Computational Visual Media
https://doi.org/10.1007/s41095-019-0152-1 Vol. 5, No. 4, December 2019, 403–416

Research Article

A three-stage real-time detector for traffic signs in large panoramas

Yizhi Song[1], Ruochen Fan[2], Sharon Huang[3], Zhe Zhu[4], and Ruofeng Tong[5] (✉)

Abstract Traffic sign detection is one of the key components in autonomous driving. Advanced autonomous vehicles armed with high quality sensors capture high definition images for further analysis. Detecting traffic signs, moving vehicles, and lanes is important for localization and decision making. Traffic signs, especially those that are far from the camera, are small, and so are challenging to traditional object detection methods. In this work, in order to reduce computational cost and improve detection performance, we split the large input images into small blocks and then recognize traffic signs in the blocks using another detection module. Therefore, this paper proposes a three-stage traffic sign detector, which connects a BlockNet with an RPN–RCNN detection network. BlockNet, which is composed of a set of CNN layers, is capable of performing block-level foreground detection, making inferences in less than 1 ms. Then, the RPN–RCNN two-stage detector is used to identify traffic sign objects in each block; it is trained on a derived dataset named TT100KPatch. Experiments show that our framework can achieve both state-of-the-art accuracy and recall; its fastest detection speed is 102 fps.

Keywords traffic sign; detection; real time

1 Department of Computer Science, Purdue University, 305 N. University Street, West Lafayette, IN 47907, USA. E-mail: song630@purdue.edu.

2 Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China. E-mail: frc16@mails.tsinghua.edu.cn.

3 College of Information Sciences and Technology, Penn State University, University Park, PA 16802, USA. E-mail: suh972@ist.psu.edu.

4 Department of Radiology, Duke University, Durham, NC 27705, USA. E-mail: zhe.zhu@duke.edu.

5 College of Computer Science and Technology, Zhejiang University, Hangzhou 310007, China. E-mail: trf@zju.edu.cn (✉).

Manuscript received: 2019-02-12; accepted: 2019-06-28

1 Introduction

Recently, self-driving cars have drawn more and more attention; they are expected to revolutionize the automobile industry. Rapid advances in environment sensing and navigation have led to significant improvements in autonomous vehicles. As road environment perception is a vital task for self-driving cars, there has been intensive research into traffic sign detection and classification. Traffic sign detection is not a new problem, and hundreds of methods have been proposed in the past decades. Although convolutional neural networks (CNNs) have shown their great power for general object detection, there are still some evident obstacles in traffic sign detection. In particular, in popular general object detection datasets, the objects typically occupy a large proportion of the image. In PASCAL VOC [1], the bounding box of a target object fills on average about 20% of the image. However, in contrast, traffic signs usually occupy a small proportion of each image in real-driving scenarios [2]. For autonomous driving, traffic signs should be detected and classified while still at a long distance, allowing decisions to be made well in advance. Under such circumstances, traffic signs in sensed images will be even smaller. In computer vision, small object detection has always been a challenge. Straightforward approaches such as enlarging the feature map, and using more small detection anchors, lead to huge computational costs. To be useful in real-world scenarios, it is necessary to develop a traffic sign detection approach that can handle small traffic signs in real time.

In this paper, we propose a novel traffic sign detection framework consisting of three detection stages. In this framework, an input image is divided into overlapping adjacent blocks. The first stage of the framework, named BlockNet, consists of several

CNN layers to extract features from each block and predict the probability that each block contains traffic signs. In the second stage, patches that are considered to contain targets are processed at a finer level by the *region proposal network* (RPN), which generates bounding box proposals. Finally, in the third stage, regions corresponding to the bounding box proposals are classified by the *region-based CNN* (R-CNN) and traffic signs recognized.

Based on the Tsinghua–Tencent 100K traffic sign dataset [2], we conduct a series of experiments to verify the effectiveness of our proposed framework. By changing key hyper-parameters of our network, such as the number of region proposals, we can strike a balance between detection speed and accuracy. Our experiments demonstrate that our network is able to achieve state-of-the-art detection accuracy when using 300 region proposals. Meanwhile, the proposed framework can process 100 frames per second, significantly faster than the previously fastest framework, yet having higher mAP.

The contributions of this paper are as follows:

- First, we propose a novel three-stage traffic sign detection framework which achieves both state-of-the-art detection accuracy and the fastest speed.
- Second, we address the challenge of detecting small traffic signs in large panoramic images by dividing them into smaller overlapping patches, then use a traffic sign pre-detection network we have designed, called BlockNet. This consists of a stack of convolutional layers with a large receptive field, so that it can efficiently and effectively classify a patch as either foreground (containing traffic signs) or background (containing no traffic sign).

The rest of the paper is organized as follows: in Section 2 we discuss related work. Details of our proposed framework are given in Section 3, including a description of the network structure and the training methodology. We give experimental results in Section 4, in which both accuracy and time efficiency are discussed. Finally we make conclusions and discuss our future work in Section 5.

2 Related work

2.1 Object detection and classification

Recent years have witnessed great advances in deep learning. The deployment of neural networks has shown promising performance for major computer vision tasks such as segmentation, detection, and classification. Neural networks of various architectures have been elaborately designed and implemented, adapted to particular situations. These frameworks compete with each other for speed and performance, and usually the leading method is replaced by another better one in a very short period of time.

Object detection methods can be divided into two kinds: those which are proposal-based, and the others which are proposal-free. The proposal-free methods avoid generating proposal bounding boxes, and try to locate and classify simultaneously, thus greatly reducing the time consumed. Overfeat [3] utilizes multi-scale sliding windows to perform the tasks of detection, localization, and classification at the same time within the same CNN framework. YOLO, introduced by Redmon et al. [4], divides the input image into grid cells. A grid cell is responsible for detecting the object whose center falls into it. This model predicts bounding-box coordinates and confidence in one shot. SSD [5], based on YOLO, replaces FC layers with a set of CNN layers for computing outputs, and performs predictions on multi-scale feature maps, giving comparable speed while reaching higher performance. Similar to SSD, RON [6] uses multi-scale feature maps for detection. The main improvement of RON is the reverse connection structure which integrates multi-scale features. Lin et al. [7] propose RetinaNet, which improves the detection performance by using focal loss to relieve the foreground–background class imbalance problem.

Among proposal-based methods, R-CNN [8], proposed in 2014, made a significant contribution to object detection, and provides a basic architecture used by successive detectors. Other methods greatly boost the performance as well as speed up the pipeline. Girshick [9] uses Fast R-CNN to simplify feature extraction with an RoI pooling layer. Ren et al. [10] train RPN to construct Faster R-CNN, which replaces selective search and provides faster convergence than all previous approaches. Recently, He et al. [11] proposed Mask R-CNN which replaces the RoI pooling layer in Faster R-CNN with RoI Align. This new operator can solve the misalignment problem in feature map quantization.

However, the detectors mentioned above are designed for general objects, and evaluated on datasets such as PASCAL VOC [12] and MS COCO [13]. As noted in the introduction, general object detection is different from traffic sign detection, so applying these detectors to traffic sign scenes does not lead to satisfactory results.

2.2 Traffic sign recognition

In the field of traffic sign recognition, it is necessary to collect and build specific datasets of street views which contain signs. Over the past few years many datasets have been constructed and released to the public. These datasets not only contribute to development of the field of piloted driving, but also offer standardized metrics for evaluation of works.

The German Traffic Sign Detection Benchmark (GTSDB) [14] is one of the most well-known publicly available traffic sign datasets; its images come from video sequences recorded in different environments. Another dataset named the German Traffic Sign Recognition Benchmark (GTSRB) [15] is intended for multi-category classification, and comprises more than 50,000 images of traffic signs from 43 classes. Timofte et al. [16] provided a dataset, BelgiumTS, containing more than 145,000 images captured on roads in Belgium, with 13,000 sign annotations included. Following publication of the GTSDB, numerous countries, including India [17], Sweden [18], and Brazil, began to show interest in traffic sign detection, collecting and releasing benchmarks of their own.

As research on autonomous automobiles gathers interest, several Chinese traffic sign datasets have been published. The Chinese Traffic Sign Dataset (CTSD) [19] is an image database with 1100 images (700 for training and 400 for testing) with typical sizes of 1024×768 and 1280×720. The signs are divided into four categories according to their color and shape. Zhu et al. [2] present a new challenging dataset created from 100,000 street view panoramas collected from city avenues and country roads all over China. It provides 10,000 images at the high resolution of 2048×2048, including sign instances for training and testing. Typical signs cover as little as 0.1% of the image area, and their instances come from more than 200 categories, both increasing the level of challenge.

Next, we introduce previous works that aim to solve the traffic sign detection and classification task. Although plenty of methods have been evaluated on GTSDB and GTSRB, TT100K is considered more challenging, since it captures tiny objects and they vary largely in illumination and weather conditions. A number of methods have been evaluated on this dataset since its release in 2016. Its creators, Zhu et al. [2], bring the branch of the network forward to the end of the 6th layer, as well as terminating the network in three streams, which enables their architecture to simultaneously detect and classify traffic signs, and outperform Fast R-CNN. Lu et al. [27] design a two-stage attention model, which is able to detect small objects from extracted attention proposals. Their model takes 0.26 s to process an image while achieving 87.0% mAP, close to the method of Zhu et al. Meng et al. [20] utilize a Small-Object-Sensitive-CNN (SOS-CNN) based on the Single Shot Multibox Detector (SSD) with a VGG-16 network. In this work, images are broken into patches and are firstly fed into an object-detection framework. Their approach slightly boosts accuracy and recall compared to previous ones. Yang et al. [21] introduce an attention network (AN) which generates potential regions of interest (RoIs) which are then filtered by a Fine Region Proposal Network (FRPN), thus greatly reducing the number of anchors. In the work of Pon et al. [22], a hierarchical network is employed to identify the class in a coarse-to-fine way, and they successfully settled the issue of combination and overlapping of datasets, which allows their architecture to detect traffic signs and lights jointly. At the expense of performance, the model is able to perform inference at more than 60 fps, much faster than all previous detectors evaluated on TT100K.

3 Approach

3.1 Preliminaries

Although many approaches have been proposed over the years to deal with the problem of detecting small objects, there is still difficulty in recognizing tiny traffic signs in real-world scenes. Since traditional proposal-based methods, for instance, Fast R-CNN and Mask R-CNN, cannot directly achieve satisfactory performance, many works try to design and add a pre-detection network that acts as a detector for

extracting coarse-level features before sending the results of such a network to another architecture like R-CNN. Some insert a coarse-level classifier, while others try to place a unit utilizing traditional computer vision characteristics like color information. However, none of the previous works can achieve a high average precision with real-time performance. The work of Pon et al. [22] consumes less time than all other works proposed for TT100K, but it does so at the cost of precision—its overall accuracy and recall are 68% and 44%, respectively. In order to perform real-time inference as well as reaching a competent mAP (mean average precision), we introduce in this paper a three-stage system that increases the detection speed to more than 100 fps.

Since in a real-driving scenario, traffic signs usually occupy a small proportion of the whole view, most of each input image is irrelevant and the targets are relatively too small to detect with ease. Detecting tiny objects requires large feature maps and many detection anchors, which significantly increases computational cost. Therefore, it is reasonable to predict block proposals containing traffic signs. We divide the input images into overlapping blocks and identify blocks possibly containing traffic signs for further consideration. We next give an overview of our framework, then discuss further details of BlockNet, which functions as a coarse classifier for dividing blocks into those containing signs or not, and a two-shot detector for the fine stage which localizes and categorizes traffic signs from more than 200 classes.

3.2 Network architecture

As shown in Fig. 1, the proposed small-object sensitive network is composed of two modules, BlockNet and a two-shot detector, with the latter combining an RPN (region proposal network) and an R-CNN for classification and coordinate regression, so it can be viewed as a three-part system. The first module is a CNN named BlockNet whose job is to locate traffic signs at the patch level. It divides the original 2048×2048 input image into 256 overlapping blocks, each of size 256×256, and predicts the probability that each either contains traffic signs or is merely background. This stage outputs 16×16 probability scores. In the second stage of the system, patches in the input image which BlockNet predicts to contain traffic signs are cropped and fed through a 101-layer ResNet [23] to extract their features. The features are then input into the RPN to generate region proposals from anchors. This way of generating region proposals is much faster than utilizing selective search as in the Fast R-CNN method. The RoI pooling layer (or RoI Align layer), located right after RPN, collects feature maps and proposals to extract proposal feature maps, which are finally passed through a network (R-CNN) to be classified. We now proceed to present more details of each stage of the overall network.

3.3 BlockNet for patch-level detection

The first network of CNN layers, BlockNet, is responsible for coarsely confining traffic signs to patches, to cut the recognition time spent by successive networks.

As illustrated in Fig. 2, BlockNet altogether has 6 convolutional layers, four of which have 3×3 kernels. The fourth convolutional layer uses a 5×5 filter with

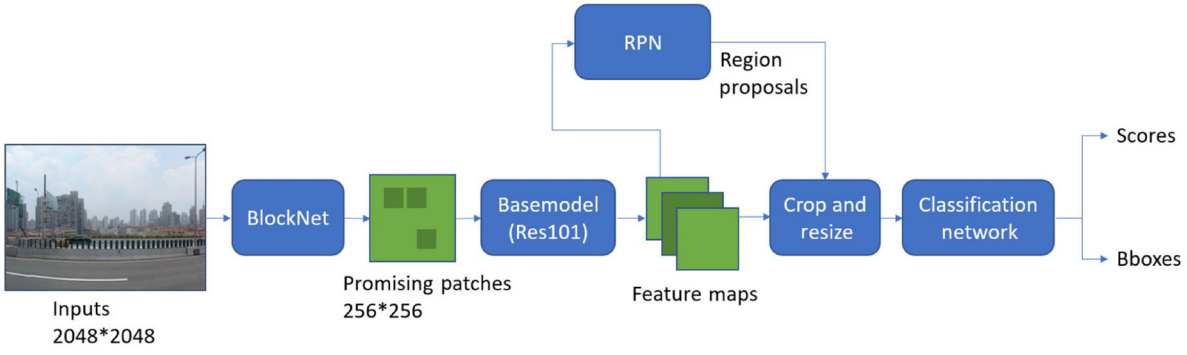

Fig. 1 Pipeline of the proposed architecture. The first stage, BlockNet, divides the input image into 256 overlapping squares and extracts features from each. After pushing these feature maps through a softmax layer, the network determines which blocks potentially include traffic signs, before passing them to later modules. All modules after BlockNet belong to the second stage. Res101 is used to extract fine features. RPN, at the side, covers the feature maps with anchors of various scales and ratios, and finally supplies region proposals back into the trunk. The classification network combines these outputs and performs regression as well as predicting the confidence.

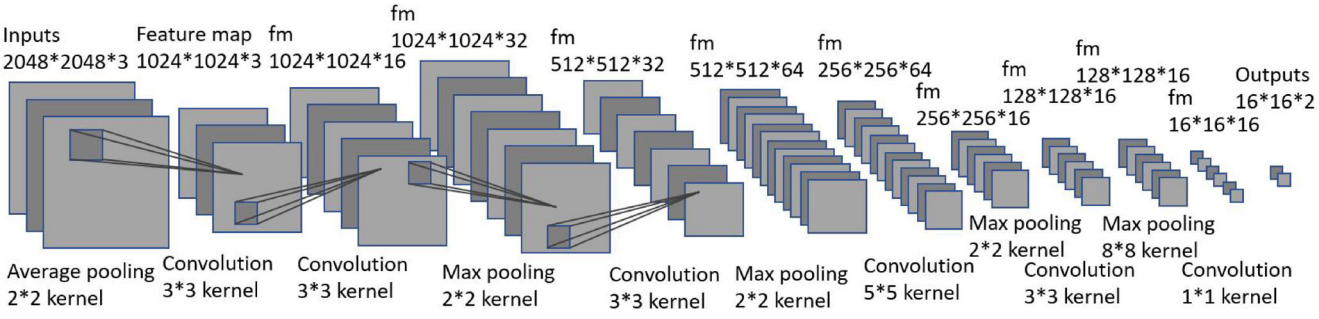

Fig. 2 CNN layers of BlockNet. This network is designed so that every computation does not produce any offset to centers of 16×16 blocks, and all blocks are distributed evenly. The output of BlockNet has two channels: one represents background, and the other represents foreground. Due to its simplicity, this network takes less than 1 ms to process an image. To boost the performance of the whole framework, some modifications must be included in this stage, replacing the 8×8 pooling layer with another two convolution layers.

a stride of 1, and the last convolutional layer employs 1×1 kernels. At the beginning of the network, the input images go through an average pooling layer to compress it by half. At the rear end, the outputs are computed using a softmax layer to produce a probability distribution.

BlockNet is simple but effectively and efficiently classifies blocks. All input images in TT100K are large photographs of size 2048×2048; they are down-sampled to 1024×1024 at the input to BlockNet. After a set of convolutional layers, the feature map has a resolution of 16×16 in two channels, with down-sampling rate of 128. Assuming the input image has been padded by 64 pixels for the first two dimensions (corresponding to width and height), each cell of the output feature can be mapped to an area of 256×256 in the processed 2304×2304 image: $(x_1, y_1) \mapsto (128x_1, 128y_1, 128x_1 + 255, 128y_1 + 255), 0 \leqslant x_1, y_1 \leqslant 15$ where the quadruple denotes a square region. Figure 3 illustrates the distribution of blocks. Since the computations of BlockNet do not produce any offsets, the 256 blocks are located evenly over the extended image, with an overlap of 128 both in the vertical and horizontal directions; the distance from the center of a marginal block to the nearest boundary is 64 pixels. We build the new block-based dataset from TT100K according to this mapping rule.

Our proposed BlockNet is able to produce a score map indicating the probability for each block of it containing traffic signs. Such score maps can be seen in Fig. 4, showing that blocks containing traffic signs are successfully recognized; they account for a small proportion of the overall number of blocks. Most blocks are identified as irrelevant and the computational cost in the downstream detection module is significantly reduced. Meanwhile, the traffic signs occupy a larger relative area in the identified blocks, which is beneficial to the further detection module. A confidence threshold (adjusted through numerous tests, in the range 0.50–0.70, to give the best performance) is applied to the outputs of the softmax layer. All cells scoring higher than the threshold in the second channel are believed to contain traffic signs. Due to the simple structure of BlockNet, the time a batch of samples takes during inferencing can almost be ignored comparing to that

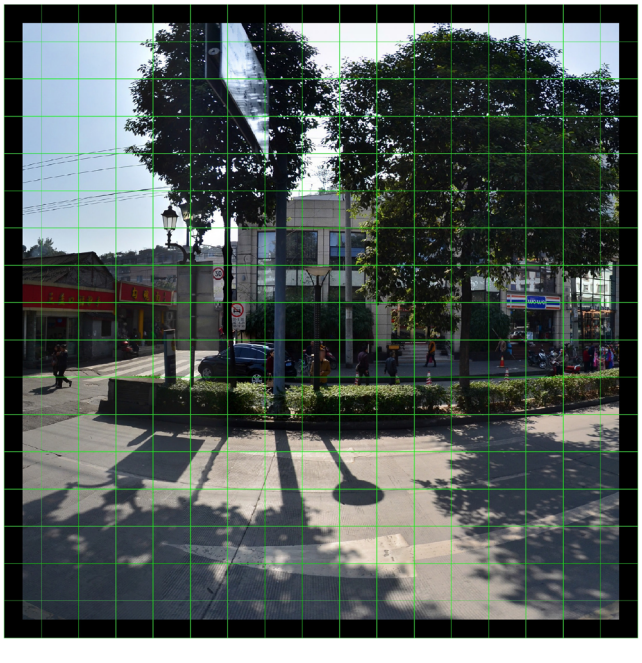

Fig. 3 Production of 256 blocks. The black frame is padding of 64 pixels. Since there are overlaps, each 2×2 green frame surrounds a block. The area of a block is just 1/64 of the original input, which greatly increases the proportion of traffic sign instances and cuts down the difficulty for later stages, since the blocks can be rescaled to 600×600 or even larger.

Fig. 4 Block score maps predicted by BlockNet. Blocks with deeper red coloring block with higher probability of containing traffic signs. These results demonstrate that our BlockNet produces correct block proposals. A great many irrelevant areas are filtered out; traffic signs occupy a relatively larger proportion of the block proposals.

required by the second stage, which is less than 1 ms.

3.4 Two-shot detector for fine localization and classification

In the localization and classification module, we use a two-stage network modified from the implementation of Yang et al. [24] to produce fine-grained classification of those foreground blocks output by BlockNet. The network can be viewed as a combination of an RPN and an R-CNN module, with the RPN generating promising RoIs for the R-CNN to classify as one of more than 200 classes of traffic sign. It outputs the confidence for each class (221 classes in TT100K) as well as coordinates of bounding boxes (up to 100 boxes are produced during testing and inferencing) in every category. This stage cuts the detection time in two ways. First, the usage of RPN improves the efficiency of selecting proposals; second, previous approaches have computed on 600×600 images rescaled from original ones, but after selection, the network only deals with 128×128 inputs down-sampled from the patches. Each image provides on average 6.17 patches through BlockNet, discarding the remainder. Furthermore, since traffic signs cover a much larger proportion of the blocks, it is simpler for the network to detect these objects: an 80×80 object occupies 0.1% of the original image but as much as 10% of a block cropped from it.

3.5 Training

Training of this framework can be divided into two separate sections, respectively concerning BlockNet and the two-shot network.

To train this three-part system, we built TT100KPatch, a new dataset derived from TT100K. After padding, each image generates 256 patches (named using a composite index of image ID and patch ID in the range 0–255) in the new dataset, and a new file of ground truth labels is saved by adding an offset to coordinates from the original annotation

file. Since most traffic signs are included in more than one patch, the number of ground truth annotations grows to be far greater than the original number.

In the first stage, given that there are sufficient blocks in the training set, images are only normalized before entering the network, without further data augmentation. The inputs then go through the CNN layers to extract features for each block. A batch-normalization layer is inserted after every convolutional layer to accelerate training. Computations during the process ensure that each cell in the output corresponds to the center of a block. We employ Adam as the optimizer and the learning rate is set to 0.001 which decays after 7 epochs out of 12 in total. The tensor output of BlockNet has two channels: one for background and the other for foreground objects. After every iteration, another softmax layer is applied to the tensor, computing probability along the dimension of channels. BlockNet utilizes cross-entropy as the loss function to measure and minimize the binary classification loss, defined as follows:

$$E(x,l) = -w_l \log \frac{\exp(x_l)}{\sum_{j=1}^{N} \exp(x_j)} \tag{1}$$

In Eq. (1), E is cross-entropy, x is the raw activation value, N is the number of dimensions, l is a label representing each target class, and w is a weight for each class. Before this stage comes to an end, indexes of promising patches whose confidence is above a predefined threshold are written to file, and saved for the next stage.

Training of the second stage is performed in several steps: preprocessing, data augmentation, anchor generation, training Fast R-CNN and RPN separately, and then together.

During preprocessing, the input images are read in and a mean vector ($[102.98, 115.95, 122.77]$) is subtracted from the RGB channels. Then the resulting images are rescaled to 128×128, to simplify the successive computations and thus accelerate testing and inferencing. Down-sampling however makes it harder for the network to extract features from a shrunken map four times smaller than the original one.

To help the model be more robust to changes in shape and location of traffic signs, flipping is used as the major data augmentation approach: it reverses the image and doubles the RoI set. Training samples are flipped and the corresponding RoI annotations are computed at the same time, before addition to the RoI database for training.

The anchor generation layer is responsible for producing anchor boxes spread over the preprocessed image. Two key parameters, anchor scales and anchor ratios, help the RPN to apply to various sizes and shapes of the target objects. To save time, the image is shrunken to one-quarter of 256×256, and scales are also adjusted, which are Scale $\in \{4, 8, 16, 32\}$. Three aspect ratios are chosen for the boxes, which are Ratio $\in \{0.5, 1.0, 2.0\}$. Then 12 sliding windows are generated by the following formula:

$$\begin{cases} h = s_i b \sqrt{r_j} \\ w = s_i b / \sqrt{r_j} \end{cases} \tag{2}$$

where s_i is scale, r_j is ratio, h is height, w is width, and b denotes length of the base window. Given that stride length is set to 16 by default, the map is divided into a mesh with spacing of 16. Generating 12 windows for every cell produces a total of more than 3000 anchors. Those lying outside the map are cropped. To train the RPN, a measure which evaluates to what degree an anchor matches the ground truth box must be defined. In this framework, IoU (intersection over union) is employed. Among all anchors produced by the proposal layer, those with an IoU above 0.7 for some ground truth bounding boxes are considered to be foreground samples,while those with an IoU of all ground truths lower than 0.3 are judged to be background samples.

Then the metric which optimizes the RPN is the one which measures the proportion of bounding boxes that are correctly predicted as positive or negative samples, as well as the distance between the coordinates of the predicted box and the target. The RPN loss function is defined as the sum of classification loss and regression loss as shown below:

$$\text{RPNLoss} = \text{Loss}_\text{C} + \text{Loss}_\text{R} \tag{3}$$

Loss_C is similar to that used in BlockNet:

$$\text{Loss}_\text{C} = \sum_{i=1}^{N} \text{CrossEntropy}(p_i, p_i^*) \tag{4}$$

where N is the total number of foreground anchors, p is a predicted anchor, and p^* denotes a target. Regression loss takes the form:

$$\text{Loss}_\text{R} = \sum_{i=1}^{N} \sum_{j \in \{x,y,w,h\}} L_1(u_{ij} - u_{ij}^*) \tag{5}$$

$$L_1(x) = \begin{cases} \sigma^2 x^2/2, & ||x|| < 1/\sigma^2 \\ ||x|| - 0.5/\sigma^2, & \text{otherwise} \end{cases} \quad (6)$$

where $\{x, y, w, h\}$ are not original coordinates in the image, but regression coefficients instead, and L_1 stands for smooth L1 loss. The computation of the RPN loss function corresponds to the two branches in the network architecture. The loss function in the classification layer near the end of the framework highly resembles that of RPN, the major difference being that it has to deal with all object classes:

$$\text{Loss} = \frac{-1}{M} \sum_{i}^{M} \log \frac{\exp(x[i][c_i])}{\sum_{j}^{N} \exp(x[i][j])} \quad (7)$$

where M is the number of samples, N is the number of classes, and x is an $M \times N$ matrix of scores.

Given the loss functions above, the training of this stage can be optimized. Following Ren et al. [10], training begins by training the RPN and Fast R-CNN alone, and ends by fine-tuning the two components.

During the training period, the model goes through 145,960 iterations in 10 epochs. The learning rate is set to 0.001 and decays after 7 epochs. Due to the large number of samples in TT100KPatch, and an increased proportion of objects in the images, the model converges rapidly and already reaches an mAP as high as 89.60% after the first two epochs (with resizing to 400×400).

3.6 Testing

During the first stage, 3070 images, normalized using a mean deviation of $[0.485, 0.456, 0.406]$ and standard deviation of $[0.229, 0.224, 0.225]$, were fed into the trained BlockNet. Near the output of the network, a score threshold was set to filter blocks with objects. Setting the threshold to 0.08 generates an average of 6.17 blocks per image on average, while cutting it down to 0.05 generates 6.29 blocks per image. This provides a tradeoff between speed and performance, since the detector will be less likely to miss targets given a higher threshold, boosting recall of this stage, but the second stage will take more time.

During the second stage, all promising blocks returned by the first stage are rescaled and normalized in the same way as in training to provide corresponding results. Because of data imbalance, we only evaluate 45 classes (out of 221) of traffic signs, although instances of all classes are localized and recognized during testing.

Two approaches are employed to accelerate testing and inferencing. One is to rescale the original 256×256 blocks to 128×128, while the other is to reduce the number of proposed anchors in the proposal layer. More specifically, 1000 boxes are kept before applying non-maximum suppression (NMS) to all proposals, while only 6 top scoring boxes are kept after NMS. This trades off performance for an increase in efficiency. During testing, the model loads pretrained weights for Res101 from ImageNet.

4 Results and discussion

4.1 Experiments

Using a GTX 1080 Ti GPU to evaluate the proposed three-stage detector, we performed experiments on the TT100K dataset, which comprises 6104 training samples and 3070 test samples containing 221 classes. As in Zhu et al. [2] and Pon et al. [22], 45 categories having more than 100 instances were selected for evaluation.

We combine our trained BlockNet model and two-shot network for testing and inferencing. In the first stage, the BlockNet model was trained with a learning rate of 0.001, decay step of 8, and weight decay of 0.0001, converging after 18,300 iterations. In the second stage, the two-shot network model was trained with a learning rate of 0.001, decay step of 8, and batch size of 4. In addition, a momentum of 0.9, weight decay of 0.0005, and stochastic gradient descent were utilized. The model trained after 145,000 iterations was saved.

During evaluation, only bounding boxes with IoU greater than 0.5 with target boxes were considered correct predictions. When computing accuracy and recall, boxes with confidence scores lower than 0.10 were discarded.

Apart from experiments on the proposed model, we also evaluated other typical proposal-based and proposal-free methods to assess the effectiveness of our framework. These were YOLOv3 and Faster R-CNN. The latter uses no auxiliary constructions or front networks but directly feeds images at the original scale into R-CNN. This experiment convincingly confirmed the indispensable role of BlockNet.

We conducted both quantitative experiments and visualization experiments. The output of BlockNet can be seen in Fig. 4; detection results are depicted in Fig. 5.

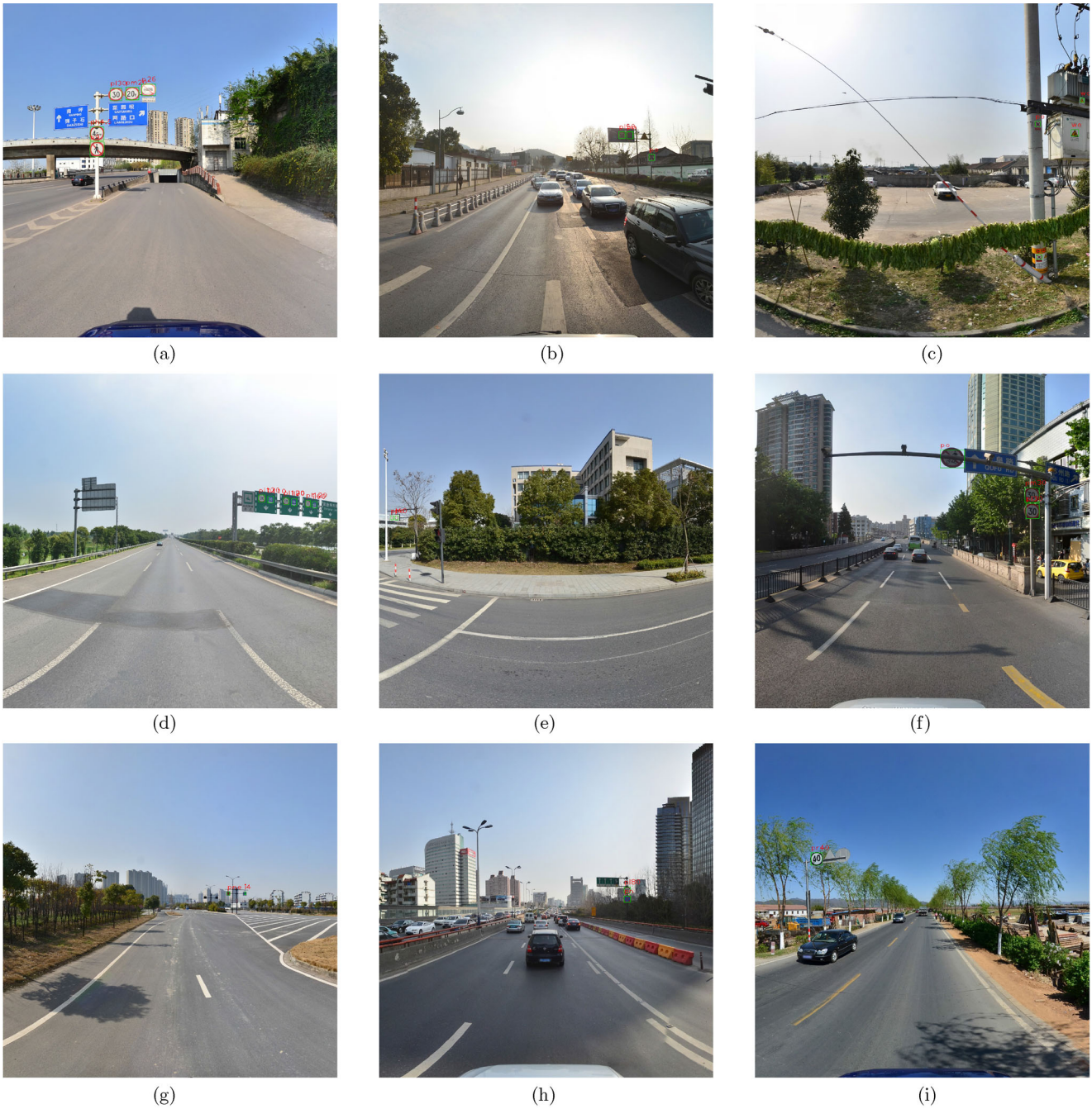

Fig. 5 Detection results. Objects are surrounded by green boxes with red class captions next to them. In real-driving scenes, traffic signs are very small from the perspective of a wide-angle lens, so are hard to be detected. The proposed framework is capable of recognizing multiple very tiny target objects from more than 200 classes in 2048 × 2048 images. Their changing and varied shapes, perspective, and illumination, again well demonstrate the difficulty of this dataset. The detector is also relatively invariant to deformation caused by the panoramas.

4.2 Performance

In this section, we measure the detection performance using the metric provided by TT100K to compute accuracy and recall, and the metric of PASCAL VOC 2007 [12] to compute mAP (since it is not present in TT100K).

Table 1 shows detection results with our approach, as well as other methods, demonstrating that our proposed framework achieves state-of-the-art performance in both recall and accuracy. Among these proposed frameworks, Meng et al. have achieved the highest performance with accuracy of 0.90 and

Table 1 Detection results: performance of state-of-the-art works. **acc**: accuracy. **rec**: recall. Results of only using Faster R-CNN for the task are also shown. **Ours**†: proposed framework with few block proposals for high speed. **Ours**‡: proposed framework adjusted for best accuracy. Our proposed framework achieves the best recall and accuracy. The proposed BlockNet significantly boosts detection recall and accuracy with respect to a standard two-shot detection framework (Faster R-CNN)

Work	mAP	acc	rec
Zhu et al.	—	0.88	0.91
Li et al. [25]	—	0.89	0.91
Meng et al.	—	0.90	**0.93**
Pon et al.	0.31	0.68	0.44
Yang et al.	0.80	—	—
Faster R-CNN	0.38	0.64	0.37
Ours†	0.52	0.69	0.65
Ours‡	**0.90**	**0.92**	**0.93**

recall of 0.93. Since the methods of Zhu et al. and Meng et al. mainly focus on a precise recognition of traffic signs, they do not offer any information on the time consumed to deal with every single image. Efficiency will be further discussed later.

We also conducted an ablation study to validate the effectiveness of BlockNet, using a basic two-shot model which only includes a Faster R-CNN. The original images are sent into it without any feature extraction in advance. Table 1 shows that the standard two-shot model Faster R-CNN provides lower recall and accuracy compared to the proposed three-stage framework. This indicates that BlockNet reduces the difficulty of detecting tiny traffic signs and helps to reach a significantly higher mAP.

Compared to standard existing detectors, our framework provides a better solution to the problem of recognizing traffic signs in large images in real time. As Table 3 shows, we firstly tested Faster R-CNN on TT100K, and the result is far from our desired goal. Our method makes two major modifications to it: decreasing the number of proposals before and after NMS, and shrinking the images the net consumes; to enhance performance, we add a patch-level stage before the net. Secondly we find out that fast as it is, YOLO is not precise enough, as it selects candidates in only one step. Though our detector extends to three steps, the first does not cost much time, while it greatly simplifies the job of the later stages.

To better analyse the sensitivity to objects of various scales, the targets were divided into three groups according to size: small instances, with area less than 32^2, medium instances with area between 32^2 and 96^2, and large ones, bigger than 96^2. Detection results are given in Table 2. Small and medium targets, constitute 52% and 45% respectively, accounting for the majority. The difficulty in identifying small objects leads to the drop in mAP.

There are two reasons for the insensitivity to tiny objects. First, during the test, when the proposal threshold of BlockNet has been set, experiments show that accuracy and recall for classification are 0.990 and 0.903 separately. A very high pre-detection performance is provided by BlockNet. Thus the whole system suffers losses mainly in the second stage, where the down-sampling of inputs (from 256 to 128) and reduction of proposals after NMS undermine the capability of the two-shot detector. Second, in our experiments, for the whole TT100K dataset, BlockNet produces as many as 5700 false positive blocks which actually contain no signs or just parts of them. All detection results predicted from these false positive blocks in the second stage result in false positives. Therefore, the performance of the BlockNet affects the overall recall and accuracy of the whole detection pipeline.

We also evaluated the performance for each class in the TT100K database: see Table 4. The results come from experiment "Test 5", which can be seen in Table 5.

4.3 Speed

Inference time is now discussed in this section. As mentioned above, there are three approaches to

Table 2 Sensitivity to objects of different sizes. Small and medium objects are dominant in the dataset, verifying the difficulty of TT100K. **gt**: number of ground truth boxes whose area falls into the interval. **rec**: recall. The model can readily identify objects larger than 32^2

Group	gt	ratio	rec
Small	11,525	52%	0.5392
Medium	9,983	45%	0.9386
Large	702	3%	0.9316

Table 3 Efficiency of state-of-the-art methods. Our proposed framework boosts the highest inference speed by 35%, while outperforming the work of Pon et al.

Work	mAP	inf speed (s)	fps
Li et al.	—	0.6	1.7
Yang et al.	0.80	0.1282	7.8
Pon et al.	0.31	0.015	66.7
YOLOv3 [26]	0.33	0.030	33.3
Faster R-CNN	0.38	0.022	45.5
Ours	0.52	**0.0098**	**102.0**

Table 4 Performance (using the fastest model trained) over 45 categories having more than 100 instances. mAP of different classes varies greatly due to varying numbers of instances and difficulty of extracting features. Classes with distinct geometrical characteristics (for instance, pne and pn), are easier to recognize, while triangular signs are harder to distinguish

Class	**i2**	**i4**	**i5**	**il100**	**il60**	**il80**	**io**	**ip**	**p10**	**p11**	**p12**	**p19**	**p23**	**p26**	**p27**
mAP	0.468	0.674	0.747	0.624	0.708	0.552	0.537	0.607	0.525	0.529	0.346	0.523	0.620	0.542	0.504
Class	**p3**	**p5**	**p6**	**pg**	**ph4**	**ph4.5**	**ph5**	**pl100**	**pl120**	**pl20**	**pl30**	**pl40**	**pl5**	**pl50**	**pl60**
mAP	0.489	0.716	0.318	0.544	0.332	0.573	0.363	0.682	0.581	0.358	0.390	0.560	0.642	0.419	0.465
Class	**pl70**	**pl80**	**pm20**	**pm30**	**pm55**	**pn**	**pne**	**po**	**pr40**	**w13**	**w32**	**w55**	**w57**	**w59**	**wo**
mAP	0.364	0.490	0.401	0.346	0.651	0.698	0.763	0.393	0.721	0.361	0.599	0.289	0.566	0.548	0.249

Table 5 Timing and performance. **pth/img**: average number of patches produced by an image. Numbers of proposals before and after NMS: **PRE-NMS**, **POST-NMS**. All inputs have a batch size of 24. The last three columns are inference time (in second) for BlockNet, the two-shot network and the whole pipeline = **pth/img** × **inf2/img** + **inf1/img**. Decreasing the number of proposals after NMS and zooming out the blocks result in a loss in performance, though earning more time. Increasing the sensitivity of BlockNet leads to a growth in the total number of blocks selected, eventually causing the whole network to slow down

No.	**pth/img**	**scale**	**PRE-NMS**	**POST-NMS**	**mAP**	**acc**	**rec**	**inf1/img**	**inf2/img**	**time/img**
test1	4.44	600	6,000	300	0.7707	0.9211	0.8136	0.00090	0.02176	0.09749
test2	6.17	600	1,000	6	0.7437	0.9385	0.7715	0.00091	0.01876	0.11667
test3	4.44	256	1,000	6	0.6770	0.8658	0.7005	0.00091	0.00406	0.01894
test4	6.17	128	1,000	6	0.5141	0.6788	0.6523	0.00089	0.00142	0.00963
test5	6.29	128	1,000	6	0.5194	0.6908	0.6532	0.00089	0.00142	0.00980
test6	5.64	800	6,000	300	0.8989	0.9204	0.9327	0.00112	0.03541	0.20083

cutting the time consumed. Because of the simplicity of the architecture of BlockNet, the time it takes accounts for a very small proportion of the overall inference time. Along with this are the reduced numbers of proposals before and after NMS at the target proposal layer.

In Table 3, we compare our model with state-of-the-art methods. The work of Yang et al. with ZF net takes 128 ms on a Tesla K20 GPU, while the system designed by Pon et al. takes 15 ms on a GTX 1080 Ti GPU; it is 7.5 times faster, at the expense of performance. We also tested a proposal-free detection network YOLOv3 (with input size 256×256); it is also slower than our three-stage framework. Our framework is not only able to perform inference at 102 fps, which exceeds all previous works on the same benchmark, but also outperforms the hierarchical detector of Pon et al.

Table 5 presents timing and performance of the best model along with several baseline experiments. Test 6 produces the best results with an mAP of 0.8989, which demonstrates the necessary role of BlockNet and the influence of image scale. This test also proves that the model is able to precisely solve the task for identification of small objects. Test 4, the only difference from Test 2 being use of a much smaller image scale, shows how down-sampling the original image results in weakened features, and thus a sharp drop in performance. The score threshold of BlockNet in Test 4 was 0.08, while in Test 5 it was reduced to 0.05. This modification slightly raises mAP by around 0.5%, but at the same time slows the computation down by 0.2 ms, because of the extra patches generated per image. Some noticeable measures are taken in Test 6 to ensure the highest performance: allowing more proposal boxes to be passed to the next stage, enlarging the image scale, adapting feature extraction to a greater extent. Among them, resizing the scale to 800×800 proves to be the most effective, but also lengthens the time consumed. Inserting another two 3×3 convolution layers near the end of BlockNet trades very little time (less than 1 ms) for a considerable increase in sensitivity. An additional timing test using YOLO v3 on the basis of the implementation by Redmon et al. [4] takes 30 ms for inference, with a lower mAP.

The experiments above reveal the factors affecting speed and performance in our framework. Larger input images (larger and more precise feature maps) and more bounding box proposals lead to higher performance but slower inferencing. Therefore, we can regulate these two hyper parameters to strike a balance between performance and speed. As mentioned above, BlockNet's accuracy and recall are

very high, and BlockNet's inference process is very fast, so adjusting BlockNet's hyper parameters has little influence.

5 Conclusions

This paper presents a three-stage detector for real-time traffic sign recognition. Taking into account the small proportion of targets in the TT100K dataset, we designed a CNN named BlockNet as a front-end module, and connected it to a two-shot network for two-stage detection. The input images are divided into overlapping blocks. BlockNet coarsely predicts targets at a patch level, while the following two stages finish the work at a fine-grained level. To train this model, a derived dataset called TT100KPatch was built which breaks every image into 256 blocks. Without boosting approaches, the network achieves an mAP of 0.7707. After acceleration, the model can perform inferencing at a speed of 100 fps, faster than the current state-of-the-art, while maintaining an mAP of 0.5194.

In the future, we will seek a more precise model that is sensitive to small objects and increases performance on boxes smaller than 32^2. The two-shot detector we employed needs to be fine-tuned. For the moment the model is only suitable for images of fixed dimensions of 2048×2048, so the next goal is to upgrade it to a scale-invariant framework. Another issue is that the confidence with which foreground objects is predicted is not strong enough, so a more robust front stage needs to be trained.

Acknowledgements

We thank all anonymous reviewers for their valuable comments and suggestions. This paper was supported by the National Natural Science Foundation of China (No. 61832016) and Science and Technology Project of Zhejiang Province (No. 2018C01080).

References

[1] Everingham, M.; van Gool, L.; Williams, C. K. I.; Winn, J.; Zisserman, A. The Pascal visual object classes (VOC) challenge. *International Journal of Computer Vision* Vol. 88, No. 2, 303–338, 2010.

[2] Zhu, Z.; Liang, D.; Zhang, S. H.; Huang, X. L.; Li, B. L.; Hu, S. M. Traffic-sign detection and classification in the wild. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2110–2118, 2016.

[3] Sermanet, P.; Eigen, D.; Zhang, X.; Mathieu, M.; Fergus, R.; LeCun, Y. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint* arXiv:1312.6229, 2013.

[4] Redmon, J.; Divvala, S.; Girshick, R.; Farhadi. A. You only look once: Unified, real-time object detection. In: Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition, 779–788, 2016.

[5] Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C. Y.; Berg, A. C. SSD: Single shot MultiBox detector. In: *Computer Vision–ECCV 2016. Lecture Notes in Computer Science, Vol. 9905.* Leibe, B.; Matas, J.; Sebe, N.; Welling, M. Eds. Springer Cham, 21–37, 2016.

[6] Kong, T.; Sun, F. C.; Yao, A. B.; Liu, H. P.; Lu, M.; Chen, Y. R. RON: Reverse connection with objectness prior networks for object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 5244–5252, 2017.

[7] Lin, T. Y.; Goyal, P.; Girshick, R.; He, K. M.; Dollar, P. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* DOI: 10.1109/TPAMI.2018.2858826, 2018.

[8] Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 580–587, 2014.

[9] Girshick, R. Fast R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, 1440–1448, 2015.

[10] Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks In: Proceedings of the Advances in Neural Information Processing Systems 28, 91–99, 2015.

[11] He, K.; Gkioxari, G.; Dollar, P.; Girshick, R. Mask R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, 2961–2969, 2017.

[12] Everingham, M.; van Gool, L.; Williams, C. K. I.; Winn, J.; Zisserman, A. The Pascal visual object classes challenge 2007 (voc2007) results. Available at http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[13] Lin, T. Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C. L. Microsoft COCO: Common objects in context. In: *Computer Vision–ECCV 2014. Lecture Notes in Computer Science, Vol. 8693.* Fleet, D.; Pajdla, T.; Schiele, B.; Tuytelaars, T. Eds. Springer Cham, 740–755, 2014.

[14] Houben, S.; Stallkamp, J.; Salmen, J.; Schlipsing, M.; Igel, C. Detection of traffic signs in real-world images: The German traffic sign detection benchmark. In: Proceedings of the International Joint Conference on Neural Networks, 1–8, 2013.

[15] Stallkamp, J.; Schlipsing, M.; Salmen, J.; Igel, C. The German traffic sign recognition benchmark: A multi-class classification competition. In: Proceedings of the International Joint Conference on Neural Networks, 1453–1460, 2011.

[16] Timofte, R.; Zimmermann, K.; van Gool, L. Multi-view traffic sign detection, recognition, and 3D localisation. *Machine Vision and Applications* Vol. 25, No. 3, 633–647, 2014.

[17] Hemadri, V. B.; Kulkarni, U. P. Recognition of traffic sign based on support vector machine and creation of the Indian traffic sign recognition benchmark. In: *Cognitive Computing and Information Processing. Communications in Computer and Information Science, Vol. 801*. Nagabhushan, T.; Aradhya, V.; Jagadeesh, P.; Shukla, S. Eds. Springer Singapore, 227–238, 2018.

[18] Larsson, F.; Felsberg, M. Using Fourier descriptors and spatial models for traffic sign recognition. In: *Image Analysis. Lecture Notes in Computer Science, Vol. 6688*. Heyden, A.; Kahl, F. Eds. Springer Berlin Heidelberg, 238–249, 2011.

[19] Yang, Y.; Luo, H. L.; Xu, H. R.; Wu, F. C. Towards real-time traffic sign detection and classification. *IEEE Transactions on Intelligent Transportation Systems* Vol. 17, No. 7, 2022–2031, 2016.

[20] Meng, Z.; Fan, X.; Chen, X.; Chen, M.; Tong, Y. Detecting small signs from large images. In: Proceedings of the IEEE International Conference on Information Reuse and Integration, 217–224, 2017.

[21] Yang, T. T.; Long, X.; Sangaiah, A. K.; Zheng, Z. G.; Tong, C. Deep detection network for real-life traffic sign in vehicular networks. *Computer Networks* Vol. 136, 95–104, 2018.

[22] Pon, A.; Adrienko, O.; Harakeh, A.; Waslander, S. L. A hierarchical deep architecture and mini-batch selection method for joint traffic sign and light detection. In: Proceedings of the 15th Conference on Computer and Robot Vision, 102–109, 2018.

[23] He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 770–778, 2015.

[24] Yang, J.; Lu, J.; Batra, D.; Parikh, D. A faster pytorch implementation of faster R-CNN 2017. Available at https://github.com/jwyang/faster-rcnn.pytorch.

[25] Li, J.; Liang, X.; Wei, Y.; Xu, T.; Feng, J.; Yan, S. Perceptual generative adversarial networks for small object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1222–1230, 2017.

[26] Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv preprint* arXiv:1804.02767, 2018.

[27] Lu, Y.; Lu, J.; Zhang, S.; Hall, P. Traffic signal detection and classification in street views using an attention model. *Computational Visual Media* Vol. 4, No. 3, 253–266, 2018.

Yizhi Song is a Ph.D. student in the Department of Computer Science at Purdue University. He received his B.E. degree from the College of Computer Science and Technology at Zhejiang University. His research interests are in the fields of computer vision, computer graphics, and information visualization.

Ruochen Fan is a master candidate in the Department of Computer Science and Technology, Tsinghua University. He received his bachelor degree from Beijing University of Posts and Telecommunications in 2016. His research interest is computer vision.

Sharon Huang received her B.E. degree in computer science from Tsinghua University in 1999, and her M.S. and Ph.D. degrees in computer science from Rutgers University in 2001 and 2006, respectively. She is currently an associate professor in the College of Information Sciences and Technology and a co-hire with Huck Institutes of the Life Sciences at Penn State University, USA. Her research interests are in the areas of biomedical image analysis, computer vision, machine learning, and computer graphics, focusing on object recognition, segmentation, registration, matching, real-time tracking, skeletonization, and deformable (non-rigid) model based methods.

Zhe Zhu is a postdoctoral associate at Duke University, working with Dr. Maciej A. Mazurowski. He received his Ph.D. degree from the Department of Computer Science and Technology, Tsinghua University, under the supervision of Prof. Shi-Min Hu. He got his B.Sc. degree from the School of

Computing, Wuhan University, under the supervision of Profs. Aiguo Yao and Zhiyong Yuan. In 2016, he worked as a research intern with Drs. Brian Price and Scott Cohen in Adobe Research, San Jose. His research interests are in computer graphics, computer vision, and medical imaging.

Ruofeng Tong is a professor in the Department of Computer Science, Zhejiang University, China. He received his B.S. degree from Fudan University, China, in 1991, and Ph.D. degree from Zhejiang University in 1996. His research interests include image and video processing, computer graphics, and computer animation.