Deep CNN-Based Real-Time Traffic Light Detector for Self-Driving Vehicles

Zhenchao Ouyang, Jianwei Niu, *Senior Member, IEEE*, Yu Liu, and Mohsen Guizani, *Fellow, IEEE*

Abstract—Due to the unavailability of Vehicle-to-Infrastructure (V2I) communication in current transportation systems, Traffic Light Detection (TLD) is still considered an important module in autonomous vehicles and Driver Assistance Systems (DAS). To overcome low flexibility and accuracy of vision-based heuristic algorithms and high power consumption of deep learning-based methods, we propose a lightweight and real-time traffic light detector for the autonomous vehicle platform. Our model consists of a heuristic candidate region selection module to identify all possible traffic lights, and a lightweight Convolution Neural Network (CNN) classifier to classify the results obtained. Offline simulations on the GPU server with the collected dataset and several public datasets show that our model achieves higher average accuracy and less time consumption. By integrating our detector module on NVidia Jetson TX1/TX2, we conduct on-road tests on two full-scale self-driving vehicle platforms (a car and a bus) in normal traffic conditions. Our model can achieve an average detection accuracy of 99.3 percent (mRttld) and 99.7 percent (Rttld) at 10Hz on TX1 and TX2, respectively. The on-road tests also show that our traffic light detection module can achieve $< \pm 1.5m$ errors at stop lines when working with other self-driving modules.

Index Terms—Traffic light detection, autonomous vehicle, deep learning, machine learning, dataset

1 Introduction

With the rapid development of sensor techniques (such as LIDAR, millimeter-wave radar, camera, and differential-GPS), HD Map and deep learning models have speeded up research on self-driving vehicles in the past decade. The most reliable solution for the management and dispatch of self-driving vehicles in the near future maybe Vehicle-to-Infrastructure (V2I) communication [1]. In a V2I communication system, the self-driving vehicle can interact with the surrounding transport infrastructure and with each other directly through the vehicular networks [2]. However, due to the lack of the V2I system, detectors for recognizing and tracking vision-based traffic lights, signs, lanes and pedestrians still play important roles in current self-driving systems or advanced driving assistant systems.

Traditional computer vision-based solutions [3], [4], [5], [6] are highly affected by the placement of cameras, change of environmental lighting conditions, distance of objects [7], and processing ability of the vehicular chip. Generally, a single set of manually set parameters based on the heuristic method is not flexible for complicated real conditions, and identification of suitable parameters is also time-consuming. Therefore, recent research interests are concentrated on using machine learning [8], [9], [10], [11], [12] or deep learning techniques [13], [14], [15], [16], [17], [18] to train the model in a data-driven way.

Furthermore, deep learning based convolutional neural network (CNN) models often achieve better performance [7], [19], [20] than the machine learning models based on Histogram of Oriented Gradients (HOG) features (such as Support Vector Machine, SVM, and Hidden Markov Model). This is basically due to the fact that convolutional layers can extract and learn more pure features from the raw RGB channels than traditional algorithm1s such as HOG. However, computation complexity of CNN models is much higher than that of most machine learning algorithms and heuristic algorithms, and special hardware is often required such as high-performance Graph card (the power of a single card is often higher than 150-250 watts). Moreover, large images are often used to capture wide views for a self-driving vehicle, and the use of binocular-camera and camera array multiplies computation complexity. This is the major bottleneck for deployment of deep learning-based traffic light detection models in low-power and low-performance vehicular computation platforms for a self-driving vehicle or a driver assistance system (DAS).

Different from previous research attempts that evaluate their own performance on the PC platforms with a high-performance graphics card [6], [14], [17], [18], we propose a real-time traffic light detection model that combines a heuristic vision-based module and a lightweight CNN classifier. The heuristic detection algorithm tries to find all candidate

- *Z. Ouyang and Y. Liu are with the State Key Laboratory of Software Development Environment, Beihang University, Beijing 100083, China. E-mail: ouyangkid@gmail.com, buaa_liuyu@buaa.edu.cn.*
- *J. Niu is with the State Key Laboratory of Virtual Reality Technology and Systems, Beijing Advanced Innovation Center for Big Data and Brain Computing (BDBC), Hangzhou Innovation Institute, Beihang University, Beijing 100083, China. E-mail: niujianwei@buaa.edu.cn.*
- *M. Guizani is with the College of Engineering, Qatar University, Doha, Qatar. E-mail: mguizani@ieee.org.*

Manuscript received 31 July 2018; revised 24 Dec. 2018; accepted 7 Jan. 2019. Date of publication 14 Jan. 2019; date of current version 6 Jan. 2020. (Corresponding author: Mohsen Guizani.)
Digital Object Identifier no. 10.1109/TMC.2019.2892451

regions Region of Interest (ROI) of traffic lights in Hue, Saturation, and Lightness (HSI) color space. The CNN model is refined to meet the requirement for available low-power ($< 10\ watt$) and low-performance on-board processing unit (NVidia Jetson TX1/TX2), and achieves 10fps processing frequency. During experiments, we test our models on both a PC platform/Server with Titan X and a vehicular platform with TX1 and TX2.

In this work, we make the following contributions:

- For real-time traffic light detection of autonomous vehicles, we combine a heuristic ROI detector and a lightweight CNN classifier model together. We also deploy our model on a mobile platform (NVidia Tx2) for on-road testing along with two self-driving vehicles (a car and a bus).
- We carefully design a lightweight CNN architecture for the classification task according to recent developments of deep learning. The architecture can achieve better performance than other models on two public traffic light datasets and on our own dataset.
- During data collection of traffic lights and evaluation of the public dataset, we find the problem of bias sampling and data imbalance in the current public dataset. To address those problems, we propose several refining strategies during data collection and preprocessing.

The rest of this paper is organized as follows. In Section 2, we first review the related work on both traffic light detection (TLD) and CNN architectures, and then summarize current challenges for TLD in the area of self-driving vehicles. We present a lightweight and real-time traffic light detector that combines the heuristic algorithm and the CNN-based classifier in Section 3. We investigate the bias in traffic light collection available in literature and propose a well-organized dataset in Section 4. In Section 5, we evaluate our model with real trace data and public data in the PC platform and on-road tests in real road conditions on the full-scale self-driving bus and car. Finally, we summarize the advantages and disadvantages of our TLD model in Section 6.

2 Related Work

In this section, we review the recent research progress in the following two aspects: the traffic light detection algorithms in self-driving vehicles, and the development of deep learning based CNN architectures. From the latter one, we can seek a solution to refine the current TLD algorithm for low-power and low-performance vehicular environments.

2.1 Vision-Based Heuristic Algorithm

Considering traffic lights are the lighting source in the outdoor environment and the blobs of traffic lights conform to traffic standards [4], early studies try to detect those unified lights according to their colors and shapes. Vision-based heuristic algorithms are the most basic and fundamental methods for traffic light detection, and the relevant procedures can be summarized as follows: 1) Mapping raw images into different color spaces (RGB [21]; Gray [3]; YUV [4], [22]; LUV [12], [23]; CIELab [24]; HSL [25]; HSV [4], [9], [10], [14]; etc.) and finding the best threshold or interval for each type of lights; 2) Finding the blob according to the shape filter [21], edge map, filled circle [22], and other morphological characteristics; 3) Fusing both color and morphological features.

Due to the simple processing logic of the heuristic algorithm, the above methods can reach a relatively high detection rate (10-100fps) [7], [19] with a single CPU. However, complex outdoor environments and safety reasons in self-driving, caused by the lens and placement of the camera, weather, and lighting, make it impossible to use the above algorithms. In addition, manual optimization of parameters is time consuming and labor-intensive.

2.2 Machine Learning-Based Algorithm

2.2.1 Traditional Machine Learning-Based Model

To automatically select and optimize the parameters from a large amount of traffic light data set, researchers try to build machine learning-based models for traffic light detection/recognition in a data-driven way. The supervised learning often uses a simple sliding window to scan each image and extract the HOG feature from each window to feed into the machine learning models (such as SVM [8], [10]; boosting [22], and tree-based model [23]).

Considering the global scan and the predefined window size of sliding window-based detection approaches usually suffer from large computation and multi-scale problems, later solutions for ROI selection, such as Adaptive Background Suppression Filter (AdaBSF) [11], template matching algorithm [9], and integrating Visual Selective Attention (VSA) [8], try to replace scanning of the global sliding window with a heuristic algorithm and to find a subset of the candidate region of traffic lights. Other improvements include using well-trained ensemble learning [10], [25] and inter-frame information for tracking [26].

The size and efficiency of traditional machine learning based TLD methods are suitable for the vehicular environment; however, the main problem is that the learning ability is still too poor to cover rich features in image processing. Moreover, valuable information may get lost in extraction of HOG features from RGB or other color spaces. The two above problems can lead to low detection accuracy in outdoor and high-speed environments.

2.2.2 Deep Learning-Based Model

In [14], the authors combine the HOG-based SVM for traffic light region detection and CNN for light state recognition. DeepTLR [18] also analyzed different configurations modified from the AlexNet [27] in dealing with traffic light data set. John et al. [13] also proposed a CNN-based TLD for saliency map generation. To deal with high-resolution images, the authors [17] deploy the YOLO [28] detecting model and stereo vision images for TL tracking.

Recent rapid development of deep learning-based TLD benefits from the following two aspects: 1) more features can be extracted from the convolution layer than HOG, and 2) the multi-layer structure of the CNN model can learn hundreds or thousands of times more feature combinations than simple machine learning models. This makes it

TABLE 1
Brief Summary of Traffic Light Detection

Categories	Paper	Dataset	Color Space	Features	Resolution	Hardware	FPS
Vision-based heuristic method	[4]	Local	YUV, HSV	Thresholds	640×480	\	275ms
	[5]	Local	RGB	Thresholds, shape	600×1000, 640×1136	\	\
	[6]	VIVA	LUV	Thresholds, clustering, Fuzzy logic	1280×960	Titan X GPU@10GHz, CPU@2.6GHz	<10ms
	[3]	VIVA	LUV	Thresholds, ACF	1280×960	\	1.275s
Traditional Machine learning-based method	[22]	Local	YCbCr	Adaboost	\	CPU@2.8GHz	15ms
	[12]	Daimler, VIVA, LaRA [29]	Gray, RGB, HSV, LUV	HOG, SVM	1920×1024, 1280×960, 640×480	CPU@3.3GHz	\
	[8]	Local	Gray, HSI	HOG, SVM	800×600	CPU@2.13GHz	142.54ms
	[11]	Local, LaRA	RGB, HSI	AdaBSF, HOG, SVM	1000×1000, 480×640	Intel i7	70ms (LaRA)
	[9]	Local	HSV	SVM	1292×1080	CPU@3.4 GHz	60-90ms
	[10]	WPI[30]	HSV	PCANet, HOS, SVM	1292×1080	CPU	300ms
Deep learning-based method	[13]	LaRA	CIELAB	CNN	480×640	\	10ms
	[18]	Local, LaRA	RGB	CNN	1280×960, 640×480	GTX980@7GHz	77ms (Local), 30ms (LaRA)
	[14]	Local	HSV	HOG, SVM, CNN	1280×800	CPU@3.6GHz, Quadro K2200 GPU@4GHz	8-15ms
	[17]	Bosch[31]	RGB	CNN	1280×720	GTX Titan Black@7GHz	<100ms

possible to learn as many potential patterns as possible from mass data, and to achieve better detection performance according to the data collected from real road conditions.

Although the above deep learning-based TLD solutions claim that they are all designed for real-time detection ($> 10 fps$ at least) in the vehicular environment; however, our investigation shows (Table 1) that most simulations are conducted on a PC platform with a high-performance graphics card in an indoor environment. Generally, the CNN model consists of millions/billions of parameters with a deep layer architecture, making it impossible to be deployed on low-performance and low-power card (such as TX1/TX2) for real-time usage. Current design of the CNN model lacks related guidance from deep learning area, which will be discussed in the later section.

2.3 Development of the CNN Architecture

Since the AlexNet [27] achieves incredible results on the ImageNet Challenge [32], deep learning-based CNN models have been deployed in a wide range of real-world applications and fundamental research, and are even better than human beings in some areas. Benefiting from the deep neural network architectures (stacked convolutional layers, optionally normalization, pooling layers, and ending up with several fully connected layers), this kind of end-to-end learning can summarize much more latent patterns than machine learning models (such as SVM, decision tree, and Bayes net).

Alex et al. [33] also find that the major components of the convolutional layers (*Conv*) and fully-connected layers (*FC*) consume 90-95 percent and 5-10 percent of the computations during model training (similar pattern appears to test), respectively. However, *FC* layers contain about 95 percent of the parameters, while the *Conv* layers contain only about 5 percent of the parameters. Therefore, researchers recommend data parallelism for *Conv* layers, while model parallelism for *FC* layers, to improve both training and running efficiency.

To improve the ability of *Conv* networks, a fundamental approach is to keep adding the layers of the network to make the network deeper [34], or use a larger kernel size [35]. However, this can lead to the problem of degradation: with the increase of network depth, stochastic gradient descent (SGD) at deeper layers can disappear, leading to higher training errors. To solve this problem, the Resnet [36] tries to add shortcut connections for cross-layer connections, and the DenseNet [37] adds direct connection to each layer.

However, these kinds of CNN models are still very large, especially for huge number of parameters calculated from those layers. Therefore, other techniques are proposed to reduce the size of current CNN models: a group of works, such as the VGG-net [38], trying to replace the large convolutional size with small ones (3×3); the GoogleNet [39], using different small size kernels (1×1, 3×3 and 5×5) and random dropouts and using a single average pooling to replace all *FC* layers; the Squeezenet [40], proposing the Fire module that consists of only 1×1 kernel to remove *FC* layers, and maintaining 50 times smaller size than the AlexNet.

It can be seen that by carefully designing the deep CNN model with those experiences, we can use smaller CNNs to solve the traffic light detection problem on the hardware with limited memory, low power, and low performance.

3 Real-Time TLD Model

In this section, we introduce our TLD model that combines the heuristic algorithm and deep learning models. We also

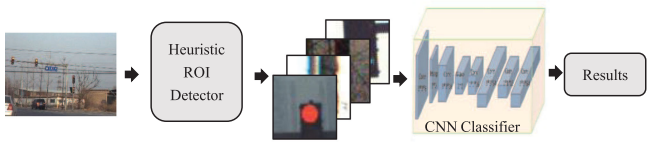


Fig. 1. Pipeline of our real-time traffic light detector: 1) a heuristic ROI detector first tries to find all possible traffic lights (including some error background); 2) a small size CNN classifier tries to identify the right class of each ROI and then gives the result.

analyze current traffic light datasets and bias of samples. We then propose a new dataset collected with HD driving map and DGPS from real road conditions.

3.1 TLD Model

For traffic light detection in the self-driving vehicular environment, there are mainly two challenges. 1) Wide angle lens are often used on self-driving vehicles to capture wide images in front of the car, and this can lead to high-resolution raw images (such as 1292×964) but larger consumption of computer resources. 2) Considering power supply and stability of both software and hardware, the traditional PC is not suitable for the self-driving vehicular environment. We design a vehicular platform–'Driving Brain'–with four NVidia Jetson TX1 and two FPGA chips. The power consumption of a single chip of the platform is lower than 10 watts (about 4 percent of the Titan X, and 11 percent of the Intel i7-7700k), and the whole device consumes less than 100 watts.

Considering that the global scan of the CNN model is inefficient, we develop a vision-based heuristic algorithm to select the candidate traffic light area, and then distinguish all possible lights and invalid regions with the CNN classifier. The whole pipeline of our detector is shown in Fig. 1.

3.1.1 *Heuristic ROI Detector*

The SSD [41] and other region proposal-based detection model will generate large amount of proposals according to pre-settings (such as the number of anchors and sizes of proposals) [42], [43], [44]. Therefore, when the resolution increases, more proposals need to be dealt with during training and testing. This is the main cause for huge GPU memory consumption, besides the complex model architecture.

Most deployments resize the input to fulfill the memory requirement, but the method is not suitable for our task of traffic light detection. Unlike moving objects of vehicles or pedestrians, the distance of traffic lights may be far away from the current vehicle, which makes the targets contain less pixels in each image. Therefore, resizing the raw images may lead to further reduction of pixels for each traffic light, and make it harder to deal with less convolution features from those pixels in the later model training and detection. We replace the region proposal with a heuristic detector, which can find about 30 candidate regions with fixed size. This helps us solve the problems regarding both memory consumption and processing time. This can be accomplished because traffic lights are standard illuminants, and by controlling the exposure of a camera, the pixel values of the lights can have fixed characteristics.

In addition, we use the HSV color space (logarithm coordinate), which is more similar to the human visual system, instead

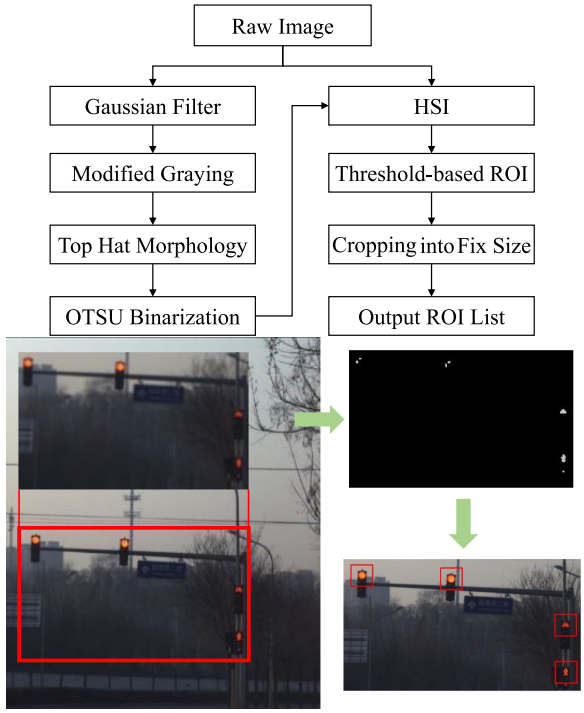


Fig. 2. Flowchart of the heuristic ROI detection algorithm using traditional vision-based modules. At the bottom is an example: through a series of processing, we get all possible pixels from the binary image. By transforming these pixels into the HSI color space, we get all possible ROIs and crop them into predefined sizes.

of the RGB color space (Cartesian coordinate). Therefore, the heuristic ROI detector can be more efficient than the region proposal method, and generate much less RPs per image.

The purpose of this module is to find all ROIs that include both traffic lights and invalid background, and the processing flowchart is illustrated in Fig. 2 along with an example of detecting the results. We combine together the commonly used image processing modules, such as Gaussian Filter, Top Hat Morphology, OTSU algorithm, and HSI transformation. The left processes in Fig. 2 try to select possible lighting sources from each image. The Gaussian Filter is a basic preprocessor used to remove the noise in each image, and then we transfer the image into the gray map and use Top-Hat to dilate the lighting area (Morphological transformation). Finally, the OTSU transfers the image into the binary map, and all the illuminants are retained. The right workflow tries to select the remaining areas that satisfy the requirement for pixel values in the HSI color space, and crops the areas into fixed ROIs. This also shortens the processing time and can get a series of ROI that may contain illuminants. However, the model proposed in this study lacks the use of the texture and contour information of traffic lights, which we plan to improve in the future.

We find that the colors of traffic lights are more robust against illumination variation and are more suitable for outdoor environments in the HSV/HSI color space than in the RGB color space, and the HSV color space has been widely used in the area of TLD [4], [8], [9], [10], [11], [12], [14].

Considering that the HSI transform is time-consuming, we use the left process (in Fig. 2) to improve the efficiency. Only the pixels in the binary map generated from OTSU Binarization with a value equal to 1 will be transformed.

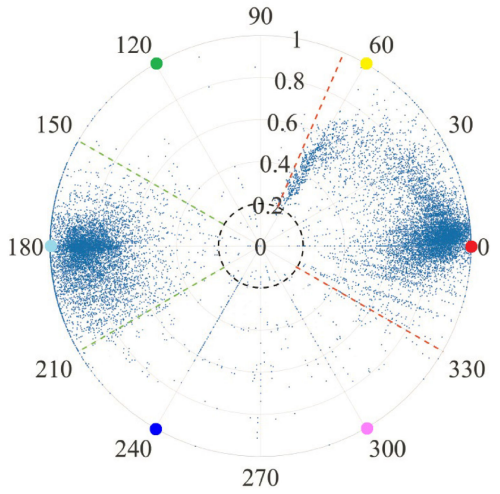


Fig. 3. By mapping the labeled raw traffic light regions (RGB channel) into chromaticity plane (HSI channel), it is easy to find the thresholds for H ($\in [0, 360]$) and S ($\in [0, 1]$) of the pixels of the traffic light. The green traffic lights are mostly distributed between 150-210 and the red/yellow traffic lights are mostly distributed between 0-65 and 330-360. Some of the noise points are the mistakenly marked boundaries.

This process can remove most of the non-lighting source background and candidate lights that are too small, and thus speed up the process.

Following that, we try to find possible candidate lights according to the HSI value of each pixel with a similar strategy in [4]. We map the labeled region of the blob (RGB channel) into the chromaticity plane (HSV channel), and select the best region to cover all possible pixels as shown in Fig. 3. The final parameters are as shown in Eq. (1). In this process, invalid background, such as the taillight of front vehicles, red/blue flag, advertising boards, etc., will be involved in the candidate ROI.

$$\begin{gathered} S > 0.2 \\ Red : 0 < H < 65 || 330 < H < 360 \\ Blue : 150 < H < 210 \end{gathered} \tag{1}$$

In the end, we crop each ROI into a fixed size for later CNN models, and set the size to cover the blob instead of the whole light for the following two reasons: 1) the arrangement of blobs may differ from city to city; therefore, it is better to detect the state of each blob rather than the whole light; 2) smaller images mean higher processing rate, and we need only limited background information for the training of the CNN model. In our case, after testing our dataset collected with an AVT camera and 16mm lens, we set the ROI size as $56 * 56$ (this value may be affected by hardware parameters, especially by the focal length of the lens and placement of the camera).

In general, the operations, Gaussian Filter, Modified Graying and Top Hat Morphology in the heuristic ROI detector are of lower time complexity than $O(n)$. The 3D OTSU Binarization is of time complexity of $O(L^3)$, where L is the number of gray levels (L is 8 in our case). The rest operation can be ignored, because the output of OTSU is only a very small subset of the original image (as shown in the top left in Fig. 2).

TABLE 2
Architecture of the Real-Time Traffic Light Detector (RTTLD)

		RTTLD	mRTTLD
Input		56*56*3	
1	Conv	3*3*16	3*3*8
2	Maxpool	2*2,stride=2	2*2,stride=2
3	Conv	3*3*32	3*3*16
4	Maxpool	2*2,stride=2	2*2,stride=2
5	Conv	1*1*16	1*1*8
6	Conv	3*3*128	3*3*32
7	Conv	1*1*16	1*1*8
8	Conv	3*3*128	3*3*32
9	Maxpool	2*2	2*2
10	Conv	1*1*32	/
11	Conv	3*3*256	/
12	Conv	1*1*32	/
13	Conv	3*3*256	/
14	Maxpool	2*2,stride=2	/
15	Route	[10,14]	[5,9]
16	Conv	1*1*n	1*1*n
17	Avgpool	n[a]	n
18	Softmax	n	n

[a] *n is the number of classes in classification tasks.*

3.1.2 The CNN Classifier

As we have a list of ROIs that contains both traffic light candidates and invalid backgrounds from each frame, we design a deep convolutional neural network classifier. The architecture of the network is illustrated in Table 2. Different from the previous deep learning-based TLD that directly deploys a CNN model, we try to improve our model with the experience mentioned in Section 2 (B). We choose the Darknet [45], a pure C language environment with CUDA, as our deep learning framework.

As shown in Table 2, the CNN model for the Real-Time Traffic Light Detector (RTTLD) uses the following four tricks to reduce the model size and speed up the processing efficiency. 1) Small kernels of 1×1 and 3×3 are used to reduce the parameters in each layer [39], [40]. In addition, the incorporation of 1×1 *Conv* layers can increase the nonlinearity of the decision function [38] and can extract cross-channel features rather than inner-channel features from the common *Conv* layer (i.e., $n \times n$). 2) The 5-8 and 10-13 blocks obtained by stacking each 1×1 *Conv* layer with a 3×3 *Conv* layer are used to imitate the Fire module of the SqueezeNet [40]. Different from the 3×3 convolution filter, the 1×1 convolution filter can generate cross-channel features, helping to reduce the model size and mitigating the problem of the vanishing gradient. 3) The fully-connected layers are replaced with average pooling (*Avgpool*) from the end of the network. This operation helps us reduce 90 percent of the model size, and thus makes it possible to deploy our model in the resource-limit device. 4) The route layer is used to combine different scales of features extracted from different depths of layers. The total RTTLD model contains less than 18 layers, and we also design a micro edition (mRTTLD) with fewer layers and parameters in each layer. Both RTTLD and mRTTLD models can reach an average classification accuracy on a real dataset.

Considering our detection mission consists of identification of no more than 20 classes of traffic lights (much less than the ImageNet challenge [32] and COCO [46]). In

TABLE 3
Statistic of Current Public Datasets: Sample Imbalance

Class \ Dataset	LaRA	[11]	WPI	VIVA	Bosch	Local
go	3381	2713	6697	10000	5207	3632
stop	5280	5245	6035	13487	3057	3997
go_arrow	\	\	646	\	\	\
left_go	\	\	1514	371	178	216
right_go	\	\	518	\	\	\
left_stop	\	\	2164	6178	1092	1079
red:green	1.56:1	1.93:1	0.87:1	1.89:1	0.52:1	1.32:1

addition, we crop the candidate ROI into 5,656 images in the first stage. Both of the above conditions help us reduce the number of layers and make the final model size less than 182 KB (mRTTLD) and 818 KB (RTTLD).

3.2 Collection of Traffic Light Dataset

3.2.1 Biased Sampling in Traffic Light Dataset

The problem of the imbalanced dataset has long been studied in data mining. Recently, Zhao et al. [47] also found that the models trained on those datasets will further amplify the existing bias. Without properly quantifying and reducing the reliance on such correlations, we may get incompletely trained models, and broad wide adoption of these models may lead to serious problems in real-world applications, especially in self-driving.

However, the imbalance of traffic light dataset has never been studied before. In our investigation of several public traffic light datasets (VIVA [48], Bosch [31], WPI [30] and LaRA [29]) and data collected on our own self-driving vehicles, we find a series of implicitly biased sampling that may lead to an imbalanced dataset, and summarize the biases as follows:

- Bias is caused by traffic rules: during data collection, vehicles have to stop during the red light, and pass the intersection when the light is green; this may lead to more red light samples.
- The intersection types of a collecting route have different kinds of traffic light, for example, a T-junction may not have a left-turn light; this may influence the distribution of the light types we collected.
- The pre-set duration of each light of different intersections maybe not the same. For example, the duration of red light is often longer than green light in most crossroads in China.
- During manual annotation, inter-observer variations may lead to different labeling results, such as the color, the arrow type, and the effective size of a light.
- For semi-automatic annotation, the setting of the thresholds (such as the values of the color space, the size of the bounding box, etc.) may lead to different outputs.

Table 3 gives a statistical analysis of the public datasets of traffic lights. It is obvious that there exist different varying degrees of sample imbalance in each dataset. The last column is the ratio of total red lights to total green lights, and the ratio can reach nearly 2:1 in VIVA and the dataset [11]. On the other hand, the number of 'left_go' in VIVA, Bosch, and Local (this dataset is arranged during our previous collection) is too small, with only several hundred in each set. Sample imbalance widely exists in previously published datasets.

3.2.2 Fine Tuning the Datasets

Due to the data-driven based training for deep learning models, less samples mean under-trained (or under-fitting) model on related classes, and may lead to amplification of biases [47] in later testing and real-world deploying. Different from the studies that aim at improving accuracy, we also consider how to reduce the above biases during data collection and then generate a well-organized dataset for model training as much as we can by using the following methods: 1) a well-designed route that contains different kinds of traffic lights; 2) maintenance of similar length of time in collecting data of each type of traffic lights; 3) manually balancing of different samples for the dataset; 4) confirmation of the final data annotation by at least three human markers.

4 Experimental Study

4.1 Traffic Light Datasets

As we investigate the literature for this kind of research and public datasets, we find that four public datasets are available from the Internet: LaRA, VIVA, WPI, and Bosch. Our testing only considers the WPI dataset, VIVA dataset, and our own dataset, for the following two reasons: 1) the image pixel (640*480) is too low and there are out-of-focus images in LaRA; 2) the targets (different lights) in the Bosch dataset are too small to be distinguished by humans. We make minor adjustment of each dataset to fit our traffic light detection algorithm.

4.1.1 WPI

The WPI traffic light dataset (WPI) is a public dataset published by the researchers from the Worcester Polytechnic Institute, and are collected in Worcester, MA, the USA during summer and winter. The original dataset contains three parts: the training data holder (Holder), training data optional (Trainset) and test data (Testset), and the images at a resolution of 1,920*1,080. According to their introduction, the Holder contains 10,034 images of 10 classes of traffic lights; however, the Trainset and Testset contain only 1,321 images and 2,159 of 6 classes of traffic lights, respectively.

Considering the small size of the Trainset and the Testset, we combine them as the test set, and use the Holder for training. In addition, only the 6 classes of traffic lights in both Trainset and Testset are considered.

4.1.2 VIVA

The VIVA traffic light detection benchmark (also called LISA) is the only public dataset that contains the traffic light collected during both day and night at urban areas of La Jolla and Pacific Beach in San Diego, California, USA. Totally, there are 6 classes of lights on either 21,452 daytime or 14,297 nighttime images at a resolution of 1,280*960. Currently, we only use the images of daytime and merge the Yellow and Red as one class–'stop'.

Fig. 4. Placements of camera on our self-driving vehicles: car (left) and bus (right).

Fig. 5. Examples of mistakenly detected ROIs.

4.1.3 YBY

This dataset is collected from the urban area of Beijing with an AVT Moke G-125c camera and 12-36 mm Lens (as shown in Fig. 4) equipped on our self-driving Ibus (Yutong bus) and car (Changan RAETON) from autumn of 2017 to spring of 2018. We collected altogether 6 classes of traffic lights of 30,886 images during the daytime with a resolution of 1,292*964. Except for common traffic lights, we also consider the lights for pedestrians and cyclists as viable supplemental detection targets when the main traffic lights are occluded or missing from the camera view. This dataset is also fine-tuned according to the methods mentioned in Section 3(B).

During data collection, we find that besides the common conditions (such as lighting, weather and the orientation of the camera), the lens, distance of intersections and placement of the traffic lights also highly affect the traffic light samples. For the only controllable element of lens, the short lens offers a wide field of view but fewer pixels per object. In contrast, the long lens offers a narrow field of view but more pixels per object. Therefore, when designing the vision-based applications for the autonomous vehicle, we need to guarantee both large view filed and clear pixels of objects. That is why we choose the dataset with large pixels for testing, and we recommend users to adjust the lens according to their own needs.

4.2 Evaluation of the Heuristic ROI Detector

Some of the related ROI detectors separate red from green lights according to the pixel value of H channel, and deal with the lights with different process modules. In our system, we treat ROIs that contain different kinds of lights as the same; therefore, we can evaluate the precision of our ROI detector on the three datasets mentioned above. In addition, the main shortage of the threshold-based heuristic ROI detector is that it cannot distinguish the objects with the same HS values as traffic lights, such as flags, car lamps, and other objects with red or green covers as illustrated in Fig. 5.

In Table 4, we evaluate the performance of Heuristic ROI detector. We first classify the detected results into two classes: 1) light, the ROI that contains all possible traffic lights, and 2) non-light, the ROI that contains only a background.

TABLE 4
Measurement of the Heuristic ROI Detector

	VIVA		WPI		YBY	
	Train	Test	Train	Test	Train	Test
TP	95.4%	96.6%	98.1%	98.7%	99.3%	99.6%
FN	4.6%	3.4%	1.9%	1.3%	0.7%	0.4%
FP	54.3%	33.3%	8.9%	8.1%	43.7%	29.9%

We then use the following three metrics to measure our heuristic ROI detector: 1) True Positive (TP), the percentage of ROI of the lights detected by the detector; 2) False Negative (FN), the percentage of the traffic lights that are not detected by the detector; 3) False Positive (FP), the percentage of mistakenly detected non-light.

It can be seen that the detector shows a slightly lower TP (95.4 and 96.6 percent) and higher FN (4.6 and 3.4 percent) and FP (54.3 and 33.3 percent) on VIVA dataset, and also high FP on YBY. As we investigate the raw images, we find that the intersections in VIVA are closed; therefore, the camera can capture both the traffic lights of current and next intersections. However, the traffic lights in the next intersection are too small and will be ignored by the heuristic detector. In WPI, larger image pixel of 1920×1080 can reduce this kind of mistakes. For the YBY dataset, we use HD driving map and only trigger the collection of about 30 meters before each intersection. The value of FP highly relies on the conditions during data collection. For example, the heavy traffic flow can lead to detection of non-lights from other vehicles (VIVA and YBY), and other red/green objects in the environment (YBY).

A low TP or a high FN means that the detector may miss some of the traffic lights in the scene. This often occurs when the target is too small and can be conquered by utilizing the HD map information and DGPS equipped on self-driving vehicles. High FP rate is highly affected by surrounding road conditions and will lead to an extra-processing cost of the later CNN classifier. We will improve the current heuristic condition to filter non-light ROI.

We also compare the number of the region proposals per image of our method with that of related region proposal based detection models, the results are listed in Table 5. As the VIVA and YBY dataset have similar resolutions, only the VIVA (1280*960) is listed; when dealing with the WPI dataset with an extreme high resolution of 1920*1080, the SSD and Faster RCNN are too big to be deployed on TX2. It can be seen that the SSD, Faster RCNN and Yolo2-tiny generate much more PRs/images than Yolo3-tiny and our Heuristic ROI Detector. One reason is that their aim is to detect the best locations of each object, and will generate amount of proposals for each anchor. We ignore accuracy and generate a much larger proposal (ROI) to cover each potential area, which speeds up the whole detection process. The only drawback of our heuristic ROI detector is the detected ROI changes dynamically for we only use color information and ignore the texture information.

4.3 Evaluation of the CNN Classifier

After candidate ROI detection, we have a group of small ROI from the raw images collected from the camera. We

TABLE 5
Evaluation of the Region Proposal

Models	Input resolution	RPs/image
SSD	1920*1080	-
	1280*960	100,000+
	608*608	34,000
Faster RCNN	1920*1080	-
	1280*960	10,000+
	608*608	2000+
Yolo2-tiny	1920*1080	5220
	1280*960	3094
	608*608	949
Yolo3-tiny	1920*1080	425
	1280*960	280
	608*608	132
Heuristic ROI Detector	1920*1080	39
	1280*960	32

TABLE 6
Reorganized Dataset for Evaluation of CNN Models

Class	VIVA		WPI		YBY	
	Train	Test	Train	Test	Train	Test
goa	\	\	250	396	\	\
gol	262	116	766	748	403	186
gor	\	\	440	78	\	\
goc	7,080	2,909	3,662	3,340	6,517	2,796
stopl	4,274	1,904	1,741	423	1,005	401
stopc	7,720	4,060	3,778	2,257	5,509	2,589
gped	\	\	\	\	3,363	1,442
rped	\	\	\	\	4,515	1,937
others	1,718	727	5,774	2,412	9,309	3,995

The meaning of each class is listed as follows: goa, green up arrow light; gol, green left arrow light; gor, green right arrow light; goc, circular green light; stopl, red left arrow light; stopc, circular red light; gped, green pedestrian light; rped, red pedestrian light; and others, backgrounds.

then use a CNN model to identify all possible traffic lights from the candidate ROIs. To evaluate our model of **R**eal **T**ime **T**raffic **L**ight **D**etector (*Rttld*), we compare it with both well-known CNN models (such as the *AlexNet*, *DenseNet*, *GoogleNet*, and *ResNet*) and related CNN models (*Deeptlr* and *Karsten*) designed for traffic light detection.

We reorganize the raw dataset of VIVA, WPI and our YBY for the task of classification, and label the mistakenly detected images (from the heuristic ROI detector) of the background as others. Details of classes of traffic lights of the three datasets are listed in Table 6. It can be seen that the VIVA, WP, and YBY contain 5, 7 and 7 classes of traffic lights, respectively. Considering the traffic lights in each dataset are different, we separately train each model on three different datasets.

All model training and evaluation are conducted under the deep learning framework of *Darknet* on a GPU server with Intel Core i5-65003.2 GHz, 32G RAM and two NVidia Titan X. The training parameters for the seven models are also unified: 800,000 images are taken as the max batches (256 images for a single batch), the starting learning rate of the stochastic gradient descent (SGD) is set to be 0.01, and the polynomial rate decay with a power of 4, weight decay of 0.0005 and momentum of 0.9 are used.

4.3.1 Gini Coefficient Index

As mentioned above, existing datasets for traffic light detection do not consider sample balance. This may lead to a series of problems of biased sampling and thus influence the training and evaluation of the data driving-based CNN model. To overcome the problems of biased sampling, we first introduce the *Gini* coefficient index [49] to evaluate the inequality of a dataset for deep learning. The *Gini* coefficient index (G) is shown in Eq. (2), where n is the number of object classes in current dataset, and y_i is the sample size of class i. Considering the class indexed in the non-decreasing order ($y_i \leqslant y_{i+1}$), we first sort the value of class and fill 0 for nonexistent classes for each dataset.

$$G = \frac{1}{n}\left(n + 1 - 2\left(\frac{\sum_n^{i=1}(n+1-i)y_i}{\sum_n^{i=1} y_i}\right)\right). \quad (2)$$

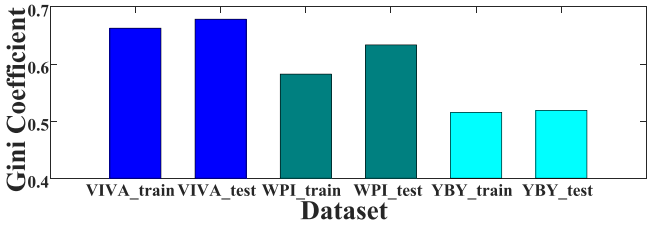


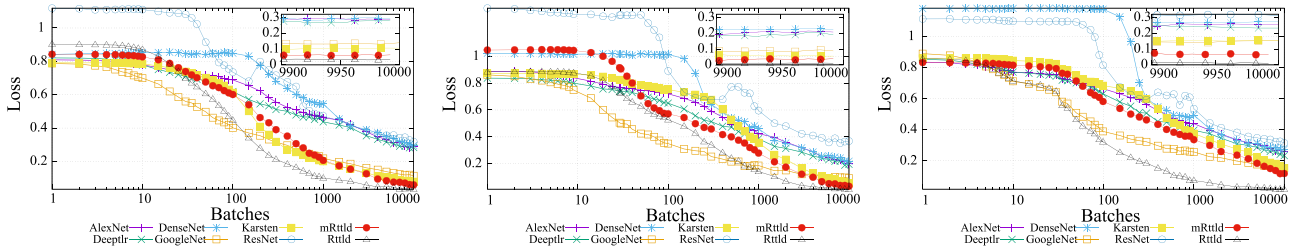
Fig. 6. Gini coefficient indexes of the datasets of VIVA, WPI, and YBY.

Because the number of each class is non-negative, the *Gini* coefficient can theoretically range from 0 to 1: 0 means perfect balance of classes in a dataset; in contrast, 1 means the maximal inequality. It can be seen that both the training and testing sets of the YBY dataset have the lowest *Gini* efficient index than those of other datasets. However, the *Gini* coefficient of 0.5 is still high, and we are still looking for a strategy to mitigate the problem of biased sampling.

4.3.2 Model Evaluation

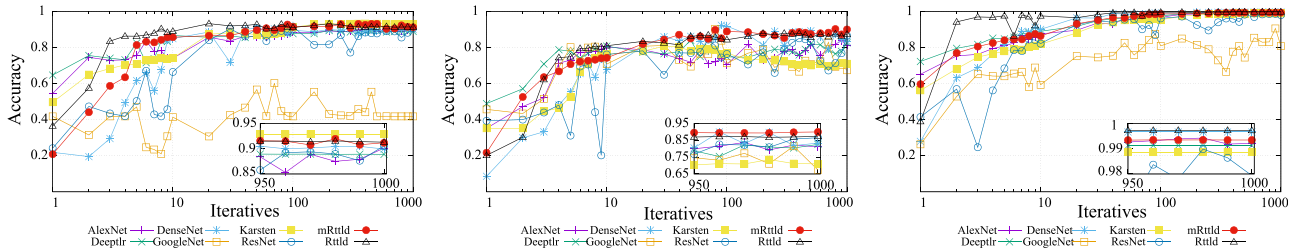
Fig. 7 illustrates the loss of each model during the training phase (first 100,00 batches) for VIVA (Fig. 7a), WPI (Fig. 7b) and YBY (Fig. 7c). It can be seen that the losses of most of the models begin to drop quickly after 100 rounds of training, and except for the ResNet, the losses of other models drop very smoothly on all of the three datasets. Among the models, the Rttld drops the quickest and can reach nearly 0 at 10000 batches, followed by the mRttld, GoogleNet, and Karten, while the rest of other models maintain a very high loss at the end of training. There is an exception as shown in Fig. 7b: the GoogleNet first has a quicker drop during the training, and several hundred rounds later, the Rttld begins to have a slower drop and reach the bottom on all the datasets after 100,00 batches. In deep learning, soft and smooth losses during training means the model is well trained, while the loss of ResNet jittered during the first 1,000 batches means the model is not well trained. The losses reach only 0.356, 0.369 and 0.314 at the 10,000-batch on VIVA, WPI and YBY dataset, respectively, meaning that the ResNet is not well trained on all the three datasets.

We also save the first 1,000 rounds of models (a model is saved every 200 batches) during the training phase and



(a) VIVA dataset (b) WPI dataset (c) YBY dataset

Fig. 7. Loss of each model during the training on the dataset of VIVA (a), WPI (b), and YBY (c). The log-coordinate for x-axis shows the losses during first 10,000 batches.



(a) VIVA dataset (b) WPI dataset (c) YBY dataset

Fig. 8. Classification accuracy of different models on the dataset of VIVA (a), WPI (b), and YBY (c). We train each model for 1,000-iterative cycles and save the weights for testing.

evaluate their performance on the corresponding Testsets. As shown in Fig. 8. We can see from Fig. 8a to Fig. 8c that almost all the models can reach a relatively stable accuracy after 100 rounds (i.e., 20,000 batches of training). For the VIVA dataset (in Fig. 8a), it can be seen that all models can finally reach about 90 percent accuracy except the GoogleNet (only 41.7 percent with severe vibration). The ResNet also jitters and sometimes the accuracy of the model can drop to 53.6 percent, meaning that both the ResNet and GoogleNet are not well trained and cannot maintain a stable performance. The rest of other models, e.g., AlexNet (87.9 percent), DenseNet (89.9 percent), Karsten (92.8 percent), mRttld (92.4 percent), Rttld (91.2 percent) and Deeptlr (88.8 percent), are very stable and can achieve about 90 percent accuracy. The Karsten, mRttld and Rttld show a very stable performance after 100 iterative, which also means that the three models can have a faster convergence during training than the other models.

In Fig. 8b, all models show less stability and lower accuracy on the WPI dataset than the performances they achieved on the dataset of VIVA before, and some of the models even declined slightly in the later batches. The Karsten (68.1-73.5 percent) and the GoogleNet (67.4-80.7 percent) fall behind all the other models. From the subgraph in Fig. 8b, the last 100 models of mRttld (87.2-91.2 percent), Rttld (86.3-87.4 percent) and DenseNet(81.8-90.4 percent) perform nearly the same. The rest of other models can also achieve accuracies ranging from 70 to 80 percent. One main problem is that the WPI dataset contains 7 different traffic lights, and the related samples (i.e., goa, gol and gor in Table 6) are too small and similar to lead to greater classification challenges. Moreover, from the distribution of samples in Table 6 and the high *Gini* coefficient indexes of the WPI training set and test in Fig. 6, the samples of different classes of traffic lights show a heavy imbalance, which means that some kinds of traffic lights are under-fitting during training, and may cause the drop of average accuracy.

On the YBY dataset which also contains 7 classes of traffic lights (in Fig. 8c), the GoogleNet and ResNet also perform the worst, but slightly better than when they perform on VIVA and WPI. The remaining five models are very stable and reach about 99 percent accuracy after the 100th model (i.e., 20,000 batches of training). Among them, the Rttld (99.7 percent) performs the best, followed by DenseNet (99.6 percent), mRttld (99.3 percent), AlexNet (99.2 percent), deeptlr (99.1 percent) and Karsten (98.8 percent). All models achieve better performance on the YBY dataset than on WPI and VIVA, and this benefits from the DGPS and HD map used during camera image collection. The camera is triggered only when the vehicle is near road crossings. Therefore, we ignore the invalid frames during driving.

4.3.3 Model Features

To deploy the models on the mobile platform of Tx1/Tx2 for an autonomous vehicle, the model size should be also considered. In Table 7, we list several model features. It can be seen that the size of the 13-layer mRttld model is 182 KB, only 1/463 of the AlexNet and 1/1,131 of the GoogleNet. The larger version of the 18-layer Rttld model is 819 KB. Although the size of Karsten is also very small (640 KB), it can reach very low accuracy of about 70 percent (Fig. 8b) when dealing with the WPI dataset. This is extremely dangerous for autonomous vehicles. In terms of size and performance of the model, both mRttld and Rttld outperform the other models on the WPI dataset. In addition, mRttld, Rttld and Karsten all work very well on the VIVA dataset and YBY dataset.

We also list the Floating Point Operations (FPO) of those models. When dealing with the VIVA/YBY and WPI

TABLE 7
Details of the CNN Models

Models	Size	Layers	FPO[a]	
			VIVA/YBY	WPI
AlexNet	82.4MB[b]	14	146Mn[c]	165Mn
Densenet	59.8MB	304	430Mn	665Mn
Karsten	640KB	**11**	**5.4Mn**	**8.4Mn**
Deeptlr	90MB	14	157Mn	176Mn
GoogleNet	201MB	26	477Mn	930Mn
Resnet	79.3MB	69	541Mn	608Mn
mRttld	**182KB**	13	25.6Mn	33.6Mn
Rttld	819KB	18	41.2Mn	53.8Mn

[a]*Floating Point Operations.*
[b]*MB=Mega Byte.*
[c]*Mn=Million.*

datasets, the input images are 56*56 and 64*64, respectively, leading to higher model FPO on WPI. In addition, the number of the classification targets (5 classes in VIVA, 7 classes in WPI and 7 classes in YBY) also slightly influences (only the last *softmax* layer) the FPO of the models; however, this shows slight differences between VIVA and YBY. It can be seen that the model size only reflects the final number of weights stored in the model, the total FPO mainly affected by the layered architecture and the operations (pooling, convolution, route, etc.) in each layer.

Other models, such as the DenseNet, Deeptlr and AlexNet, also perform very well in processing different dataset; however, their sizes are too big to be deployed on the mobile platform of vehicles, and their FPOs are also very high. We will analyze their time consumption in the following section.

4.3.4 Time Consumption

In the design for a less powerful self-driving platform, we should also consider the time consumption/processing efficiency of a model. Therefore, we evaluate our heuristic detector, all the CNN models and dataset on a GPU server and our driving brain platform of both Tx1 and Tx2 modules.

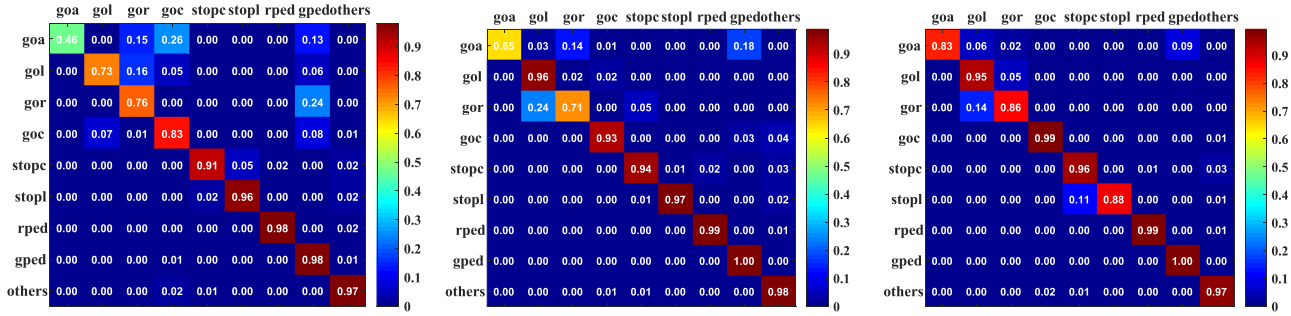
Each time, we randomly select 1000 images from the Testset to test the CPU-based detector and the CPU-based classifiers (for each classifier we only use the last trained model) for 10 times. We calculate the average and standard deviation of time consumption for each model on different hardware platforms, as shown in Table 8. It can be seen that the detector can have an overwhelming advantage over the GPU server, with only 1/3 of time consumption of the ARM-based Tx1/Tx2. The improvement of our detector on Tx2 is not so remarkable (about 20 percent) than on Tx1, and the WPI dataset takes the longest time to process the high resolution dataset of 1920*1080 images. In general, the detector can achieve averagely 37FPS, 26.8FPS and 52.6FPS on the GPU server for VIVA dataset, WPI dataset, and YBY dataset, respectively. For the mobile platform, the detector can handle the VIVA dataset (12FPS for Tx1 and 14FPS for Tx2) and YBY dataset (14FPS for Tx1 and 16FPS for Tx2) at a performance higher than 10FPS (a basic time requirement for most self-driving systems). However, both modules fail to reach 10FPS when dealing with the WPI dataset, and we still need to seek solutions for handling extremely high-resolution images by using parallelization on GPU.

For the time consumption of different CNN classifiers, it can be seen that the time of processing WPI dataset is generally longer than that of VIVA and YBY dataset. Besides high-resolution of raw images and bigger ROI ($64 * 64$) in WPI, the detector also outputs more ROI candidates from each image and thus leads to lower classifier FPS. Generally, the GPU server can reach 2-5 times higher performance and 4-10 times higher performance than the Tx2 and Tx1, respectively. Meanwhile, the mRttld and Rttld models achieve the 1st and 2nd performances for all settings. It can be seen easily from Table 7 that the Karsten has fewer layers and lower FPO than the mRttld. However, the Karsten uses large kernel size (7*7) for convolution and 3 fully connected layers, leading to lower parallel processing efficiency.

In summary, when using the high-performance GPU server, our two-phase traffic light detector can easily achieve enough FPS for autonomous vehicle requirement: 1) detector+mRttld, 34.4FPS on VIVA, 23.2FPS on WPI and 51FPS on YBY. 2) detector+Rttld, 31.6FPS on VIVA, 19FPS on WPI and 49.8FPS on YBY. However, the Rttld+detector

TABLE 8
Evaluation of Processing Time on Different Hardware Platforms

			GPU Server			Tx1			Tx2		
Platform	*CPU*		Intel Core i5-6500 3.2GHz			Quad ARM A57/2 MB L2			HMP Dual Denver 2/2 MB L2 Quad ARM A57/2 MB L2		
	GPU		GTX Titan X 12G (Maxwell) 3840 CUDA cores			NVidia Pascal 256 CUDA cores			NVidia Maxwell 256 CUDA cores		
	Memory		32GB			4 GB 64 bit LPDDR4 25.6 GB/s			8 GB 128 bit LPDDR4 59.7 GB/s		
	Dataset		VIVA	WPI	YBY	VIVA	WPI	YBY	VIVA	WPI	YBY
Evaluation	***Detector***		26.7 (0.2)	37.3 (0.2)	19.0 (0.2)	82.4 (2.5)	131.4 (4.1)	71.3 (3)	69.5 (0.8)	112.7 (0.4)	61.4 (0.1)
	CNN Models	AlexNet	6.8 (0.1)	17.4 (0.2)	1.7 (0.1)	21.5 (0.4)	26.7 (0.2)	16.6 (0.7)	17.8 (0.2)	17.5 (0.2)	14.5 (0.4)
		DenseNet	174.5 (0.7)	404.6 (0.2)	45.6 (0.1)	304.8 (0.3)	689.5 (0.6)	173.4 (0)	153.7 (0.1)	335 (0.2)	87.5 (0.2)
		Karsten	6 (0.1)	7.1 (0.1)	5.7 (0.2)	19.1 (0.5)	33.3 (0.2)	17.6 (0.6)	14.4 (0.3)	24.8 (0.2)	14.7 (0.4)
		Deeptlr	10.2 (0.1)	23.5 (0)	2.4 (0.3)	33.5 (0.3)	32.5 (0.2)	27.9 (0.7)	28.9 (0.3)	29.5 (0.2)	25.2 (0.4)
		GoogleNet	39.1 (0.1)	89.2 (0.1)	9.7 (0.3)	51.9 (0.4)	122.1 (0.4)	30.8 (0.7)	37.7 (0.4)	84.2 (0.2)	22.1 (0.5)
		ResNet	40.3 (0.2)	95.5 (0)	9.8 (0.2)	77.8 (0.3)	182.6 (0.4)	43.3 (0.5)	48.5 (0.1)	110.2 (0.1)	27.5 (0.2)
		mRttld	**2.4 (0)**	**5.7 (0)**	**0.6 (0.1)**	**10.8 (0.3)**	**25.8 (0.2)**	**5.3 (0.2)**	**5.4 (0.1)**	**13.9 (0.1)**	**2.5 (0.1)**
		Rttld	4.9 (0.1)	15.3 (0)	1.1 (0.1)	16.3 (0)	41.9 (0.3)	8.5 (0.3)	8.9 (0.1)	21.1 (0.1)	3.9 (0.2)



(a) Karsten (b) mRttld (c) Rttld

Fig. 9. Confusion matrix heatmaps of Karsten, mRttld, and Rttld on the merged testing set.

can achieve 10.7FPS on VIVA and 13FPS on YBY, the mRttld +detector can achieve 13.3FPS on VIVA and 15.6FPS on YBY on Tx2, respectively.

Although the model size, deep layers, and higher FPO often lead to shorter processing time, the structure of the model is the most important factor. For example, the GoogleNet model is 201MB with 477Mn FPO on VIVA, but it has lower time consumption than the DenseNet (only 59.8 MB with 430 Mn FPO). Therefore, we design the Rttld and mRttld models with a reasonable structure to guarantee that the accuracy, size and processing time of the model can satisfy autonomous vehicle hardware at the same time.

4.3.5 *Merged Dataset Testing*

Considering that the dataset contains different types of traffic lights, we merge the images from WPI, VIVA, and YBY as one union set to evaluate the model performance. Therefore, the merged set will contain 9 different classes of traffic lights and background from different places. In addition, according to the model size, the average accuracy and other metrics analyzed above, the mRttld, Rttld, and Karsten are selected to be trained and tested on the merged dataset. All the three models trained use the same configuration used before. Different from average accuracy that only evaluates the overall precision of a model, we plot the confusion matrix as heat maps (Fig. 9) to show the detailed classification precision of each class of the models, with the red box meaning higher accuracy and blue box the contrary.

Each row of the matrix represents the instances (percentage) in a predicted traffic light, while each column represents the instances in an actual traffic light. The numbers on the main diagonal are the correctly classified results, while the rest are the mistakenly classified results, as illustrated in different colors. The Rttld model shows better accuracy for the classes of 'goa (arrow of go)' and 'gor (circle of go)' than mRttld, and only slightly worse when dealing with 'stopl (left arrow of stop)'. Both of the small models (mRttld and Karsten) perform worse when dealing with the merged dataset that contains more types of traffic lights. Therefore, we prefer to use Rttld in complex road conditions with more kinds of traffic lights. When dealing with closing scenes, such as scenic areas and local areas, the mRttld is also stable enough.

TABLE 9
Comparison of Different Detection Models

Model	Resolution	FPS	IOU	Recall
Yolo2	608*608	4.7	27.21%	15.15%
Yolo2-tiny	1280*960	3.9	27.65%	19.03%
	608*608	11.5	22.55% ↓	13.29% ↓
Yolo3	608*608	1.96	19.9%	6.9%
Yolo3-tiny	1280*960	5.2	38.13%	**32.47%**
	608*608	**14.5**	17.71% ↓	5.59% ↓
SSD	608*608	12.4	11.37%	3.3%
Faster RCNN	608*608	3.5	14.77%	7.3%
detector+Rttld	1280*960	**10.6**	**40.45%**	31.4%

4.4 Comparison with Other Detection Models

We compare our detector+Rttld model with other region proposal based detection models on the NVidia Tx2 platform with the VIVA dataset (dayClip1-10 for training and dayClip11-13 for testing), and the results are listed in Table 9. Except Yolo2-tiny, both Yolo3-tiny and detector +Rttld can deal with full resolution of 1280*960. For the rest models, we have to resize the input resolution to satisfy the memory. Our model achieves the highest FPS (10.6) and IOU (40.45 percent), and Recall (31.4 percent) when dealing with full resolution images, which is about twice the speed of Yolo3-tiny. When resizing the input to $608 * 608$, the Yolo2-tiny, Yolo3-tiny and SSD can achieve up to 10 FPS (a basic requirement for autonomous vehicle); however, the resizing (down-sampling) process also leads to degradation of detection performance (see Yolo2-ting and Yolo3-tiny). As mentioned above, resizing of the input reduces the pixel features of each light, which is the main reason for low detection accuracy on small target of traffic lights.

Generally, the Recall should be always larger than IOU. However, when calculating the Recall, we only count the proposals with $IOU > 50\%$, leading to lower Recall. In addition, our model only achieves 40.45 percent of IOU and 31.4 percent of Recall. The results are not so remarkable because our heuristic ROI detector can generate only one large fixed ROI $(56 * 56)$ to contain the whole target, leading to larger denominator when calculating the IOU. In summary, our model performs very well and achieves the highest FPS for self-driving.

5 On-road Testing

To evaluate the mRttld and Rttld in a real autonomous vehicular environment, we deploy the models on one of the Tx2 modules of our Driven Brain (with 4 NVidia Jetson Tx2 and 2 FPGA) along with the whole self-driving system. Two

Fig. 10. Full-scale autonomous vehicles used for the on-road test in our test field: the left one is Changan RAETON (This vehicle won the Leadership Awards in the Urban Street Race and Country Street Race in the 2nd World Intelligent Driving Challenge, Tianjin, China.), and the right one is Yutong iBus.

types of full-scale autonomous vehicles (as shown in Fig. 10) are used for testing, with each being equipped with a Driven Brain, two AVT G-125c Cameras (with 5 mm Lens for lane line tracking and 16 mm Lens for traffic light detection) and a POS320 (DGPS system) for real-time positioning. The Robot Operating System (ROS) Kinetic Kame is used for communication between the vehicle Controller Area Network (CAN) bus, the sensors, and our decision system. The whole system runs on the Driving Brain at 10 Hz, and the vehicles travel at $40km/h - 60km/h$ on a straight road, and at $< 20km/h$ at intersections. The HD driving map we used during testing is illustrated in Fig. 11. It can be seen that the HD map contains some basic road information, such as lane lines, intersections, stop marks, the position of traffic lights, safety barriers, curbs, and other elements for self-driving.

As our aim is concentrated on the task of traffic light detection, we therefore ignore the navigation, vehicle control, and lane/stop line detection systems. The traffic light detection module will be activated when the distance between the traffic light and the vehicle is less than 50 m, and then will be turned off when the vehicle passes the intersection. When the traffic light detection module is activated, the vehicle will stop only when the red light is detected and when the vehicle is near to the stop line (offered by another camera with 5 mm lens). An overview of the procedure is given in Algorithm 1.

Algorithm 1. Passing Intersection

Require: Traffic Light Distance (D_{tl}), Traffic Light State (S), Stop Line Distance (D_{sl}), Aim Speed (AS)

update (D_{tl}, S, D_{tl}, AS)
while $0 < D_{tl} < 50$ **do**
  if S=Green **then**
    $AS \leftarrow 20km/h$
  else
    if $10 < D_{sl} < 30$ **then**
      $AS \leftarrow 10km/h$
    end if
    if $3 < D_{sl} <= 10$ **then**
      $AS \leftarrow 5km/h$
    end if
    if $D_{sl} <= 3$ **then**
      $AS \leftarrow 0km/h$
    end if
  end if
end while

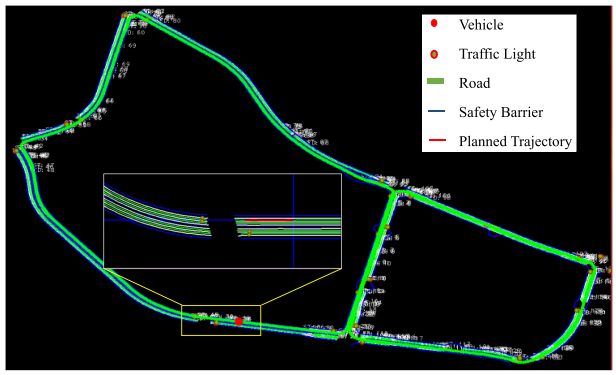


Fig. 11. On-road test field: the red mark and line are the current position of the vehicle and the planned trajectory generated from the navigation system, and the blue mark with the red annulus is the position of a traffic light.

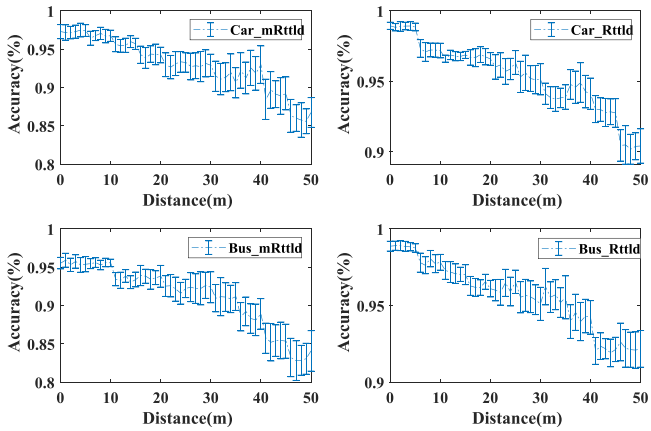


Fig. 12. Distance-accuracy evaluation: we calculate the average detection accuracy and the standard deviation according to the images and GPS coordinates recorded during self-driving mode.

In this experiment, we only care about the longitudinal control (i.e., the speed) of vehicle and use the aim speed to control the gas (throttle) and break. Both the car and the bus travel along the '8-shaped route' clockwise and counter-clockwise at daytime for 10 rounds (4.9 miles for a single round roughly). We also evaluate the average detection accuracy (with standard deviation as the error bar) of mRttld and Rttld models within 50m distance to each stop line at the intersections, as shown in Fig. 12. It is easy to find that all the models achieve higher accuracy and narrow error bars when the vehicle gets closer to the traffic lights/stop lines (the width of intersection is also affected by number of the lanes). Meanwhile, the Rttld model performs slightly better (2-6.2 percent) than the mRttld model. As our decision system will control the vehicle at low speed near the intersection, this will lead to accurate traffic light detection results and make the vehicle stop at the stop line. In addition, with a series of delay caused by hardware and network, both models can guarantee $> 10fps$ on the Tx2 module.

Furthermore, we use the distance between the vehicle and stop line after the vehicle stops as another indirect measurement to evaluate our system (as shown in Fig. 13). The distribution of errors when vehicles pass each interaction in the self-driving mode is calculated according to the DGPS coordinate (centimeter-level accuracy) of the distance from the vehicle head to the stop line on the HD map. As the bus

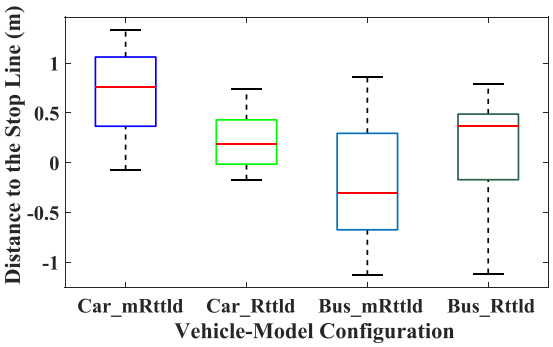


Fig. 13. Stop distance (meters) between the vehicle and the stop lines when mRttld and Rttld models are used on the car and bus (traveling 20 rounds of the '8-shaped route' using the strategy of Algorithm 1). The box and the band inside the box are the quartiles and the median of the distance, respectively. The ends of the whiskers are represented by minimum and maximum of each set of distances.

(with at least 6 operators) is much heavier than the car (with at least 2 operators), it often stops cross the stop line at a range of $[-1.2, 1]$. This means we should increase the braking distance for the bus. Using both models, the car can stop at the stop line at a range of $[-0.2, 1.34]$, but the distances fluctuate widely.

6 Conclusion

We have presented a practical traffic light detection system that combines the popular CNN classifier model and the heuristic ROI candidate detection algorithm to satisfy the requirement of self-driving hardware platform (NVidia Jetpack Tx1/2 on the Driving Brain). In this way, we develop a high-performance traffic light detection module which can handle high-resolution images to guarantee wide view and fulfill low weak computational vehicular hardware. We compare our model with several existing CNN models on different hardware platforms to show the performance of our model from different perspectives. In addition, we also conduct real on-road testing on different full-scale self-driving vehicles, the RAETON and Yutong Ibus, to evaluate our real-time traffic light detection module along with the whole self-driving system. Both simulation and real testing show acceptable performances of our hardware platform and self-driving system at low vehicle speed.

However, the current module still needs to be improved from the following aspects in the future. 1) The heuristic ROI detector can be improved by simple machine learning models. 2) To train and test the current module, the current dataset needs to be extended with more traffic light classes and images from worse light conditions. 3) The current model architecture can be improved with newly developed techniques in deep learning.

Acknowledgments

This work has been supported by the National Key R&D Program of China (2017YFB1301100), National Natural Science Foundation of China (61572060, 61772060, 61728201), State Key Laboratory of Software Development Environment (SKLSDE-2017ZX-18), and CERNET Innovation Project (NGII20160316, NGII20170315).

References

[1] O. Popescu, et al., "Automatic incident detection in intelligent transportation systems using aggregation of traffic parameters collected through V2I communications," *IEEE Intell. Transp. Syst. Mag.*, vol. 9, no. 2, pp. 64–75, Apr. 2017.

[2] R. Atallah, M. Khabbaz, and C. Assi, "Multihop V2I communications: A feasibility study, modeling, and performance analysis," *IEEE Trans. Veh. Technol.*, vol. 18, no. 2, pp. 416–430, Mar. 2017.

[3] G. Trehard, et al., "Tracking both pose and status of a traffic light via an interacting multiple model filter," in *Proc. 17th Int. Conf. Inf. Fusion*, 2014, pp. 1–7.

[4] A. Almagambetov, S. Velipasalar, and A. Baitassova, "Mobile standards-based traffic light detection in assistive devices for individuals with color-vision deficiency," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 3, pp. 1305–1320, Jun. 2015.

[5] J. Al-Nabulsi, A. Mesleh, and A. Yunis, "Traffic light detection for colorblind individuals," in *Proc. IEEE Jordan Conf. Appl. Electr. Eng. Comput. Technol.*, 2017, pp. 1–6.

[6] X. Li, et al., "Traffic light recognition for complex scene with fusion detections," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 1, pp. 199–208, Jan. 2018.

[7] M. B. Jensen, et al., "Vision for looking at traffic lights: Issues, survey, and perspectives," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 2, pp. 1800–1815, Jul. 2016.

[8] Y. Ji, et al., "Integrating visual selective attention model with hog features for traffic light detection and recognition," in *Proc. IEEE Intell. Veh. Symp.*, 2015, pp. 280–285.

[9] Z. Chen, Q. Shi, and X. Huang, "Automatic detection of traffic lights using support vector machine," in *Proc. IEEE Intell. Veh. Symp.*, 2015, pp. 37–40.

[10] Z. Chen and X. Huang, "Accurate and reliable detection of traffic lights using multiclass learning and multiobject tracking," *IEEE Intell. Transp. Syst. Mag.*, vol. 8, no. 4, pp. 28–42, Oct. 2016.

[11] Z. Shi, Z. Zou, and C. Zhang, "Real-time traffic light detection with adaptive background suppression filter," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 3, pp. 690–700, Mar. 2016.

[12] X. Du, et al., "Vision-based traffic light detection for intelligent vehicles," in *Proc. 4th Int. Conf. Inf. Sci. Control Eng.*, 2017, pp. 1323–1326.

[13] V. John, et al., "Saliency map generation by the convolutional neural network for real-time traffic light detection using template matching," *IEEE Trans. Comput. Imag.*, vol. 1, no. 3, pp. 159–173, Sep. 2015.

[14] S. Saini, et al., "An efficient vision-based traffic light detection and state recognition for autonomous vehicles," in *Proc. IEEE Intell. Veh. Symp.*, 2017, pp. 606–611.

[15] J. Campbell, et al., "Traffic light status detection using movement patterns of vehicles," in *Proc. 19th Int. Conf. Intell. Transp. Syst.*, 2016, pp. 283–288.

[16] Z. Ouyang, et al., "A cGANs-based scene reconstruction model using lidar point cloud," in *Proc. IEEE Int. Symp. Parallel Distrib. Process. Appl.*, 2017, pp. 1107–1114.

[17] K. Behrendt, L. Novak, and R. Botros, "A deep learning approach to traffic lights: Detection, tracking, and classification," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 1370–1377.

[18] M. Weber, P. Wolf, and J. M. Zollner, "DeepTLR: A single deep convolutional network for detection and classification of traffic lights," in *Proc. IEEE Intell. Veh. Symp.*, 2016, pp. 342–348.

[19] A. Mogelmose, M. M. Trivedi, and T. B. Moeslund, "Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 4, pp. 1484–1497, Dec. 2012.

[20] M. Chen, U. Challita, and E. A. Walid Saad, et al., "Machine learning for wireless networks with artificial intelligence: A tutorial on neural networks," *[J]. arXiv: Information Theory*, 2017. http://arxiv.org/abs/1710.02913

[21] M. Diaz-Cabrera, P. Cerri, and P. Medici, "Robust real-time traffic light detection and distance estimation using a single camera," *Expert Syst. Appl.: An Int. J.*, vol. 42, no. 8, pp. 3911–3923, 2015.

[22] H.-K. Kim, J. H. Park, and H.-Y. Jung, "Effective traffic lights recognition method for real time driving assistance systemin the daytime," *World Academy Sci. Eng. Technol.*, vol. 5, pp. 1424–1427, 2011.

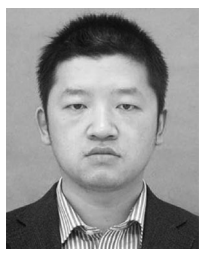
[23] M. P. Philipsen, et al., "Traffic light detection: A learning algorithm and evaluations on challenging dataset," in *Proc. IEEE 18th Int. Conf. Intell. Transp. Syst.*, 2015, pp. 2341–2345.

[24] V. H., et al., "Semantic segmentation based traffic light detection at day and at night," in *Proc. German Conf. Pattern Recognit.*, 2015, pp. 446–457.

[25] M. B. Jensen, et al., "Traffic light detection at night: Comparison of a learning-based detector and three model-based detectors," in *Proc. Int. Symp. Visual Comput.*, 2015, pp. 774–783.
[26] A. E. Gomez, et al., "Traffic lights detection and state estimation using hidden Markov models," in *Proc. Int. Symp. Visual Comput.*, 2014, pp. 750–755.
[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[28] J. Redmon, et al., "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 779–788.
[29] Lara traffic lights recognition (TLR) public benchmarks. [Online]. Available: http://www.lara.prd.fr/benchmarks/trafficlightsrecognition, Accessed: Feb. 2018.
[30] WPI traffic light dataset. [Online]. Available: http://computing.wpi.edu/dataset.html, Accessed: Feb. 2018.
[31] Bosch small traffic lights dataset. [Online]. Available: https://hci.iwr.uni-heidelberg.de/node/6132, Accessed: Feb. 2018.
[32] O. Russakovsky, et al., "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.
[33] A. Krizhevsky, "One weird trick for parallelizing convolutional neural networks," *[J]. arXiv: Neural and Evolutionary Computing*, 2014, https://arxiv.org/abs/1404.5997
[34] M. Lin, Q. Chen, and S. Yan, et al., "Network in network," *[J]. International conference on learning representations*, 2014, https://arxiv.org/abs/1312.4400
[35] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. 13th Eur. Conf. Comput. Vis.*, 2014, pp. 818–833.
[36] K. He, et al., "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[37] G. Huang, et al., "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2261–2269.
[38] S. Karen and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *[J]. International conference on learning representations (ICLR 2015)*, 2015, https://arxiv.org/abs/1409.1556
[39] C. Szegedy, et al., "Going deeper with convolutions," in *Proc. 14th Int. IEEE Conf. Intell. Transp. Syst.*, 2011, pp. 1609–1615.
[40] F. N. Iandola, S. Han, M. W. Moskewicz, et al., "SqueezeNet: Alexnet-level accuracy with 50x fewer parameters and $< 0.5mb$ model size," *[J]. arXiv: Computer Vision and Pattern Recognition*, 2017, https://arxiv.org/abs/1602.07360
[41] W. Liu, et al., "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 21–37.
[42] T.-Y. Lin, et al., "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 936–944.
[43] J. Dai, Y. Li, K. He, et al., "R-FCN: Object detection via region-based fully convolutional networks," *[J]. Neural information processing systems*, 2016, pp. 379–387, https://arxiv.org/abs/1605.06409
[44] S. Ren, et al., "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 91–99.
[45] Darknet. [Online]. Available: https://pjreddie.com/darknet/, Accessed: Feb. 2018.
[46] Cocodataset. [Online]. Available: http://cocodataset.org/, Accessed: Feb. 2018.
[47] J. Zhao, et al., "Men also like shopping: Reducing gender bias amplification using corpus-level constraints," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 2979–2989.
[48] Viva traffic light detection benchmark. [Online]. Available: http://cvrr.ucsd.edu/vivachallenge/index.php/traffic-light/traffic-light-detection/, Accessed: Feb. 2018.
[49] C. Gini, "Concentration and dependency ratios" *(in Italian). English translation in Rivista di Politica Economica*, vol. 87, pp. 769–789, 1997.

Zhenchao Ouyang received the MS degree from Inner Mongolia University, in 2014. Currently, he is working toward the PhD degree in the College of Computer and Science, Beihang University, Beijing, China. His research interests are machine learning and artificial intelligence in self-driving and mobile computing.

Jianwei Niu received the MS and PhD degrees in computer science from Beihang University, Beijing, China, in 1998 and 2002, respectively. He was a visiting scholar at the School of Computer Science, Carnegie Mellon University from Jan. 2010 to Feb. 2011. He is a professor with the School of Computer Science and Engineering, BUAA, and an IEEE senior member. His current research interests include mobile and pervasive computing, and mobile video analysis.



Yu Liu received the MS and PhD degrees in computer science from Beihang University, Beijing, China, in 2006 and 2010, respectively. He is an associate professor with the School of Computer Science and Engineering, BUAA. His current research interests include self-driving, eHealth, and machine learning.

Mohsen Guizani (S'85-M'89-SM'99-F'09) received the BS (with distinction) and MS degrees in electrical engineering and the MS and PhD degrees in computer engineering from Syracuse University, Syracuse, NY, USA, in 1984, 1986, 1987, and 1990, respectively. He is currently a professor in the CSE Department at Qatar University, Qatar. Previously, he served in different academic and administrative positions at the University of Idaho, Western Michigan University, the University of West Florida, the University of Missouri-Kansas City, the University of Colorado-Boulder, and Syracuse University. His research interests include wireless communications and mobile computing, computer networks, mobile cloud computing, security, and smart grid. He is currently the editor-in-chief of the *IEEE Network Magazine*, serves on the editorial boards of several international technical journals and is the founder and editor-in-chief of the *Wireless Communications and Mobile Computing* journal (Wiley). He is the author of nine books and more than 500 publications in refereed journals and conferences. He guest edited a number of special issues in IEEE journals and magazines. He also served as a member, chair, and general chair of a number of international conferences. Throughout his career, he received three teaching awards and four research awards. He also received the 2017 IEEE Communications Society WTC Recognition Award as well as the 2018 AdHoc Technical Committee Recognition Award for his contribution to outstanding research in wireless communications and Ad-Hoc Sensor networks. He was the chair of the IEEE Communications Society Wireless Technical Committee and the chair of the TAOS Technical Committee. He served as the IEEE Computer Society Distinguished Speaker and is currently the IEEE ComSoc Distinguished Lecturer. He is a Fellow of the IEEE and a senior member of the ACM.