A SINGLE LEVEL DETECTION MODEL FOR TRAFFIC SIGN DETECTION USING CHANNEL SHUFFLE RESIDUAL STRUCTURE

Yuanzhi Luo and Jie Hao

College of Computer Science and Technology,
Nanjing University of Aeronautics and Astronautics, China

ABSTRACT

Traffic sign recognition (TSR) is a challenging task for unmanned systems, especially because the traffic signs are small in the road view image. In order to ensure the real-time and robustness of traffic sign detection in automated driving systems, we present a single level detection model for TSR which consists of three core components. The first is we use channel shuffle residual network structure to ensure the real-time performance of the system, which mainly uses low-level features to enhance the representation of small target feature information. Secondly, we use dilated convolution residual block to enhance the receptive field to detect multi-scale targets. Thirdly, we propose a dynamic and adaptive matching method for the anchor frame selection problem of small traffic signs. The experimental surface on Tsinghua-Tencent 100k Dataset and Chinese Traffic Sign Dataset benchmark has better accuracy and robustness compared with existing detection networks. With an image size of 800 × 800, the proposed model achieves 92.9 running at 120 FPS on 2080Ti.

KEYWORDS

Computer Vision, Traffic Sign Detection, Convolutional Neural Networks, Label Assignment.

1. INTRODUCTION

Traffic sign recognition (TSR) is a critical component of Automated Driving Systems (ADS) [1] and High-Definition Maps (HD Map) [2], which can provide TSR aims to help make drivers more aware and able to make better safer driving decisions. Therefore, TSR must fulfill the requirements of high precision and real time. However, TSR still faces big challenges, on the one hand, the complex road conditions and natural environments that appear in the images [3], and on the other hand, most traffic signs in the Tsinghua-Tencent 100k dataset (TT100K) [4] are smaller than 32 × 32 pixels, which means that most traffic signs account for less than 0.3% of the image and detecting small objects is more challenging than large ones.

As a kind of object detection problem, TSR usually shares the same detection algorithms based on convolutional neural networks. On one hand, two-stage approaches such as Fast RCNN [5], Faster RCNN [6], Mask RCNN [7] and Cascade RCNN [8] use region proposals to detect objects. Although these neural networks are superior in terms of accuracy, the low per-frame detection speed limits their application in real-time TSR. On the other hand, one-stage detectors such as SSD [9], YOLO series [10]-[11] [12] [13], YOLOX [14], etc. possess good speedups. Classic detectors still play a great role in TSR. Faster R-CNN [15] and SSD [16] are adapted to reduce the computational complexity for TSR. Yuan [17] et al. extracted regions of interest by adding attention module to CNN to refine the feature extraction of traffic signs in complex

David C. Wyld et al. (Eds): AI, AIMLNET, BIOS, BINLP, CSTY, MaVaS, SIGI - 2022
pp. 81-93, 2022. CS & IT - CSCP 2022 DOI: 10.5121/csit.2022.121808

backgrounds. Zhang [18] et al. introduced image enhancement and spatial pyramid pooling (SPP) modules based on YOLOv3 to effectively fuse low-level and high-level features. These efforts have had some effect, but it is still not enough to satisfy the demands of TSR. This has inspired a series of works targeting lightweight architecture design and better speed-accuracy trade-offs, including MobileNet [19], MobileNetV2 [20], ShuffleNet [21], ShuffleNet V2 [22], and so on. Although efficient, they are still not sufficient for low-dimensional feature extraction. In order to achieve real-time, high accuracy, and we present a single level detection model for TSR which can extract the low-dimensional feature more quickly and efficiently, as shown in Figure 1.

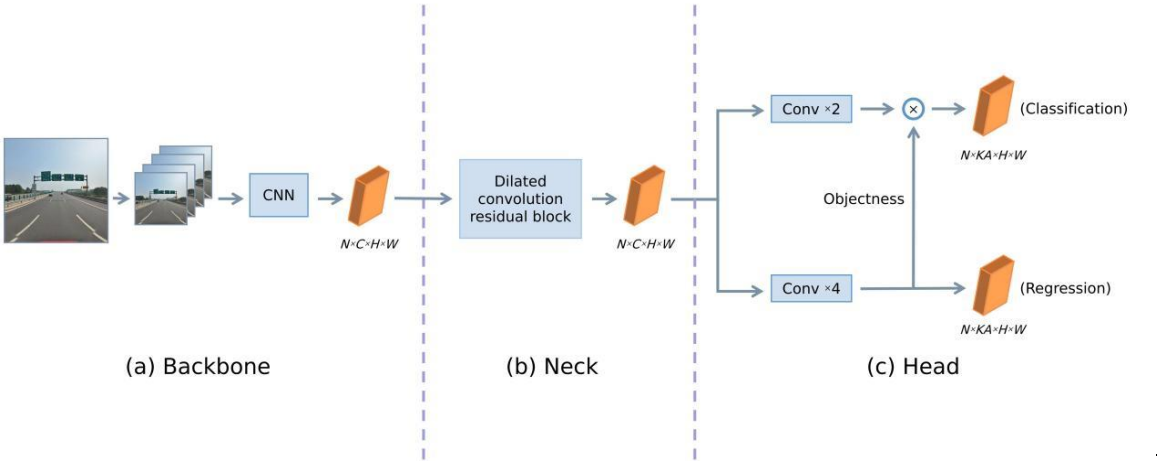


Figure 1. The network structure of TSD.

The main contributions of this paper are as follows:

1) We propose an effective TSR detection model, which uses channel shuffle residual structure for low-level feature extraction, which can effectively reduce the computational overhead of the model.

2) We introduce dilated convolution to increase the receptive field [23] and enhance the feature representation using an attention mechanism to improve the robustness of the model.

3) We design an adaptive matching training samples algorithm to solve the problem of unbalanced distribution of small-scale traffic sign training samples, which can dynamically assign positive anchors according to the targets of different scales to improve the detection effect and robustness of traffic signs.

4) We evaluate our model on the Tsinghua-Tencent 100k dataset (TT100K) [4] and the Chinese traffic sign dataset (CTSD) [24], and the results show that our model achieves outstanding performance in terms of faster speed and higher accuracy compared with state-of-the-art work.

2. PREPARATION

As a typical small object detection problem, the ability to detect small targets is critical for a TSR model [25]. CNNs use multi-level convolution and pooling operations to obtain deeper semantic features, These operations result in small objects existing only at shallow layers, but shallow features are not powerful enough in complex traffic scenes due to the lack of deep semantic

information. Therefore, the researchers have devoted their efforts to improving the small object feature representation capacity, including designing multiscale feature fusion backbones, adding feature pyramids to neural networks [26], and exploiting attention mechanisms [27].

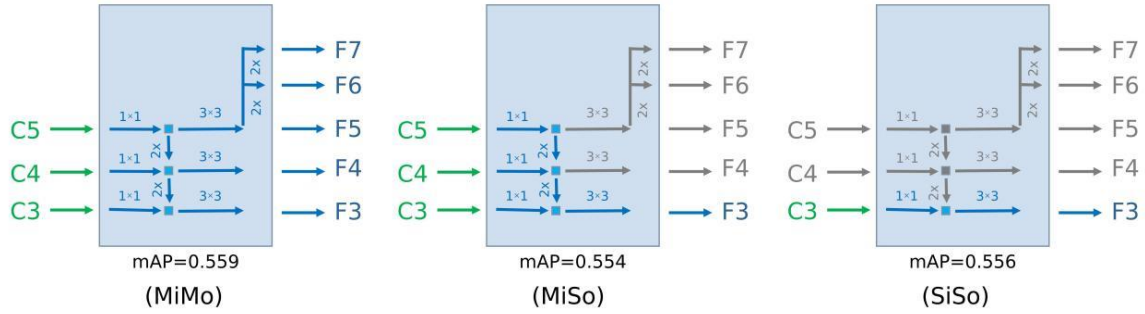


Figure 2. Three structures of FPN in the RetinaNet network. C3~C5 are the input feature layers with downsampling multiplier 8, 16, 32, respectively, and F3~F7 are the output layers after FPN, respectively. The gray lines denote unused channels.

Among all, FPN [26] is a popular module as it can rise the detection accuracy by fusing multi-scale features, in which small scale features contributes the most to TSR. Therefore, before we go to the elaboration of our detector, we first evaluate to what extent of the contribution of FPN in TSR, which we hope can inspire our design of TSR detector. Unfortunately, we found that the depth pyramid level plays a secondary role in TSR. Figure 2 shows the three structures of FPN we've evaluated, and Table. 1 shows the results of FPNs on RetinaNet [28] network and TT100K dataset.

Table 1. The result under TT100K dataset with 800×800 input.

model	mAP	AP_{50}	AP_s	AP_m	AP_l
MiMo	0.559	0.841	0.266	0.690	0.778
MiSo	0.554	0.841	0.283	0.698	0.720
SiSo	0.556	0.847	0.285	0.698	0.750

From Table. 1, we found that MiMo, MiSo, and SiSo achieve comparable performance in TSR, and even SiSo structure is the best for small targets. We speculate that the small target information on C3 feature layer is detailed enough without using the complex information on C4 and C5 layers. Table. 1 demonstrates that FPN is of limited help in TSR. This is also claimed by YOLOF [29]. Motivated by this observation, we design a detector based on the SiSo structure and remove the deep layer to reduce the inference time. Also we redesign the network structure to extract traffic sign features efficiently, and design an adaptive allocation strategy in order to balance the problem of under-allocation of positive samples for small traffic sign training.

3. METHODOLOGY

This section proposes a real-time yet efficient single-stage detection framework based on SiSo structure for TSR, denoted as TSD hereafter. Its schematic diagram is shown in Figure 1. The backbone is used for feature extraction, the neck further processes the features and distributes them to the head, and the head performs the classification and regression tasks and generates the final prediction result. In this section, we will elaborate the three components of TSD respectively.

3.1. Backbone

The residual structure [30] can effectively solve the problem of gradient disappearance and gradient explosion caused by too deep network structure. However, a larger network structure also brings a boosting number of parameters and computational overhead. In order to better achieve the accuracy and real-time performance of TSR, we designed a new residual structure as shown in Figure 3. We replaced the conventional residual structure with CS residual structure, and the parameters were reduced by 36.7% with comparable performance.

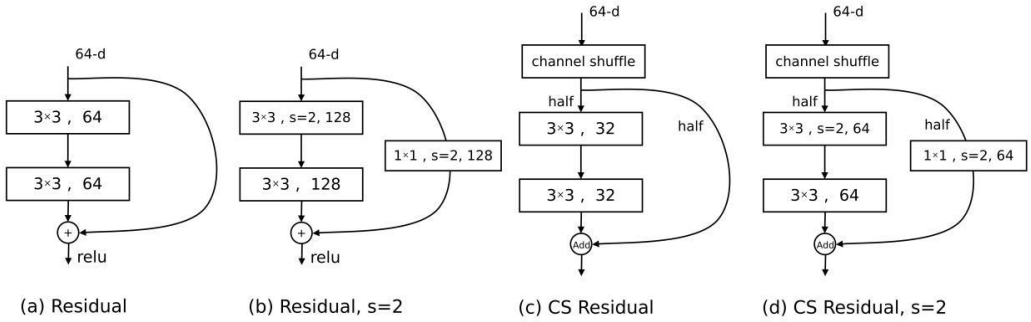


Figure3. Residual structures. (a),(b) are traditional residual structures, c and d are CS residual structures we designed. where 64-d denotes the input dimension, Channel shuffle randomly disrupts the channel, which allows information exchange between channels of the same group.

The network parameters of the entire backbone are shown in Table 2. The Focus module can obtain a two-fold downsampling feature map without information loss, and the CBR module consists of Convolutional, Batchnorm [31], and ReLU activation functions [32].

Table 2. Structure of Backbone.

layer-name	Parametric	Output-size
Input	-	800×800×3
Focus	$\begin{bmatrix} \text{Slice} \\ \text{CBR } 1 \times 1 \end{bmatrix}$	400×400×32
CBR	$\begin{bmatrix} \text{conv } 3 \times 3, \ s = 1 \\ \text{Batchnorm} \\ \text{ReLU} \end{bmatrix}$	400×400×32
CS Residual, s=2	$\begin{bmatrix} \text{half1: CBR } 3 \times 3, s = 2 \\ \text{half2: CBR } 1 \times 1, \ s = 2 \\ \text{half1: CBR } 3 \times 3 \end{bmatrix}$	200×200×64
CS Residual	$\begin{bmatrix} \text{half1: CBR } 3 \times 3 \\ \text{half1: CBR } 3 \times 3 \end{bmatrix}$	200×200×64
CS Residual, s=2	$\begin{bmatrix} \text{half1: CBR } 3 \times 3, s = 2 \\ \text{half2: CBR } 1 \times 1, \ s = 2 \\ \text{half1: CBR } 3 \times 3 \end{bmatrix}$	100×100×128
CS Residual	$\begin{bmatrix} \text{half1: CBR } 3 \times 3 \\ \text{half1: CBR } 3 \times 3 \end{bmatrix}$	100×100×128

3.2. Neck

The input of the neck is the output of backbone, which we first simply process using a 1 × 1 convolution and a 3 × 3 convolution to obtain a 128-channel feature layer. Then, in order to make the output features of neck cover all the targets on various scales, we use the dilated convolution [33] to form the residual block, whose rate is 2, 3, 5 in order, and the channel of the dilated convolution is set to half of the input channel, and we add SElayer [34] to enhance the feature expression after the convolution, and the overall structure of the neck is shown in Figure 4.

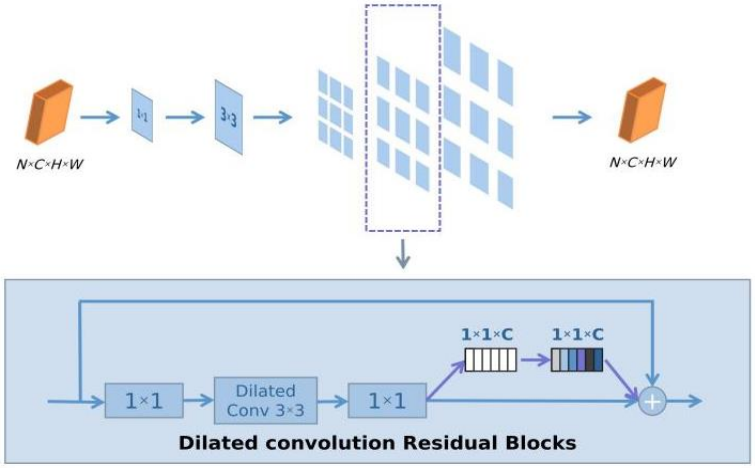


Figure 4. The neck structure. 1×1 and 3×3 represent the size of the convolution kernel, and three consecutive dilated convolution residual blocks can increase the receptive field range, and in the residual blocks we add an attention mechanism to enhance the feature representation, and all convolutional layers are followed by a Batchnorm layer and a ReLU layer.

3.3. Head

For the head, we use two parallel task-specific head compositions: a classification head and a detection head. We follow the design of the FFN in DETR [35], so that the two heads have different number of convolution layers. The classification head has a 1 × 1 convolution and a 3 × 3 convolution, and the regression head has a 1 × 1 convolution and three 3 × 3 convolutions, followed by batch normalization layer and ReLU layer. Also, we follow the design in Auto assign [36] by adding an implicit objectness prediction (without direct supervision) for each anchor on the regression head. The final classification scores for all predictions are generated by multiplying the classification output with the corresponding implied objectivity.

The loss function consists of categorical loss and regression loss, where we use Focal Loss as the category loss function and GIoU Loss as the location loss function. The total loss is shown in Equation 1.

$$\text{Loss} = \gamma_1 \frac{1}{N_{pos}} \sum_i L_{cls}^i + \gamma_2 \frac{1}{N_{pos}} \sum_j L_{reg}^j \quad (1)$$

Where N_{pos} denotes the number of positive samples, i denotes all positive and negative samples, j denotes all positive samples, and γ_1 and γ_2 are the values of the weights learned by the network.

3.4. Label Assignment

The definition of a positive sample is crucial for the optimization of the target detection problem, and in the anchor-based approach, the definition of a positive sample is based on the IoU between the anchor and the GT frame. For example, in RetinaNet [37], an anchor is a positive sample if the maximum IoU between it and the GT frame is greater than a given threshold value of 0.5. This strategy is called Max-IoU matching. When facing small-scale targets and large scale gaps between targets, this methods have certain drawbacks, the large target will match more positive anchors than the small target, which will lead to an unbalanced distribution of positive and negative anchors. Small targets may match few or no positive anchors, which makes the detector pay more attention to the large ground-truth boxes and ignore the small ground-truth boxes during training, resulting in poor detection. Thus we propose an adaptive matching strategy.

Based on the pre-generated anchors, we additionally use the predicted bbox as supplementary information and propose an adaptive matching strategy: calculate the IoU value of the predefined anchor and GT, select the IOU value greater than 0.15 At the same time, we use the predicted bbox with the IoU value of GT greater than 0.15 as a supplementary candidate positive sample. Then we calculate the mean and standard deviation based on the total candidate positive samples to obtain a threshold, and then use the threshold to filter the final set of positive samples. The specific algorithm is as follows:

Algorithm 1 Adaptive matching training samples

Input:
G is a set of ground-truth boxes on the image
A is a set of all anchor boxes
P is a set of all Prediction boxes
Y is the initial IoU threshold with a default value of 0.15
Output:
O is a set of positive samples
N is a set of negative samples
I is a set of ignore samples
Detail:

for each ground-truth $g \in G$ **do**
  build an empty set for candidate positive samples of the ground-truth g: C_g;
  for each $a_i \in A$ **do**
    compute IoU between g and a_i :$A_g = \mathrm{IoU}(a_i, g)$;
    if $A_g = \mathrm{IoU}(a_i, g) > Y$ **then**
      $C_g = C_g \cup a_i$;
    end if
  end for
  for each $p_i \in P$ **do**
    compute IoU betweengandp_i :$P_g = \mathrm{IoU}(p_i, g)$;
    if $P_g = \mathrm{IoU}(p_i, g) > Y$**then**
      $C_g = C_g \cup p_i$;
    end if
  end for
  compute IoU between C_g and g:$D_g = \mathrm{IoU}(C_g, g)$;
  compute mean ofD_g : $m_g = \mathrm{Mean}(D_g)$;
  compute standard deviation of D_g: $v_g = \mathrm{Std}(D_g)$;

```
coumpute IoU threshold for ground-truth g: t_g = m_g + v_g;
for each candidate c ∈ C_g do
    if IoU(c, g) ≥ t_g then
        P_2 = P_2 ∪ c;
    elif Y ≤ IoU(c, g) ≤ t_g then
        I = I ∪ c;
    end if
end for
end for
N = A − P_2 − I;
return P_2, N, I;
```

4. EXPERIMENTS AND RESULTS

We evaluated our TSD on TT100k and CTSD, using the COCO benchmark [38]. Most of the category in the TT100K are less than 100, and to make full use of the annotation information, we divided both datasets into three categories (danger, prohibitory, mandatory). We divide all traffic signs into small, medium, and large groups, which can compare the performance of traffic signs at different scales in more detail. We compare with some currently popular lightweight detection frameworks, such as YOLOv4-Tiny [39]. A description of the datasets and specific experimental details are given below.

4.1. Datasets

CTSD has 700 annotated images in the training set and 400 in the test set, and the traffic signs are classified into three categories (danger, prohibitory, mandatory). The training set of TT100K includes 6105 labeled images and the test set has 3071 images, with a total of 232 categories. Considering many of them have very few, and in order to make full use of their labeling information, we process them into the same three categories as CTSD, As shown in Figure 5. For TT100K, we divided the traffic signs into three categories: small, medium, and large according to their bounding box sizes, with those smaller than 32 × 32 being small targets, those with sizes ranging from 32 × 32 to 96×96 being medium targets, and those larger than 96×96 being large targets. The original annotation of the training set of TT100K consists of 51.43% small targets, 41.45% medium targets, and 7.12% of large targets, as shown in Figure 6. When we resize it to an image size of 800 × 800 for training, the percentages of small, medium, and large targets are 89.28%: 10.69%: 0.04%, respectively. Table 3 contains the image size and annotation size for both datasets.

Table 3. Details of TT100K and CTSD.

Dataset	train images	test images	image size(px)	sign size(px)
TT100K	6105	3071	2048×2048	8×9~285×343
CTSD	700	400	768×1024 ~720×1280	20×20 ~ 380×378

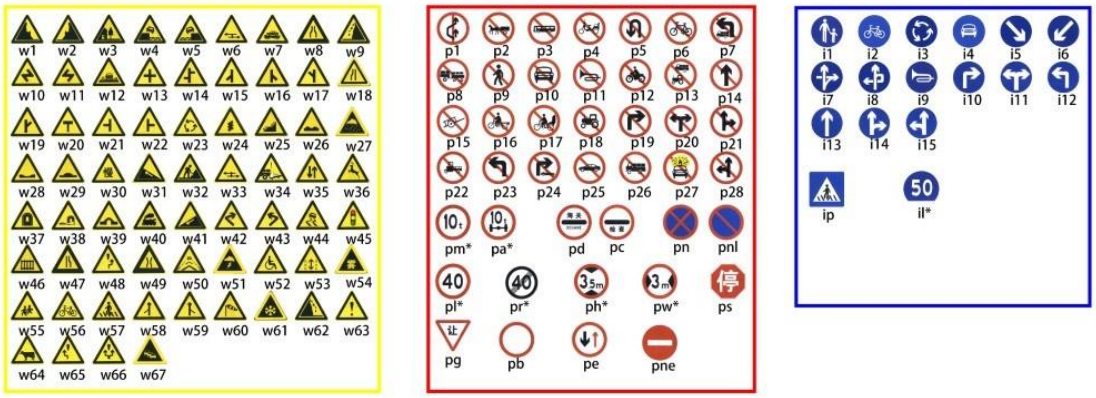


Figure 5. Traffic sign categories in TT100K.

4.2. Experiment Environment

All experiments in this paper are conducted on Ubuntu 20.04LTs system with GeForce RTX 2080Ti, based on MMdetection [40] framework, except for SSDLite where the input image size is 320×320. The rest of the image preprocessing is exactly the same, each image is resized to 800×800, RandomFlip, RandomShift, Normalize. RandomFlip, RandomShift, Normalize. Batchsize is set to 8, and we adopt the learning rate setting in DETR [28], the initial learning rate is set to 0.02, and a smaller learning rate is set in the backbone network, i.e., 1/3 of the initial learning rate to stabilize the training at the beginning. We set the warmup times as 1500 times. For model inference, we post-processed the results using an NMS with a threshold of 0.5.

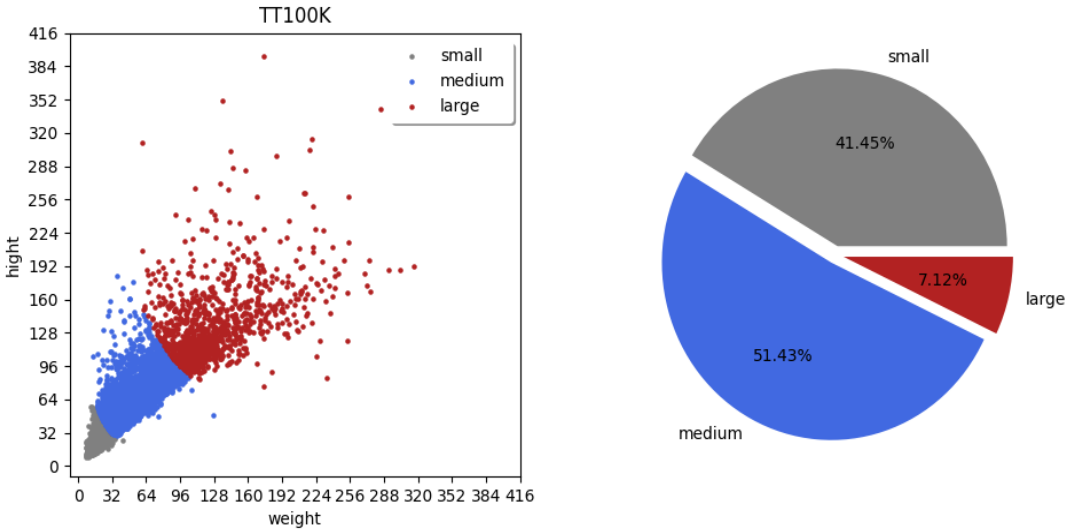


Figure 6. Sample distribution in the used dataset. The left figure shows the two-dimensional distribution of large, medium and small targets in the original annotation of TT100K training set, and the right one shows the percentage of large, medium and small targets in the training set.

4.3. Comparison With Previous Work

We used AP (Average Precision) (AP_s, AP_m, AP_l), AP_{50}, and AP_{75}which were introduced by MS COCO [38] benchmark to evaluate the accuracy and the number of images processed per second (FPS) to evaluate the inference speed. Since the image size in CTSD is not uniform, FPS metric is not evaluated. The results are shown in Table 4 and Table 5. The readers can also refer to Figure 7 for a more intuitive comparison result.

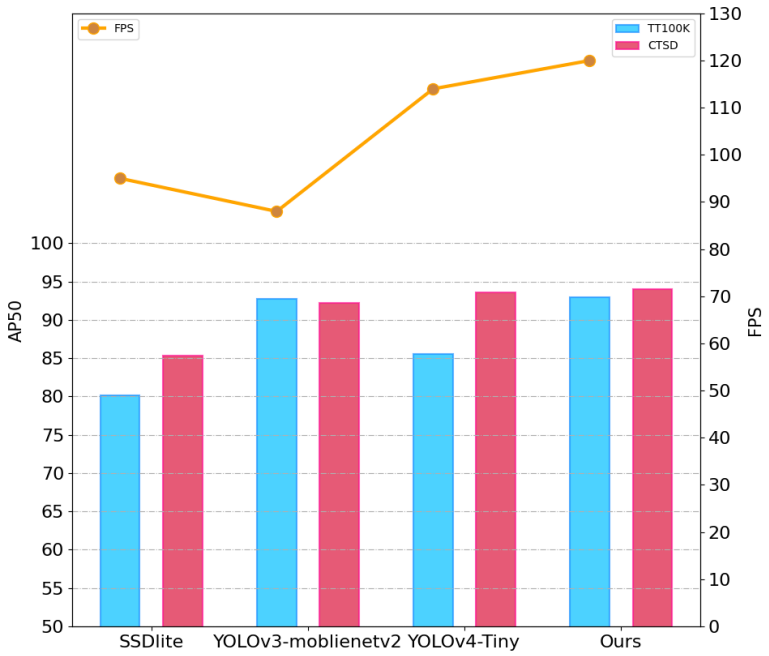


Figure 7. Compare our method with the previous method on TT100K and CTSD.

We can see that our detector can achieve the best results in most cases. Both YOLOv3-MobileNetV2 and YOLOv4-Tiny used the Kmeans method [41] to obtain the anchor sizes, and the anchor settings for TT100K and CTSD were [[(73, 73), (110, 111), (157, 154)], [(31, 31), (42, 41), (56, 55)], [(16, 16), (20, 20), (25, 25)]] and [[(23, 25), (30, 32), (43, 45)], [(12, 14), (15, 16), (19, 20)], [(6, 7), (8, 9), (10, 11)]].

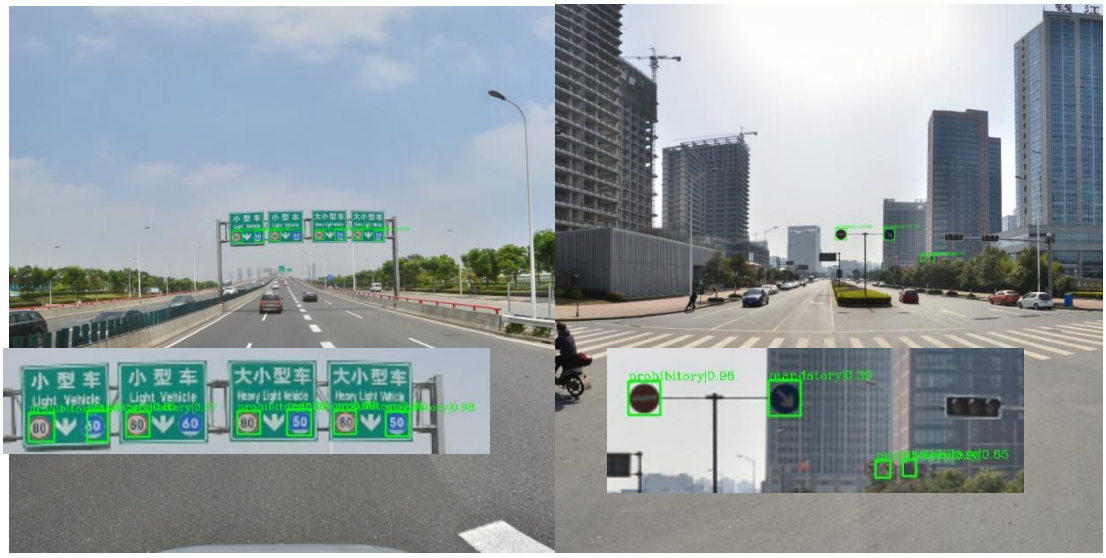
Table 4. Comparison of our method with other methods on TT100K.

method	mAP	AP_{50}	AP_s	AP_m	AP_l	FPS
SSDLite	0.460	0.802	0.264	0.582	0.635	95
YOLOv3-MobileNetV2	0.545	0.927	0.455	0.597	0.602	88
YOLOv4-Tiny	0.437	0.855	0.264	0.530	0.612	114
TSD (Ours)	**0.643**	**0.929**	**0.498**	**0.717**	**0.746**	**120**

In terms of accuracy, our method achieves the highest AP metric of 92.9% on TT100K, which is 7.4% higher than YOLOv4-Tiny, which indicates that our method has good results for small object detection. In the CTSD, YOLOv4-Tiny is comparable to our method. Please note, TSD is able to reach 94.3% if inputting 1024×1024 images for inference.

Table 5. Comparison of our method with other methods on CTSD.

method	mAP	AP_{50}	AP_s	AP_m	AP_l
SSDLite	0.538	0.853	0.268	0.576	0.563
YOLOv3-MobileNetV2	0.644	0.922	0.411	0.687	0.532
YOLOv4-Tiny	0.722	0.936	0.430	0.767	**0.765**
TSD (Ours)	**0.736**	**0.940**	**0.556**	**0.771**	0.731



(a_1) (a_2)

(b_1) (b_2)

Figure 8. Partial visualization results, a_1 and a_2 are the visualization results of Yolov4-Tiny, b_1 and b_2 are our results.

In terms of inference speed, we test on TT100k with input image size of 800×800 for inference, our method has FPS of 120, which is slightly faster than YOLOv4-Tiny and fully satisfies the real-time requirement.

To show the results of our experiments more visually, Figure 8 shows the visualization of some tested samples on YOLOv4-Tiny and TSD.

4.4. Ablation Experiment

We investigated the effectiveness of the backbone and neck components of TSD, and all experiments were performed on TT100K.

In Table 6, we replace the CS Residual structure of the backbone part with the conventional residual structure, and we can see that CS Residual structure achieves comparable performance with the conventional residual structure, but CS Residual structure parameters were reduced by 70% and FLOPs by 61.8% when the output channel of the backbone is 128.

Table 6. backbone uses different residual structures for comparison.

Backbone	AP_{50}	Params(M)	FLOPs (GFLOPs)
Residual	0.929	0.667	12.101
CS Residual(ours)	0.929	0.2 (70% ↓)	4.621 (61.8%↓)

We tested different choices of dilation for the expansion convolution in the neck part, and the results are shown in Table 7. We can see that the difference in model results is not significant for different choices of dilation, and we guess the reason is that the target scale of TT100K is small, and the small receptive field is enough to cover most of the targets in the image. We learned that the grid effect occurs when the dilation of the dilated convolution is greater than 1 [33]. To ensure the robustness of the model, we set the dilation to 2, 3, 5.

Table 7. Different dilations，RF means receptive field.

Dilation	AP_{50}	AP_s	AP_m	AP_l	RF
1,2,3	0.926	0.490	0.713	0.748	13x13
2,2,2	0.925	0.492	0.711	0.751	13x13
1,2,5	0.924	0.486	0.712	0.761	17x17
2,3,5	**0.929**	0.498	0.717	0.746	21x21
2,4,6	0.921	0.479	0.710	0.754	25x25

5. CONCLUSIONS

In this work, we aim to improve the speed and accuracy of small traffic sign detection, and we find that FPN have limited improvement in detection capability for small targets. We propose an effective framework TSD for small TSR, detailing various optimization strategies, including for proposing CS Residual structure, Dilated convolution blocks and adaptive matching training sample methods. Experiments show that our method outperforms some lightweight detection algorithms on TT100K and CTSD.

Our model has some limitations when facing the detection of very large targets, because we focus more on the feature information within a certain range.If you need to, we suggest that you can further increase the range of receptive field or create another branch to detect large objects.

For future work, we will try to optimize the detection algorithm using the anchor-free approach, which can eliminate our customization work for anchor and enhance the robustness of the model.

ACKNOWLEDGEMENTS

This work is supported in part by the National Key R&D Program of China under Grant 2019YFB2102000, and in part by the Collaborative Innovation Center of Novel Software Technology and Industrialization.

REFERENCES

[1] A H U , A N K , B S Y . A Safety Knowledge Representation of the Automatic Driving System[J]. Procedia Computer Science, 2016, 96:869-878.
[2] Takeuchi E , Yoshihara Y , Ninomiya Y . Blind Area Traffic Prediction Using High Definition Maps and LiDAR for Safe Driving Assist[C]// IEEE International Conference on Intelligent Transportation Systems. IEEE, 2015.
[3] Muhammad K , Ullah A , Lloret J , et al. Deep Learning for Safe Autonomous Driving: Current Challenges and Future Directions[J]. IEEE Transactions on Intelligent Transportation Systems, 2020, PP(99):1-21.
[4] Zhe Z , Liang D , Zhang S , et al. Traffic-Sign Detection and Classification in the Wild[C]// 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2016.
[5] Girshick R. Fast R-CNN[J]. Computer Science, 2015.
[6] Ren S, He K, Girshick R, et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2017, 39(6):1137-1149.
[7] He K ,Gkioxari G , P Dollár, et al. Mask R-CNN[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2017.
[8] Cai Z , Vasconcelos N . Cascade R-CNN: Delving into High Quality Object Detection[J]. 2017.
[9] Liu W ,Anguelov D , Erhan D , et al. SSD: Single Shot MultiBox Detector[J]. 2015.
[10] Redmon J ,Divvala S , Girshick R , et al. You Only Look Once: Unified, Real-Time Object Detection[J]. IEEE, 2016.
[11] Redmon J , Farhadi A . YOLO9000: Better, Faster, Stronger[J]. IEEE, 2017:6517-6525.
[12] Redmon J , Farhadi A . YOLOv3: An Incremental Improvement[J]. arXiv e-prints, 2018.
[13] BochkovskiyA , Wang C Y , Liao H . YOLOv4: Optimal Speed and Accuracy of Object Detection[J]. 2020.
[14] Ge Z , Liu S , Wang F , et al. YOLOX: Exceeding YOLO Series in 2021[J]. 2021.
[15] Zuo Z , Kai Y , Qiao Z , et al. Traffic Signs Detection Based on Faster R-CNN[C]// 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW). IEEE, 2017.
[16] You S , Bi Q , Ji Y , et al. Traffic Sign Detection Method Based on Improved SSD[J]. Information, 2020, 11.
[17] Yuan Y ,Xiong Z , Wang Q . VSSA-NET: Vertical Spatial Sequence Attention Network for Traffic Sign Detection[J]. IEEE Transactions on Image Processing, 2019:1-1.
[18] Zhang H , Qin L , Li J , et al. Real-Time Detection Method for Small Traffic Signs Based on Yolov3[J]. IEEE Access, 2020, PP(99):1-1.
[19] Howard A G , Zhu M , Chen B , et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications[J]. 2017.
[20] Sandler M , Howard A , Zhu M , et al. MobileNetV2: Inverted Residuals and Linear Bottlenecks[J]. IEEE, 2018.
[21] Zhang X , Zhou X , Lin M , et al. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices[J]. 2017.

[22] Ma N , Zhang X , Zheng H T , et al. ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design[J]. Springer, Cham, 2018.
[23] Li Y , Chen Y , Wang N , et al. Scale-Aware Trident Networks for Object Detection[C]// 2019 IEEE/CVF International Conference on Computer Vision (ICCV). IEEE, 2019.
[24] Yi Y , Luo H , Xu H , et al. Towards Real-Time Traffic Sign Detection and Classification[J]. IEEE Transactions on Intelligent Transportation Systems, 2016, 17(7):2022-2031.
[25] Deng C . A Review on the Extraction of Region of Interest in Traffic Sign Recognition System[C]// 2020 International Conference on Computing and Data Science (CDS). 2020.
[26] Lin T Y , Dollar P , Girshick R , et al. Feature Pyramid Networks for Object Detection[J]. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.
[27] Vaswani A ,Shazeer N , Parmar N , et al. Attention Is All You Need[C]// arXiv. arXiv, 2017.
[28] Lin T Y , Goyal P , Girshick R , et al. Focal Loss for Dense Object Detection[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2017, PP(99):2999-3007.
[29] Chen Q , Wang Y , Yang T , et al. You Only Look One-level Feature[J]. 2021.
[30] He K , Zhang X , Ren S , et al. Deep Residual Learning for Image Recognition[C]// 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2016.
[31] Ioffe S, Szegedy C . Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift[J]. JMLR.org, 2015.
[32] GlorotX ,Bor De S A , Bengio Y . Deep Sparse Rectifier Neural Networks[C]// Journal of Machine Learning Research. 2011:315-323.
[33] Yu F ,Koltun V , Funkhouser T . Dilated Residual Networks[J]. IEEE Computer Society, 2017.
[34] Jie H , Li S , Gang S , et al. Squeeze-and-Excitation Networks[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017, PP(99).
[35] CarionN , Massa F , Synnaeve G , et al. End-to-End Object Detection with Transformers[J]. 2020.
[36] Zhu B , Wang J , Jiang Z , et al. AutoAssign: Differentiable Label Assignment for Dense Object Detection[J]. 2020.
[37] Lin T Y , Goyal P , Girshick R , et al. Focal Loss for Dense Object Detection[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2017, PP(99):2999-3007.
[38] Lin T Y , Maire M , Belongie S , et al. Microsoft COCO: Common Objects in Context[C]// European Conference on Computer Vision. Springer International Publishing, 2014.
[39] BochkovskiyA , Wang C Y , Liao H . YOLOv4: Optimal Speed and Accuracy of Object Detection[J]. 2020.
[40] Chen K , Wang J , Pang J , et al. MMDetection: Open MMLab Detection Toolbox and Benchmark[J]. 2019.
[41] Bahmani B , Moseley B , Vattani A , et al. Scalable K-Means++[J]. Proceedings of the VLDB Endowment, 2012, 5(7):622-633.

AUTHORS

Yuanzhi Luo is a graduate student at Nanjing University of Aeronautics andAstronautics, College of Computer Science and Technology. His research interests include deep learning, image processing, and computer vision.

Jie Hao received her BS degree from Beijing University of Posts and Telecommunications, China, in 2007, and the Ph.D. degree from University of Chinese Academy of Sciences, China, in 2014. From 2014 to 2015, she has worked as post-doctoral research fellow in the School of Computer Engineering, Nanyang Technological University, Singapore. She is currently an Assistant Professor at College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China. Her research interests are ubiquitous perception, multimodal data fusion and Internet of Things.