acad. Congr. (INSAC)*, vol. 18, 2018.

[20] J. L. Mundy, "Object recognition in the geometric era: A retrospective," in *Toward Category-Level Object Recognition*. Springer, 2006, pp. 3–28.

[21] J. Ponce, M. Hebert, C. Schmid, and A. Zisserman, *Toward Category-Level Object Recognition*, vol. 4170. Springer, 2007.

[22] E. Osuna *et al.*, "Training support vector machines: An application to face detection," in *Proc. CVPR*, 1997, pp. 130–136.

[23] H. A. Rowley, "Neural network-based face detection," Dept. Comput. Sci., Carnegie-Mellon Univ., Pittsburgh, PA, USA, Tech. Rep., 1999.

[24] H. Murase and S. K. Nayar, "Visual learning and recognition of 3-D objects from appearance," *Int. J. Comput. Vis.*, vol. 14, no. 1, pp. 5–24, Jan. 1995.

[25] C. Schmid and R. Mohr, "Local grayvalue invariants for image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, no. 5, pp. 530–535, May 1997.

[26] M. J. Swain and D. H. Ballard, "Color indexing," *Int. J. Comput. Vis.*, vol. 7, no. 1, pp. 11–32, 1991.

[27] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Mar. 1991, pp. 586–591.

[28] D. G. Lowe *et al.*, "Object recognition from local scale-invariant features," in *Proc. ICCV*, no. 2, 1999, pp. 1150–1157.

[29] Z. Zhang, R. Deriche, O. Faugeras, and Q.-T. Luong, "A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry," *Artif. Intell.*, vol. 78, nos. 1–2, pp. 87–119, Oct. 1995.

[30] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 4, pp. 509–522, Apr. 2002.

[31] P. Viola *et al.*, "Rapid object detection using a boosted cascade of simple features," in *Proc. CVPR*, 2001, vol. 1, no. 1, pp. 511–518.

[32] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR05)*, vol. 1, 2005, pp. 886–893.

[33] T. Ojala, M. Pietikäinen, and T. Mäenpää, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 971–987, Jul. 2002.

[34] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004.

[35] F. Porikli, "Integral histogram: A fast way to extract histograms in Cartesian spaces," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 1, Jun. 2005, pp. 829–836.

[36] Q. Zhu, M.-C. Yeh, K.-T. Cheng, and S. Avidan, "Fast human detection using a cascade of histograms of oriented gradients," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 2, Jun. 2006, pp. 1491–1498.

[37] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[38] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 580–587.

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

[40] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "OverFeat: Integrated recognition, localization and detection using convolutional networks," 2013, *arXiv:1312.6229*. [Online]. Available: http://arxiv.org/abs/1312.6229

[41] K. Lenc and A. Vedaldi, "R-CNN minus R," 2015, *arXiv:1506.06981*. [Online]. Available: http://arxiv.org/abs/1506.06981

[42] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *Int. J. Comput. Vis.*, vol. 104, no. 2, pp. 154–171, Sep. 2013.

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2014, pp. 346–361.

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

[45] J. Dou, J. Xue, and J. Fang, "SEG-VoxelNet for 3D vehicle detection from RGB and LiDAR data," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 4362–4368.

[46] T. S. Combs, L. S. Sandt, M. P. Clamann, and N. C. McDonald, "Automated vehicles and pedestrian safety: Exploring the promise and limits of pedestrian detection," *Amer. J. Preventive Med.*, vol. 56, no. 1, pp. 1–7, Jan. 2019.

[47] G. Brazil and X. Liu, "Pedestrian detection with autoregressive network phases," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 7231–7240.

[48] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 1, pp. 142–158, Jan. 2016.

[49] J. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machines," Tech. Rep., 1998.

[50] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, Sep. 2015.

[51] K. Grauman and T. Darrell, "Pyramid match kernels: Discriminative classification with sets of image features (version 2)," Tech. Rep., 2006.

[52] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 2, Jun. 2006, pp. 2169–2178.

[53] R. G. Cinbis, J. Verbeek, and C. Schmid, "Weakly supervised object localization with multi-fold multiple instance learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 1, pp. 189–203, Jan. 2017.

[54] B. Hariharan, P. Arbelaez, R. Girshick, and J. Malik, "Object instance segmentation and fine-grained localization using hypercolumns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 4, pp. 627–639, Apr. 2017.

[55] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Is object localization for free?—Weakly-supervised learning with convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 685–694.

[56] Z. Bolei, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Object detectors emerge in deep scene CNNs," Tech. Rep., 2015.

[57] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2921–2929.

[58] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3431–3440.

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

[60] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," 2016, *arXiv:1612.08242*. [Online]. Available: http://arxiv.org/abs/1612.08242

[61] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[62] 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. Vis. (ICCV)*, Dec. 2015, pp. 1026–1034.

[63] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, *arXiv:1804.02767*. [Online]. Available: http://arxiv.org/abs/1804.02767

[64] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556

[65] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, *arXiv:1502.03167*. [Online]. Available: http://arxiv.org/abs/1502.03167

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

[67] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proc. AAAI*, vol. 4, 2017, p. 12.

[68] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.

[69] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*. [Online]. Available: http://arxiv.org/abs/1704.04861

[70] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

[71] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[72] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

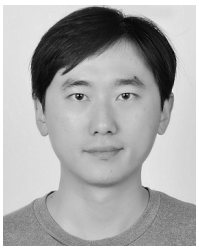
[73] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 1717–1724.

[74] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2014, pp. 818–833.

[75] J. Huang *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors," in *Proc. IEEE CVPR*, vol. 4, Jul. 2017.

[76] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3320–3328.

[77] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," 2018, *arXiv:1801.04381*. [Online]. Available: http://arxiv.org/abs/1801.04381



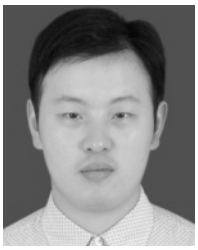
Long Chen (Senior Member, IEEE) received the B.Sc. degree in communication engineering and the Ph.D. degree in signal and information processing from Wuhan University, Wuhan, China, in 2007 and 2013, respectively. From October 2010 to November 2012, he was a co-trained Ph.D. student with the National University of Singapore. He is currently an Associate Professor with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China. His areas of interest include autonomous driving, robotics, and artificial intelligence. He has contributed more than 70 publications in these areas. He received the IEEE Vehicular Technology Society 2018 Best Land Transportation Paper Award, the IEEE Intelligent Vehicle Symposium 2018 Best Student Paper Award, and the Best Workshop Paper Award. He serves as an Associate Editor for the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS and the IEEE Technical Committee on Cyber-Physical Systems newsletter and a Guest Editor for the IEEE TRANSACTIONS ON INTELLIGENT VEHICLES and the IEEE INTERNET OF THINGS JOURNAL.

Shaobo Lin received the B.S. degree from Southwest University, Chongqing, China, in 2018. He is currently pursuing the M.S. degree with the School of Data and Computer Science, Sun Yat-sen University, China, under the supervision of Long Chen. His research interests are in the areas of computer vision and deep learning. He is recently focusing on object detection based on deep neural networks.

Xiankai Lu (Member, IEEE) received the Ph.D. degree from Shanghai Jiao Tong University in 2018. He is currently a Research Associate with the Inception Institute of Artificial Intelligence (IIAI), United Arab Emirates. His research interests include computer vision and machine learning.

Dongpu Cao (Member, IEEE) received the Ph.D. degree from Concordia University, Canada, in 2008. He is currently a Canada Research Chair in driver cognition and automated driving and an Associate Professor and the Director of the Waterloo Cognitive Autonomous Driving (CogDrive) Lab, University of Waterloo, Canada. He has contributed more than 180 publications, two books, and one patent. His current research focuses on driver cognition, automated driving, and cognitive autonomous driving. He received the SAE Arch T. Colwell Merit Award in 2012, and three Best Paper Awards from the ASME and IEEE conferences. He serves on the SAE Vehicle Dynamics Standards Committee and acts as the Co-Chair for the IEEE ITSS Technical Committee on Cooperative Driving. He serves as an Associate Editor for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, the IEEE/ASME TRANSACTIONS ON MECHATRONICS, the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, and the *Journal of Dynamic Systems, Measurement and Control* (ASME). He was a Guest Editor of *Vehicle System Dynamics* and the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS.

Hangbin Wu received the B.Sc. degree in surveying and mapping engineering and the Ph.D. degree in geodesy from Tongji University, Shanghai, China. He is currently an Associate Professor with the College of Surveying and Geo-Informatics, Tongji University. His research interests are the collection and processing of point cloud, high definition maps for autonomous driving, and the data mining from BIG navigation data, where he has contributed more than 50 publications.



Chi Guo received the M.Eng. and Ph.D. degrees in computer science from Wuhan University, Hubei, China. He is currently an Associate Professor and the Deputy Director with the National Satellite Positioning System Engineering Technology Research Center, Wuhan University. His current research interests include beidou applications, unmanned system navigation, and location services.

Chun Liu (Member, IEEE) was the Research Fellow with The Hong Kong Polytechnic University, Nottingham University and The Ohio State University. He has been the Deputy Director of the Key Lab of Advanced Engineering Surveying of Ministry of Nature Resources since 2005. He currently works as the Deputy Director of the Office of Science and Technology with the duty on the research project management with Tongji University. He is also a Professor with the College of Surveying and Geo Informatics, Tongji University, Shanghai, China. His research interests focus on LiDAR data processing, vision-based navigation, and integrated geographical data and global positioning system measurements for urban and engineering applications. As a Principle Investigator (PI), he has acquired near 20 research funds, which are mainly supported by MOST, NSFC, State 973 Plan, State 863 Plan, and Local Shanghai Government. He now occupies the stair as the PI in State key RD plan projects, the Scientific Member of the General Expert Group of the State 863 Major scientific projects of Navigation and LBS, and some academic participation in different related international academic organizations. So far, he has eight patents, nine types of copyright-protected software, six cooperation books, and more than 100 publications covering the geographic information science, integration of 3S, and imagery processing. He received the second-class award of China Ministry of Education in Science and Technology Progress in 2018, and Shanghai Science and Technology Progress in 2011.

Fei-Yue Wang (Fellow, IEEE) received the Ph.D. degree in computer and systems engineering from Rensselaer Polytechnic Institute, Troy, NY, USA, in 1990. He joined the University of Arizona in 1990 and became a Professor and the Director of the Robotics and Automation Lab (RAL) and Program in Advanced Research for Complex Systems (PARCS). In 1999, he founded the Intelligent Control and Systems Engineering Center, Institute of Automation, Chinese Academy of Sciences (CAS), Beijing, China, under the support of the Outstanding Overseas Chinese Talents Program from the State Planning Council and 100 Talent Program from CAS, and in 2002 was appointed as the Director of the Key Lab of Complex Systems and Intelligence Science, CAS. From 2006 to 2010, he was the Vice President of research, education, and academic exchanges with the Institute of Automation, CAS. In 2011, he became the State Specially Appointed Expert and the Director of the State Key Laboratory of Management and Control for Complex Systems. His research focuses on methods and applications for parallel systems, social computing, and knowledge automation. He has been elected as a fellow of INCOSE, IFAC, ASME, and AAAS. In 2007, he received the National Prize in Natural Sciences of China and was awarded the Outstanding Scientist by ACM for his research contributions in intelligent control and social computing. He received the IEEE ITS Outstanding Application and Research Awards in 2009, 2011, and 2015, and the IEEE SMC Norbert Wiener Award in 2014. Since 1997, he has been serving as a General or Program Chair for more than 20 IEEE, INFORMS, ACM, and ASME conferences. He was the President of the IEEE ITS Society from 2005 to 2007, the Chinese Association for Science and Technology (CAST), USA, in 2005, the American Zhu Kezhen Education Foundation from 2007 to 2008, and the Vice President of the ACM China Council from 2010 to 2011. Since 2008, he has been the Vice President and Secretary General of the Chinese Association of Automation. He was the Founding Editor-in-Chief (EiC) of the *International Journal of Intelligent Control and Systems* from 1995 to 2000, Founding EiC of *IEEE ITS Magazine* from 2006 to 2007, EiC of the IEEE INTELLIGENT SYSTEMS from 2009 to 2012, and EiC of the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS from 2009 to 2016. He is currently an EiC of the IEEE TRANSACTIONS ON COMPUTATIONAL SOCIAL SYSTEMS, a Founding EiC of the IEEE/CAA JOURNAL OF AUTOMATICA SINICA, and *Chinese Journal of Command and Control*.