Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 5761414, 15 pages
https://doi.org/10.1155/2020/5761414

Research Article

Fast Vehicle and Pedestrian Detection Using Improved Mask R-CNN

Chenchen Xu,[1] Guili Wang,[1,2] Songsong Yan,[1] Jianghua Yu,[3] Baojun Zhang,[1] Shu Dai,[1] Yu Li,[1] and Lin Xu[4]

[1]*School of Physics and Electronic Information, Anhui Normal University, Wuhu 241002, China*
[2]*Anhui Provincial Engineering Laboratory on Information Fusion and Control of Intelligent Robot, Wuhu Anhui, 241002, China*
[3]*School of Communications and Information Engineering, Xi'an University of Posts & Telecommunications, Xi'an 710061, China*
[4]*School of Mathematics and Statistics, Anhui Normal University, Wuhu, 241002, China*

Correspondence should be addressed to Guili Wang; xlpyhwgl@ahnu.edu.cn

Received 16 February 2020; Revised 16 April 2020; Accepted 24 April 2020; Published 31 May 2020

Guest Editor: Feng-Jang Hwang

This study presents a simple and effective Mask R-CNN algorithm for more rapid detection of vehicles and pedestrians. The method is of practical value for anticollision warning systems in intelligent driving. Deep neural networks with more layers have greater capacity but also have to perform more complicated calculations. To overcome this disadvantage, this study adopts a Resnet-86 network as a backbone that differs from the backbone structure of Resnet-101 in the Mask R-CNN algorithm within practical conditions. The results show that the Resnet-86 network can reduce the operation time and greatly improve accuracy. The detected vehicles and pedestrians are also screened out based on the Microsoft COCO dataset. The new dataset is formed by screening and supplementing COCO dataset, which makes the training of the algorithm more efficient. Perhaps, the most important part of our research is that we propose a new algorithm, Side Fusion FPN. The parameters in the algorithm have not increased, the amount of calculation has increased by less than 0.000001, and the mean average precision (mAP) has increased by 2.00 points. The results show that, compared with the algorithm of Mask R-CNN, our algorithm decreased the weight memory size by 9.43%, improved the training speed by 26.98%, improved the testing speed by 7.94%, decreased the value of loss by 0.26, and increased the value of mAP by 17.53 points.

1. Introduction

To improve driving safety and reduce driver fatigue, research is being conducted on the development of intelligent driving technology [1]. In intelligent driving, we need to first guarantee the human's safety, and therefore, the assisted driving system (ADS) [2] to improve safety is a hot spot in intelligent driving research. The collision avoidance warning system (CAWS) [3] is particularly important for ADS in smart cars. One key issue of CAWS is the awareness of the driver's surroundings. The images of vehicles and pedestrians captured by car cameras are to be identified, detected, and divided by object detection technology, which faces challenges due to complex scene information.

The two main methods for vehicles and pedestrians detection are machine learning-based [4] approaches and deep-learning-based [5] approaches. Machine learning approaches first define features using one of the feature acquisition descriptors such as histogram of oriented gradient (HOG) [6] and then perform classification using a technique such as a support vector machine (SVM) [7]. The HOG + SVM approach shows superior performance but suffers from low mean average precision (mAP) and is not suitable for multistage process feature extraction [8]. Deep learning systems, such as convolutional neural networks (CNNs), show superiority in object detection because they aim to discover discriminative features from raw data [9]. The CNN was developed in the 1980s and 1990s [10], but since experiencing a resurgence of interest [11] in 2012, it has established a foothold in the field of computer vision and has grown at a rapid pace.

As the requirements for algorithm accuracy and speed continue to increase, vehicle and pedestrian recognition

algorithms such as R-CNN, Fast R-CNN, Faster R-CNN, and Mask R-CNN have been proposed [12–15]. Mask R-CNN was proposed in 2017. Without adding any skills, Mask R-CNN outperformed all the single recognition models at the time and defeated the 2016 champion in the Microsoft COCO dataset [16] challenge. It became the leader in image recognition, detection, and segmentation [15]. In view of the fact that Mask R-CNN has the advantages of fast speed and high accuracy in target detection tasks, it is applied to many fields [17, 18].

Mask R-CNN consists of three parts: feature pyramid network (FPN), regional proposal network (RPN), and detection. It can accomplish three tasks: target recognition [19], detection [20], and segmentation [21]. Mask R-CNN is dramatically driving the development of computer vision, leading to a series of proposed algorithms. The FPN is the first part of Mask R-CNN, which uses Resnet-101 [22] as its backbone to detect 81 types of targets. The depth of network structure is very important for many visual recognition tasks. Resnet is a residual learning framework to ease the training of networks that reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions.

Aiming at the research target of this paper, which is vehicles and pedestrians on the road, the following four improvements are made based on Mask R-CNN. First, in order to obtain more precise feature semantic information combined with the characteristics of semantic feature graph information, the side fusion calculation is added in the FPN structure, and the Side Fusion FPN (SF-FPN) algorithm is proposed. The second is to improve the Resnet-101 network to Resnet-86 network. The residual block is reduced from 23 to 18, which effectively improves the calculation speed. The third is to select 500 RPN training frames and 250 prediction frames to further reduce algorithm redundancy. The fourth is to classify and supplement the 81 samples in the COCO dataset [23] and change the rectangular label in the original image to a polygon label to improve the training effect.

2. Mask R-CNN

Mask R-CNN is a conceptually simple, flexible, and general framework for object recognition, detection, and instance segmentation, which can efficiently detect objects in an image, while generating a high-quality segmentation mask for each instance. Feature pyramid networks (FPNs) for object detection [24], the first block structure of Mask R-CNN, are responsible for feature extraction. The regional proposal network (RPN) [25], the second piece of Mask R-CNN, shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals [26]. We then expanded the Faster R-CNN to form the Mask R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition.

The RPN was applied to Mask R-CNN instead of selective search [27] so that the RPN can share the convolution feature of the full map with the detection network. It can predict both boundary position and object scores at each location, and it is also a fully convolutional network (FCN) [28]. As shown in Table 1, the Faster R-CNN uses the RPN as a region generation network to generate candidate regions. The FPS based on the Fast R-CNN algorithm is as high as 5, and its MAP tested on VOC 2012 is also increased to 70.4% [14].

To further improve the detection accuracy of the target, Mask R-CNN uses the bilinear interpolation algorithm region of interest (ROI) align instead of ROI pool [29] on the basis of Faster R-CNN. The ROI align layer removes the harsh quantization of the ROI pool and properly aligns the extracted features with the input. This method of ROI align avoids any quantization of the ROI boundaries or bins. The algorithm of ROI align is used to compute the exact values of the input features based on bilinear interpolation [30] at four regularly sampled locations in each ROI bin and aggregate the results. This method improves the accuracy of Mask R-CNN by 10% [15].

In order to enable Mask R-CNN to implement the mask function, Mask R-CNN adds mask branches to achieve high-precision instance segmentation from pixel-to-pixel alignment. Mask R-CNN can accomplish three tasks: target recognition, detection, and segmentation. Its detection speed can still reach 5 FPS. The flowchart of Mask R-CNN is shown in Figure 1. At the input, after the image passes through the FPN, five sets of feature maps of different sizes are generated, and the candidate frame area is generated by the RPN. After the candidate region is combined with the feature map, the system can achieve the detection, classification, and mask of the target. To further improve the computing speed of the algorithm, it can adapt to the real-time requirements of the intelligent driving anticollision warning system.

Based on Mask R-CNN, we propose a method to improve the detection of accuracy and speed through SF-FPN with Resnet-86. In this study, the dataset, FPN structure, and RPN parameter settings are improved. The improved method proposed in this study can realize the recognition, detection, and segmentation of the target at the same time.

3. Improvement Based on Mask R-CNN

3.1. Feature Pyramid Networks for Object Detection (FPN). Feature extraction is an important part of the field of machine vision. With the development of machine learning, methods based on neural network feature extraction, including featurized image pyramid [31], single feature map [32], pyramidal feature hierarchy [33], and feature pyramid network (FPN), have been proposed.

As shown in Figure 2(a), featurized image pyramid refers to an image input that passes through different convolutional layers, sets convolution kernels of different sizes, generates multiscale feature maps, and outputs feature maps of different sizes. Although this method obtains feature maps at different scales, it adds a large amount of calculation time and the semantic information from the feature map is not sufficient. Figure 2(b) shows a method of single feature map feature extraction. The idea is to input an image and pass different convolutional layers from bottom to top, with the

Table 1: Frame comparison of R-CNN, Fast R-CNN, Faster R-CNN, and Mask R-CNN.

Network framework	R-CNN	Fast R-CNN	Faster R-CNN	Mask R-CNN
Propose time	2014	2015	2016	2017
Region proposal	Selective search	Selective search	RPN	RPN
Feature extraction	CNN	CNN + ROI pool	CNN + ROI pool	CNN + ROI align
Feature classification	SVM			
Function	Classification, detection	Classification, detection	Classification, detection	Classification, detection, segmentation
Test time per image (with proposals)	47 seconds	2 seconds	0.2 seconds	0.2 seconds
mAP (VOC 2012)	62.4%	68.4%	70.4%	—

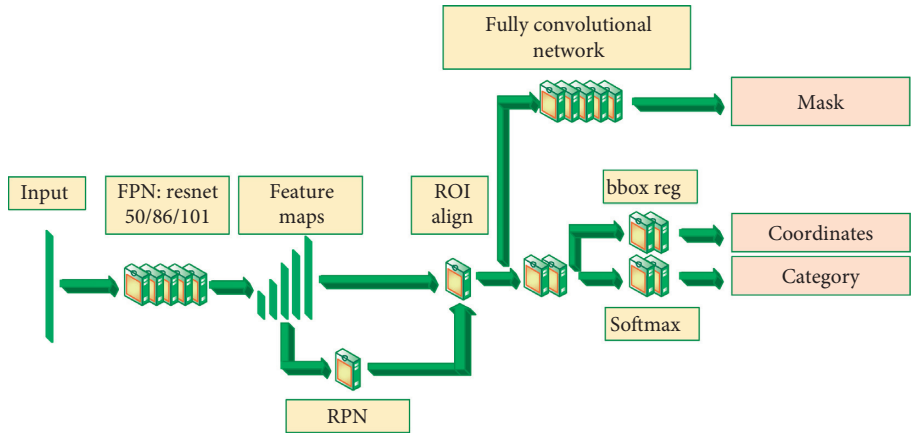


Figure 1: The Mask R-CNN framework flowchart.

output of the last convolutional layer used as the final feature output of the network. This method offers faster operation and utilizes the semantic information on each layer. For this reason, it has been previously applied to SPP-net, Fast R-CNN, and Faster R-CNN. But the performance of multiscale target detection is poor. Figure 2(c) shows that the pyramidal feature hierarchy method is still about inputting a picture from the bottom to top and passing through different convolutional layers. But it extracts different scale features of different layers as predictions, which will not increase the amount of calculation and can obtain multiscale features. Although the pyramidal feature hierarchy method can maintain speed and generate multiscale feature information at the same time, this method can neither make full use of lower level semantic information nor achieve good results of small target detection. To this end, the FPN adds a side link algorithm based on the pyramidal feature hierarchy algorithm, that is, when inputting one image, it passes through different convolutional layers from the bottom up and then links from top-down at side and combines low resolution and strong semantic features with high resolution and weak semantic features. It not only maintains the original calculation speed but also generates accurate multiscale feature information.

The top-down pyramid [34] algorithm in Figure 2(d) uses convolutional sampling to first reduce the size and then uses upsampling to increase the feature map. The network has no horizontal connection, that is, the top-down process does not integrate the original features, which will cause the location characteristics of the target to become more inaccurate after multiple downsampling and upsampling processes. The finest level only [35] algorithm shown in Figure 2(e) is capable of taking only the last layer P2 of PFN as the output and does not produce multiscale output. The sliding of the RPN stage window at different layers of the pyramid will increase the robustness of the scale change, so the FPN is useful for identifying different sizes of the target in robustness, which is significantly better than the finest level only algorithm.

As can be seen from Table 2, compared to the outputs of C4 and C5 of the featured image pyramid algorithm, the FPN algorithm improves the accuracy by nearly 21.7%. Particularly in the small target detection, it increases a significant advantage of 12.9 points. From Figure 2, we can also see that the FPN adds top-to-bottom side links and multiscale output compared with the single feature map and pyramidal feature hierarchy algorithm. In this way, low resolution and strong semantic features can be fully integrated with high resolution and weak semantic features. From the data table, we can see that when 1000 anchors are generated as predictions, the AR is improved by 6.8 points on average, of which 14.4 points are improved on the detection result of small targets. Compared with the top-down pyramid algorithm, the FPN adds horizontal connection and multiscale output, which increases 10.2 points in AR, including 18.4 points for small target detection. Finally, compared with the final level algorithm, FPN increases multiscale output and improves 4.0 points in AR. The small

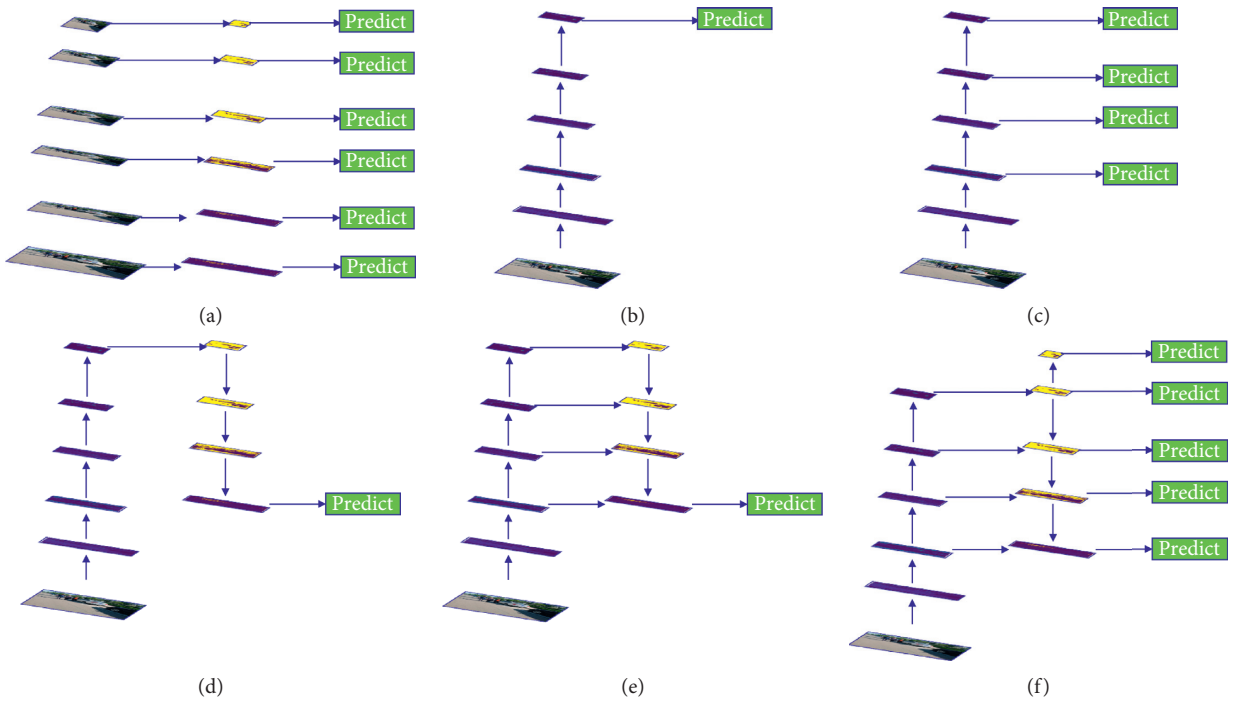


Figure 2: Six network structures for feature extraction. (a) Featurized image pyramid. (b) Single feature map. (c) Pyramidal feature hierarchy. (d) Top-down pyramid. (e) Finest level only. (f) Feature pyramid network.

Table 2: Feature extraction network data comparison.

Network structure	Feature	Anchors	Lateral	Top-down	AR^{1k}	AR_s^{1k}	AR_m^{1k}	AR_l^{1k}
Featurized image pyramid	C4	47k	No	No	48.3	32.0	58.7	62.2
Featurized image pyramid	C5	12k	No	No	44.9	25.3	55.5	64.2
Bottom-up pyramid	$\{P_k\}$	200k	Yes	No	49.5	30.5	59.9	68.0
Top-down pyramid	$\{P_k\}$	200k	No	Yes	46.1	26.5	57.4	64.7
Finest level only	P2	200k	Yes	Yes	51.3	35.1	59.7	67.6
FPN	$\{P_k\}$	**200k**	**Yes**	**Yes**	**56.3**	**44.9**	**63.4**	**66.2**

The first column is the feature extraction algorithm, where the second and third rows are the two-layer outputs of the algorithm, and the fourth row contains the single feature map and pyramidal feature hierarchy. The second column is the name of the output layer, where the "{}" symbol indicates the independent prediction at each layer. The evaluation standard uses average recall (AR). The number in the upper right corner of the AR indicates the number of anchors generated by each image. The letters "s," "m," and "l" in the bottom right corner denote the small goal, medium goal, and large goal, respectively.

targets have improved by 9.8 points, which greatly improves the robustness.

The FPN in Mask R-CNN uses Resnet-101 as the backbone. The design of the deep residual network overcomes the problem that the learning efficiency becomes lower due to the deepening of the network and the inability to effectively improve accuracy. The deep residual network divided a training series into one block for training so that the error of each block is minimized in order to achieve the goal of the smallest overall error. Resnet-101 is an internationally used classical deep residual network. It can be roughly divided into five stages of convolutional layers. The output scale is reduced by half at each stage. This FPN + Resnet network has robustness and adaptability and can not only send high-level features to low-level features but also make full use of all high-level feature information and underlying feature information through side links, thereby improving feature extraction capabilities. The FPN is the first part of Mask R-CNN, which can obtain feature maps. As shown in Figure 3, the FPN is built on the basis of image pyramids [36]. Its input image is obtained from the convolutional layer to obtain five sets of characteristic maps (C1, C2, C3, C4, and C5), and all five are upconverted or reduced to 256 dimensions by a $1 * 1$ size convolution kernel. Since the upsampling of C5 is as same as the one of C4, we thus use the dimensionality reduction results of C4 directly. This connection method can connect the high-level features of low resolution and high-semantic information and the low-level features of high resolution and low-semantic information from top to bottom so that the features of all scales have rich semantic information. The same is true for the connection method of P5, P4, P3, and P2. In order to make the large target detection effect better, P5

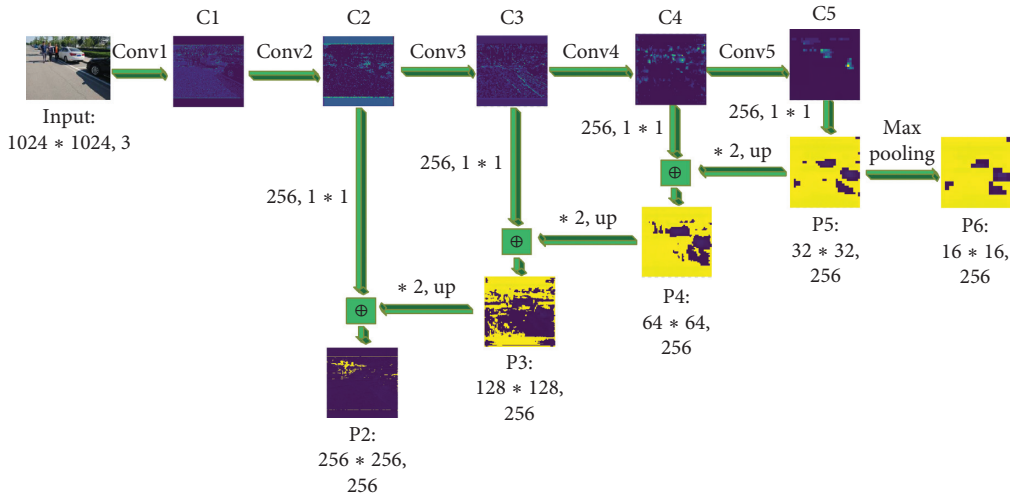


Figure 3: Network structure diagram of FPN + Resnet.

performs the maximum pooling at the end and forms a feature map P6 of 16 ∗ 16 size.

The COCO datasets contain 81 categories. The FPN in Mask R-CNN uses Resnet-101 as the backbone to detect 81 types of targets. However, since the detection target in this study consists of only three categories—person, car, and bus—there is a problem of parameter redundancy when using Resnet-101 to detect three types of targets. To reduce the redundancy and improve the computing speed, this study designs a Resnet-86 with only 86 layers as the backbone of the network and FPN to detect the three types of targets. As can be seen from Table 3, the Resnet-86, Resnet-50, and Resnet-101 structures are all composed of five-part convolutional layers. The number of residual blocks of Resnet-50 and Resnet-101 at Conv_4 are 6 and 23, respectively. In this study, the number of residual blocks of Resnet-86 at Conv_4 is changed to 18. It can be seen from the experimental results in Table 4 and Figure 4 that although the Resnet-50 structure is faster in recognition speed, its recognition accuracy cannot meet our requirements. Compared with Resnet-101, Resnet-86 not only increases the computing speed by about 7.94% but also reduces the weight memory by 9.43%. This can effectively promote the development of deep learning in the field of embedded development. Therefore, this study uses Resnet-86 as the backbone of Mask R-CNN to extract the features of the picture.

3.2. Side Fusion FPN (SF-FPN). In view of the excellent feature extraction performance of the FPN, researchers in the field of machine vision in the past two years have successively proposed models such as path aggregation network (PANet) [37], neural architecture search (NAS-FPN) [38], and bidirectional feature pyramid network (BiFPN) [39, 40] and applied them to image recognition, detection, and segmentation of various scenarios based on their research applications. We proposed the Side Fusion FPN, the main idea of which is to make full use of feature semantic information on feature fusion and feature extraction while increasing the amount of calculation as little as possible. The aim is to take full advantage of the feature map of the information with high semantics, that is, P2 to P6 mentioned above, and combine them side by side.

As shown in Figure 5, as a pioneering method for feature extraction, the FPN proposes a top-down and side-to-side connection method to combine multiscale features. Following this idea, PANet is proposed, which adds an additional path on the basis of the FPN bottom-up to the aggregating network. This method further combines feature semantic information for better feature extraction. The NAS-FPN uses a neural architecture search to obtain irregular feature network topologies. This method can cross-range fusion characteristics and adopt the neural network search technology to form a new feature pyramid structure. Although the NAS-FPN can achieve better performance, it requires thousands of GPU hours in the search process, and the generated feature network is only one thing, so it is difficult to explain. The next method to emerge was BiFPN. It uses two-way cross-scale connection and weighted feature fusion to improve the detection accuracy, but compared with the FPN, it still requires a good deal of calculation.

Again, this is why we proposed the SF-FPN algorithm. It is based on the FPN algorithm, but without increasing any output, it reduced the amount of calculation as much as possible, making full use of high semantic feature information, adding 6 fusions lines, and making the fusion between P2 and P5 and simultaneously making P6 the final fusion output. We also proposed a fully connected FPN as a comparison, that is, on the basis of PANet, all the semantic feature information will have a through-connection.

From the classic FPN structure, compared to C2–C5, we know that P2–P6 have rich feature semantic information. It is cost-effective to fuse these five feature maps. Therefore, this paper designs the Side Fusion FPN in such a way that we add only six side fusion curves on the basis of the FPN. Curve 1: transfer P5 feature semantic information to P3. Curve 2: transfer P5 feature semantic information to P2. Curve 3: transfer P4 feature semantic information to P2.

Table 3: Network structure contrast of Resnet-50/Resnet-86/Resnet-101.

Backbone	Output size	Resnet-50	Resnet-86	Resnet-101
Conv_1	512 * 512	7 * 7, 64, stride2		
Conv_2	256 * 256	3 * 3 maxpool, stride2		
Conv_3	128 * 128	$\begin{bmatrix} 1*1, 64 \\ 3*3, 64 \\ 1*1, 256 \end{bmatrix} *3$	$\begin{bmatrix} 1*1, 64 \\ 3*3, 64 \\ 1*1, 256 \end{bmatrix} *3$	$\begin{bmatrix} 1*1, 64 \\ 3*3, 64 \\ 1*1, 256 \end{bmatrix} *3$
Conv_4	64 * 64	$\begin{bmatrix} 1*1, 128 \\ 3*3, 128 \\ 1*1, 512 \end{bmatrix} *4$	$\begin{bmatrix} 1*1, 128 \\ 3*3, 128 \\ 1*1, 512 \end{bmatrix} *4$	$\begin{bmatrix} 1*1, 128 \\ 3*3, 128 \\ 1*1, 512 \end{bmatrix} *4$
Conv_5	32 * 32	$\begin{bmatrix} 1*1, 512 \\ 3*3, 512 \\ 1*1, 2048 \end{bmatrix} *3$	$\begin{bmatrix} 1*1, 512 \\ 3*3, 512 \\ 1*1, 2048 \end{bmatrix} *3$	$\begin{bmatrix} 1*1, 512 \\ 3*3, 512 \\ 1*1, 2048 \end{bmatrix} *3$
	1 * 1	Average pool, 1000-d fc, softmax		

Table 4: Experimental data.

Backbone_class	FPN + resnet101_81	FPN + resnet86_81	FPN + resnet50_81	FPN + resnet101_3	FPN + resnet86_3	FPN + resnet50_3
Epoch_steps	160_1000	160_1000	160_1000	160_1000	**160_1000**	160_1000
Total params	64158584	58549624	45088120	63744170	**58135210**	44673706
Trainable params	64047096	58453496	45028856	63632682	**58039082**	44614442
FLOPs	130205828	118834293	91542609	129377924	**118006389**	90714705
Memory_size	257.6 M	235.0 M	180.9 M	255.9 M	**233.3 M**	179.2 M
Train_time	27.98 h	21.77 h	18.25	23.02 h	**20.43 h**	18.73 h
Test_avg_time (M4,952)	2.14 s	2.014 s	1.39 s	2.10 s	**1.97 s**	1.36 s

The first row is the network structure and the number of identification categories. For example, FPN + resnet101_81 uses the Resnet-101 residual network to identify 81 types in the FPN. The second row indicates that all network structures are trained 160 * 1000 = 160,000 times. Memory_size refers to the weight of memory after each network structure is trained. Train_time refers to the time taken for each network structure training. Total params and Trainable params represent the total memory parameters and training memory parameters, respectively, of the network structure. Floating point operations (FLOPs) indicate the number of floating-point operations for each network structure, that is, the amount of calculation. Test_avg_time (M4952) refers to the average time to test 4952 images for each network structure. Min_train_loss refers to the minimum value of the weight loss after 160,000 training for each network structure.

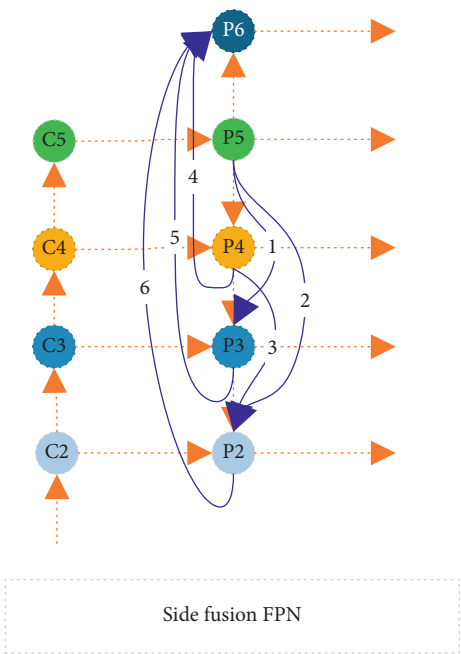


Figure 4: The Side Fusion FPN algorithm structure chart.

Curve 4: transfer P4 feature semantic information to P6. Curve 5: transfer P3 feature semantic information to P6. Curve 6: transfer P2 feature semantic information to P6. After adding the three feature fusion curves P1, P2, and P3, we can see that P2 and P3 can fully converge feature semantic information. At the same time, we have modified it in P6. In the original FPN structure, P6 is directly downsampled by P5, the semantic information only comes from P5, and some of the semantic information is lost during the downsampling process from P5 to P6. We output P6 as the last scale and aggregate the rich feature semantic information of the four scales P2, P3, P4, and P5 at the same time.

This article uses the Side Fusion FPN we proposed as the first part of Mask R-CNN and uses our design of the deep residual network Resnet-86 as the backbone to obtain five scale feature maps. As can be seen from Tables 5 and 6, with the SF-FPN algorithm we designed in the entire network framework, the amount of calculation has only increased by 2.54×10^{-7}. Although this calculation amount is almost minimal, in the test results, mAP has increased by 2.77 points, so there is an obvious improvement in accuracy.

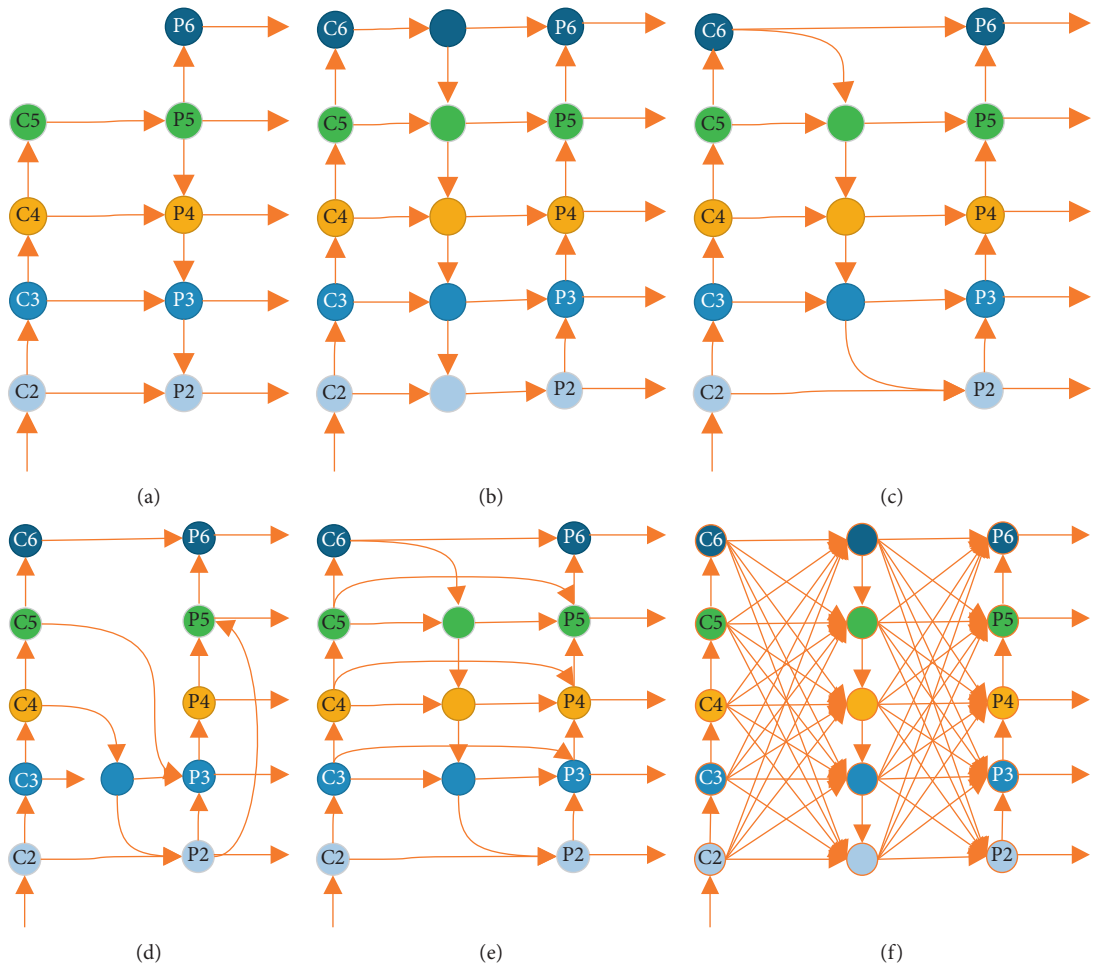


Figure 5: Six kinds of network structure diagrams all improved based on the FPN. (a) FPN. (b) PANet. (c) Simplified PANet. (d) NAS-FPN. (e) BiFPN. (f) Fully connected FPN.

Table 5: Experimental data.

Backbone_class	FPN + resnet101_81	FPN + resnet86_81	FPN + resnet50_81	FPN + resnet101_3	FPN + resnet86_3	FPN + resnet50_3
$mAP^{IOU=0.5}$	59.11	58.74	46.16	75.69	**74.84**	67.09
$mAP^{IOU=0.6}$	55.02	54.88	42.04	70.24	**69.23**	62.15
$mAP^{IOU=0.7}$	47.45	47.88	35.13	60.34	**58.77**	51.98
$mAP^{IOU=0.8}$	33.35	33.88	22.85	39.96	**39.04**	31.59
$mAP^{IOU=0.5}_{person}$	76.76	75.96	70.60	80.10	**78.89**	73.21
$mAP^{IOU=0.5}_{car}$	53.29	50.87	45.77	52.73	**51.38**	46.99
$mAP^{IOU=0.5}_{bus}$	83.15	81.26	75.86	84.09	**83.28**	76.71
Min_train_loss	0.9564	0.9092	1.287	0.6592	**0.7138**	0.8643

The evaluation standard uses the mAP. The number in the upper right corner of mAP indicates the number of input/output units (IOUs). The words "person," "car," and "bus" in the bottom right corner mean the detection of the single category. Min_train_loss refers to the loss value of each model after training.

3.3. Regional Proposal Network (RPN). The five sets of feature maps generated by the FPN are sent to the RPN. As can be seen from Figure 6, the RPN uses a small grid to slide across five sets of feature maps to produce 758664 boxes and form regional recommendations. We introduce the algorithm for generating candidate frames for RPN. As shown in Figure 6, suppose there are n feature maps in total, the width of each feature map is P_{W_i}, and the feature map is P_{H_i}, we can obtain that the feature maps have $P_{W_i} \bullet P_{H_i}$ pixels. Using each pixel as an anchor point, three scales and three scale candidate frames are generated at the same time, that is, nine candidate frames are generated for each pixel. In this way, we can obtain the number of candidate frames for each feature map as $9 \bullet (P_{W_i} \bullet P_{H_i})$. Therefore, RPN generates the total number of candidate frames after passing all the feature maps.

$$\text{TotalBoxes} = \sum_{i=1}^{n} \left[9 \bullet \left(P_{W_i} \bullet P_{H_i}\right)\right]. \tag{1}$$

Table 6: Experimental data with the SF-FPN algorithm.

Backbone_class	SF-FPN + resnet101_81	SF-FPN + resnet86_81	SF-FPN + resnet50_81	SF-FPN + resnet101_3	SF-FPN + resnet86_3	SF-FPN + resnet50_3
Epoch_steps	160_1000	160_1000	160_1000	160_1000	**160_1000**	160_1000
Total params	64158584	58549624	45088120	63744170	**58135210**	44673706
FLOPs	130205858	118834323	91542639	129377954	**118006419**	90714735
$mAP^{IOU=0.5}$	62.64	58.89	46.69	76.82	**76.64**	69.85
$mAP^{IOU=0.6}$	57.32	55.18	42.79	72.93	**71.88**	64.92
$mAP^{IOU=0.7}$	49.86	48.16	36.03	63.93	**62.41**	54.75
$mAP^{IOU=0.8}$	36.02	34.02	23.16	44.55	**42.04**	34.38
$mAP^{IOU=0.5}_{person}$	79.62	79.20	73.05	82.76	**82.46**	75.21
$mAP^{IOU=0.5}_{car}$	55.82	46.96	48.00	53.58	**52.18**	47.99

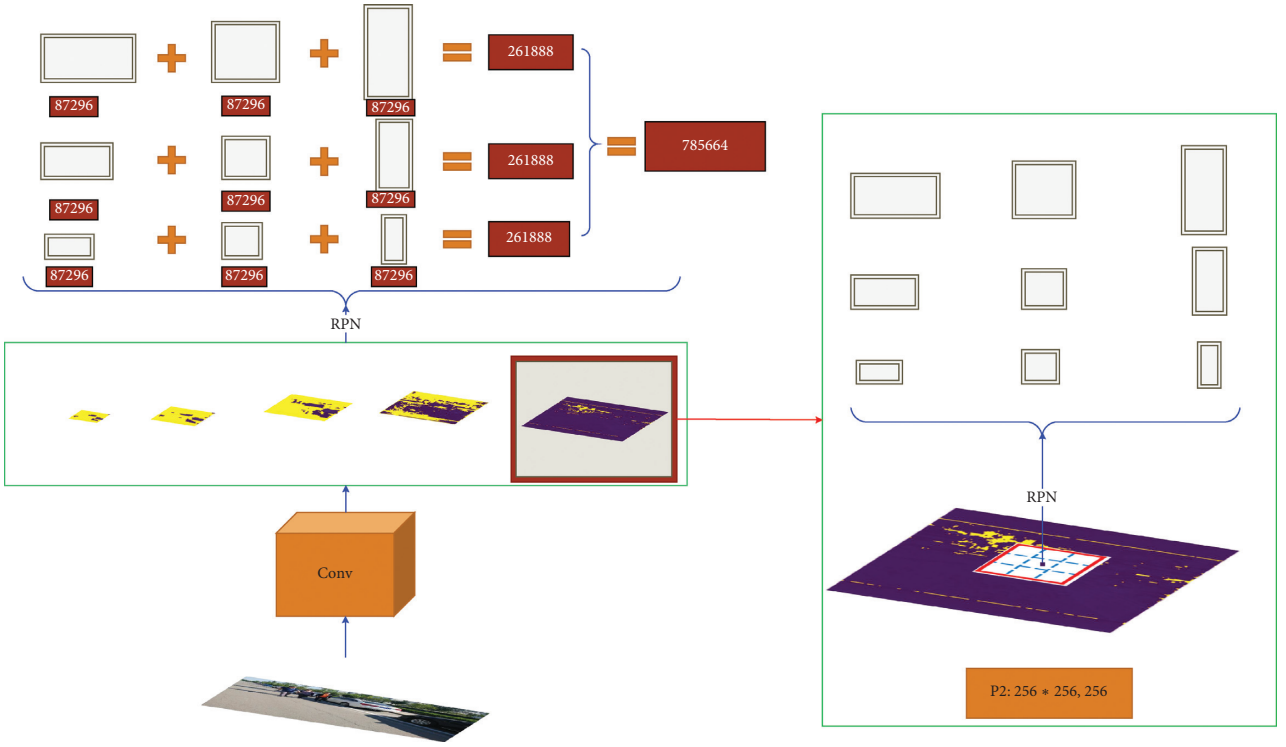


Figure 6: The RPN operation uses each pixel as an anchor point on each feature map and simultaneously generates candidate frames of three sizes and three ratios. All candidate frames are then subjected to NMS screening according to the score, and a certain number of candidate frames are selected and saved, which are used for subsequent training and prediction.

From the above, we can know that this paper generates five sets of feature maps. The size of these five feature maps is $16 * 16$, $32 * 32$, $64 * 64$, $128 * 128$, and $256 * 256$. Through the above formula, we can see that the phases of RPN can generate 758,664 boxes.

Next, the network will calculate 758,664 boxes through nonmaximum suppression (NMS) [38]. The network sorts the scores from large to small according to the four factors of color, texture, total area after merging, and the total area of the merged box in its bounding box and retains 2000 train boxes and 1000 inference boxes. Through the NMS algorithm, a large amount of calculation is required when the network selects to reserve 3000 boxes, and the network performs subsequent training and prediction on the selected 3000 boxes, with a large amount of calculation required. This is therefore the most time-consuming part of the RPN. The 3000 boxes are used to detect 81 types of targets in the COCO dataset, but there are only three categories of detection targets in this study. To increase the speed of the network without affecting the detection accuracy of the target, we need to retain 500 train boxes for training and 250 inference boxes as predictions.

3.4. Dataset Improvement. The detection objects of this paper are cars, buses, and pedestrians. The Microsoft COCO public dataset contains 81 categories, and it contains 82,081 image samples. The parameters in mask_rcnn_COCO.h5 obtained by training the dataset through Mask R-CNN are for detecting 81 kinds of targets. Using this weight directly to detect vehicles and pedestrians can make the calculations too complicated. Therefore, two changes have been made to the COCO dataset: for the first change, we screened the three

categories of images in the COCO dataset (car, bus, and person) and formed a new dataset, which we named the COCO_pcb dataset. For the second change, we labeled the new dataset with an open-source image annotation tool called VGG Image Annotator (VIA). The annotated file is named via_pcb_data.json. The dataset uses 1000 images as the training set, 100 images as the verification set, and 50 images as the test set. VIA was developed by the Visual Geometry Group, and it can be used online or offline. As shown in Figure 7, we can label the target using annotation methods for rectangles, circles, ellipses, polygons, points, and lines. This way, we not only make the sample a valid sample of the three categories of car, bus, and person but also ensure that the number of datasets is sufficient. This will assist in the precise training of the experimental part and improve the accuracy of the target detection.

4. Experiment and Results Analysis

The experimental configuration environment consisted of the Ubuntu 16.04 operating system, Intel Xeon E5 2678 V3 processor (∗2), Nvidia GTX1080TI 11G display card (∗4), Samsung 850 EVO1 TB solid state disk plus 4 TB hard disk, E5 radiator (∗2), RECC DDR4 2133 with 64 GB memory, and QuickPath Interconnect (QPI) 9.6 GT/s motherboard. The operational environment consists of the Python 3 language, TensorFlow 1.3 open-source software library, and Keras 2.0.8 neural network API modeling system.

We used the VIA image annotation tool to polygonize the original image and then sent the newly obtained annotation file via_pcb_data.json to Mask R-CNN network for 160,000 iterations. We named the weight as mask_rcnn_-via.h5. Finally, we used mask_rcnn_via.h5 as the initial weight for migration learning. We used the coco_pcb dataset, which has only three categories of car, bus, and person, to perform 160,000 iterations of training with FPN + Resnet101, FPN + Resnet50, and our designed FPN + Resnet86. Under the same environment, 1,000 iterations were performed for SF-FPN + Resnet101, SF-FPN + Resnet50, and SF-FPN + Resnet86, respectively.

This experiment was conducted with FPN + Resnet101, FPN + Resnet86, FPN + Resnet50, SF-FPN + Resnet101, SF-FPN + Resnet86, and SF-FPN + Resnet50 as backbones. We trained it on a COCO dataset containing 81 classes and a COCO_pcb dataset containing three classes. In addition, we used the SF-FPN designed by the author of this article as a backbone to train the COCO_pcb dataset containing only three classes. The epoch for the above 12 sets of experiments is set to 160. Each epoch contains 1000 iterations, and the total number of iterations is 1,920,000. During the experiment, we recorded the memory size of the weights, the total time spent on training, the average length of time on test 4952 images, and the value of mAP on test 4952 images. We also tested the params and FLOP values of 12 groups of experiments. As shown in Figures 8 and 9, we call the tensor board function to draw loss image. Finally, as shown in Figures 8 and 9, we selected the smallest loss value for each set of experiments and recorded it in Tables 4 and 5. We tested 4952 images with the best weights in each set of experiments and calculated the detection time for each network.

As can be seen from Table 4, total params is the first parameter comparison for the six network structures. Params refers to the amount of parameters included in the network structure. For example, for each layer of the n-layer convolutional neural network, the convolution the kernel width is C_{W_i}, kernel length is C_{H_i}, the number of input channels is C_{IN_i}, and the number of output channels is C_{OUT_i}. Then, we can get the memory parameters of each layer of the convolutional neural network:

$$P_{C_i} = \left(C_{W_i} \bullet C_{H_i} \bullet C_{\mathrm{IN}_i} + C_{\mathrm{OUT}_i}\right). \tag{2}$$

Usually, a complete neural network structure also includes the last fully connected layer. We assume that the input of the fully connected layer is N_{IN}, and the output is N_{OUT}. The memory parameters of this fully connected layer are

$$P_N = \left(N_{\mathrm{IN}} \bullet N_{\mathrm{OUT}} + N_{\mathrm{OUT}}\right). \tag{3}$$

Hence, we can get the number of total params:

$$\mathrm{TotalParams} = \left(\sum_{i=1}^{n} P_{C_i}\right) + P_N = \left[\sum_{i=1}^{n}\left(C_{W_i} \bullet C_{H_i} \bullet C_{\mathrm{IN}_i} + C_{\mathrm{OUT}_i}\right)\right] + \left(N_{\mathrm{IN}} \bullet N_{\mathrm{OUT}} + N_{\mathrm{OUT}}\right). \tag{4}$$

4.1. Experimental Data Analysis of FPN + Resnet-50, FPN + Resnet-86, and FPN + Resnet-101. As can be seen from Table 4, compared to Resnet101_81, the params value of Resnet86_3 is reduced by 6,023,374, and the FLOP value is decreased by 12,199,439, which saved nearly 7.55 hours of training time. When testing 4,952 images, the average detection time was reduced by 0.17 seconds, and its detection speed was increased by 7.94%. The speed of Resnet86_3 in training and detection was improved, and its weight memory was reduced by 24.3 MB, which contributes to its implementation in embedded development. As can be seen from Table 5, the minimum loss value of Resnet86_3 is 0.7138, which is 15.73 points higher than the mAP value of Resnet101_81, and it also has a certain improvement in accuracy.

Compared with Resnet86_81, Resnet86_3's params value is reduced by 414,414, and the FLOP value is reduced by 827,904, which shortens the training time by 1.35 hours. When testing 4,952 images, the average detection time is shortened by 0.04 seconds. In terms of loss, it is 0.1954 points lower than the loss value of Resnet86_81, and the mAP value is 16.4 points higher. By comparison, the two structures of Resnet-86_3 and Resnet-86_81 reflect the impact of the number of detection categories in the network.

Compared with Resnet-50_81, although Resnet-86_3 does not display an advantage in params, FLOPs, and speed, we can see from Figure 8 that it is not suited for our purpose. This is because the training effect of Resnet50_81 shows that the smallest loss is high (1.287) and mAP (28.68) is lower than the one in Resnet86_3. This does not meet our standard for detection accuracy. We can also see in Figure 8 that the

Figure 7: Comparison of the rectangular and polygonal marking method: the upper line is the rectangular dimension and the lower line is the polygon dimension.

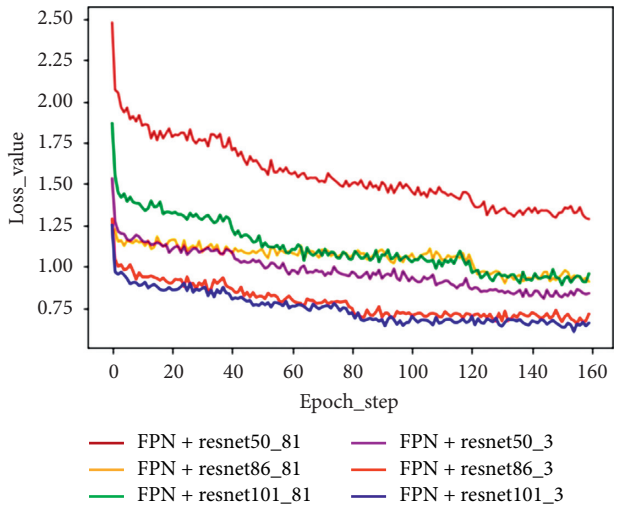


Figure 8: Six experimental loss function graphs with the FPN algorithm.

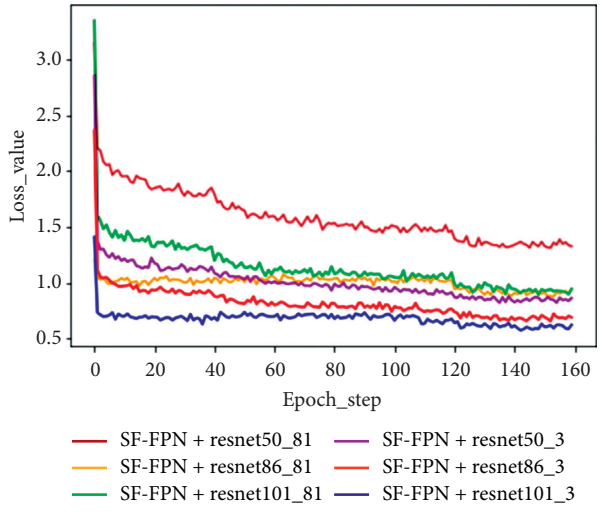


Figure 9: Six experimental loss function graphs with the SF-FPN algorithm.

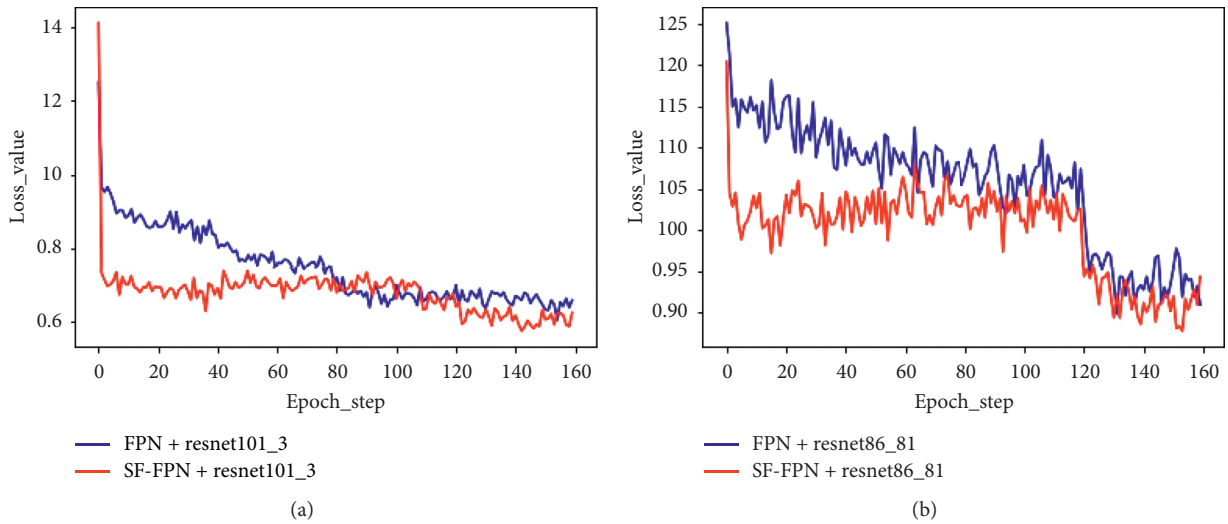


Figure 10: For two algorithms of FPN and SF-FPN, two groups of loss function curves are selected for comparison. (a) Loss function curve of FPN + resnet101_3 and SF-FPN + resnet101_3. (b) Loss function curve of FPN + resnet86_81 and SF-FPN + resnet86_81.

Figure 11: Comparison between the original algorithm and the experimental test result of the algorithm proposed in this study. (a) Original image. (b) Effect map after processing by original algorithm. (c) Effect map after processing by our algorithm.

smallest loss of Resnet50_3 is very high (0.8643) and mAP (7.75) is lower than that of Resnet86_3.

Compared with Resnet101_3, Resnet86_3's params are reduced by 5,608,960, and FLOPs are reduced by 11,371,535, which saves 2.58 hours in training time. When testing 2,895 images, the average test time per image was reduced by 0.13 seconds, and the speed of the test increased by 6.19%. The weight memory of Resnet86_3 was also 22.6 M smaller than the one of Resnet101_3, and the value of mAP was only 0.85 lower than that of Resnet101_3.

It can be seen from Figure 8 that when the Resnet-86 is designed as a backbone in the Mask R-CNN to detect the three major categories of car, bus, and person, its training effect is significantly better than the original Mask with Resnet-101 as the backbone R-CNN. Although the mAP value of Resnet-86_3 is slightly lower than the mAP value of Resnet-101_3, the detection speed of the former one is faster, which meets the goal of our design. Therefore, in the end, we used Resnet-86_3 as the backbone of Mask R-CNN and applied it to the target detection algorithm.

(a) (b) (c) (d)

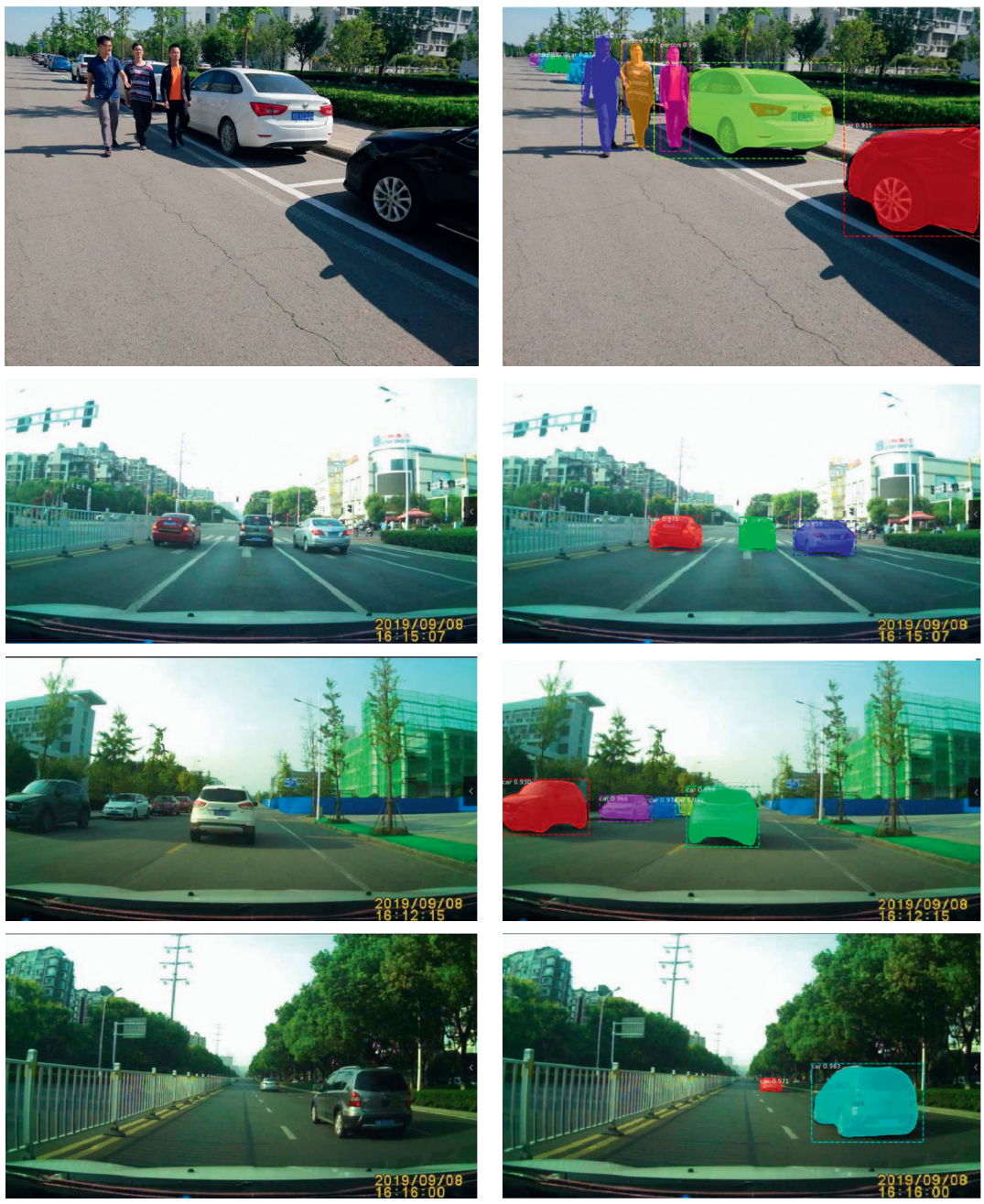
Figure 12: Comparison with state-of-the-art vehicle and pedestrian detection methods. (a) SSD. (b) YOLOv3. (c) Faster R-CNN. (d) Improved Mask R-CNN.

4.2. Analysis of FPN and SF-FPN Experimental Data. By comparing the experimental data of 12 groups in Table 4 and 6, we can see that the params value of the SF-FPN network structure designed in this paper remains unchanged without adding any memory parameters compared with the FPN. As for the SF-FPN, the integration of six side connections has been added, so the six network structures designed based on the SF-FPN algorithm have increased by 30 compared with the original structure FLOPs. Compared with the FPN + resnet101_81 network structure, the total increase is only 2.54×10^{-7}. By testing 4952 images, we can also see that the speed of the test is basically the same as the original speed. However, the mAP value has increased by 2.00. The SF-FPN algorithm, combined with resnet-101 to test 81 classes, leads to a mAP increase of 2.73 points. The SF-FPN algorithm, combined with resnet86 to test 81 classes, leads to a mAP increase of 2.18 points. The SF-FPN algorithm, combined with resnet50 to test 81 classes, leads to a mAP increase of 0.62 points. The SF-FPN algorithm, combined with resnet101 to test three categories, leads to a mAP increase of 2.93 points. The SF-FPN algorithm, combined with resnet-86 to test three categories, leads to a mAP increase of 2.77 points. The SF-FPN algorithm, combined with resnet-50 to test three categories, leads to a mAP increase of 0.75 points.

A comparison of the loss curves for the two groups of experiments selected from Figures 8 and 9 is shown in Figure 10. When the parameters such as the dataset, the number of network layers, the number of detection categories, the training rate, and the number of training iterations during training are the same, we find that the effect is obviously better than the FPN, and the loss value of the SF-FPN's loss curve has always been lower than the FPN's loss value. Particularly in the early stage, it can quickly reach a lower loss value, which has basically been ahead of the FPN training effect.

Taking into account the quality characteristics of the SF-FPN, we finally adopted the SF-FPN as the feature extraction structure, in which the SF-FPN uses Resnet86 as the network structure for object recognition, detection, and classification of the three categories of car, bus, and person. Compared with the original FPN + Resnet101 structure recognition class 81, we designed the network structure SF-FPN + Resnet86_3 to improve the training speed by 26.98% and improve the mAP accuracy evaluation by 17.53 points. As shown in Figure 11, we tested two images with the above two algorithms, respectively, in which the FPN + resnet101_81 algorithm missed the small target vehicle in the red frame in the first image, while the SF-FPN + resnet86_3 algorithm proposed in this paper accurately detected the small target vehicle in the red frame. When detecting the second image, the FPN + resnet101_81 algorithm does not segment the red table frame accurately in the segmentation process, while our algorithm SF-FPN + resnet86_3 segments the part accurately, distinguishing between the vehicle and nonvehicle parts. At the same time, the network structure framework we designed can be easily migrated to other network structure models, such as Faster R-CNN, SSD, and YOLOv3.

At the end of this section, we would like to provide a short comparative remark on our method and some state-of-art methods. After testing, the mAP value of SSD-based vehicle detection algorithm [41] is only 50.4%, which is 26.24% lower than our new algorithm. As shown in Figure 12(a), in the detection of the first test diagram, the target at the lower right corner is not detected, and there is a missed detection. The mAP value of vehicle detection algorithm [42] based on YOLOv3 is 57.9%, which is 18.74% lower than the new algorithm. It can be seen from Figure 12(b) that the detection and positioning of the leftmost target "car" in the second image is not accurate enough. The mAP value of Faster R-CNN vehicle detection algorithm [43] is 59.1%, but it is still 17.54% lower than the new algorithm. It can be seen from Figure 12(c) that there is a missed inspection in the vehicle detection of both figures. At the same time, in conjunction with the *cascaded superpixel pedestrian object segmentation*



(a) (b)

Figure 13: The four images on the left are the original images, and the four images on the right are the experimental results. The four results are obtained by the network algorithm designed in this paper, which improves on the FPN, RPN, and categories. (a) Road original image. (b) Mask R-CNN test chart.

algorithm [44], *bibox regression for pedestrian detection algorithm* [45] and other algorithms are compared, and the experimental results show that the vehicle and pedestrian detection based on the improved Mask R-CNN is slightly more accurate in the task of case segmentation.

5. Conclusions

The main research content of this study is about how to make the Mask-RCNN algorithm detect and segment cars, buses, and persons on the road more accurately and more quickly in the anticollision warning system. To increase the accuracy of the training effect of the network, we filtered and supplemented the dataset. To meet the real-time requirements of smart driving, we designed the Resnet-86 network and used it as a network backbone. To further increase the detection speed of the network, we modified the number of reserved RPN candidate frames. For greater accuracy, we designed the SF-FPN algorithm for feature extraction. Through improving the dataset, FPN, and RPN, our network

improved the detection speed by 7.94%, and the detection accuracy increased the value of mAP by 17.53 points over the original Mask R-CNN network.

Based on Mask R-CNN, we improved the network to integrate the functions of image recognition, detection, and segmentation. As we can see from Figure 13, the improved network can accurately detect a distance of about 200 m, even though the target is occluded by 95%. It can be seen that the network can be applied to the intelligent driving anticollision warning system to identify the car, bus, and person ahead.

Although the recognition speed of this network has reached 5 FPS, for some real-time system applications, the recognition speed still needs to be increased and the hardware configuration requirements need to be reduced. For example, in the vehicle tracking task, the target detection speed needs to be completed faster; in the precision instrument segmentation task, the target needs to be segmented more accurately; in the vehicle emergency braking device, the target needs to be detected faster to complete emergency braking.

In the future, in order to improve the detection accuracy, we can further improve and design the feature extraction algorithm to make the feature semantic information more abundant. It can also further optimize the depth residual network, reduce the loss function value, and improve the network training effect. In order to improve the detection speed, we can combine and optimize the deep convolution neural network algorithm, reduce the network computing redundancy, and improve the network detection speed. It can also be combined with hardware configuration to enhance the network computing ability and further improve the speed of target detection. Therefore, how to improve the speed and accuracy of target detection is still our key research work in the future.

Data Availability

The data used to support the findings of this study are currently under embargo while the research findings are commercialized. Requests for data, 6 months after publication of this article, will be considered by the corresponding author.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was supported in part by the National Natural Science Foundation of China (no. 11971034).

References

[1] J. Nie, "Design of visual feature detection system for intelligent driving of electric vehicle," in *Proceedings of the International Conference on Robots & Intelligent System (ICRIS)*, ACM, Amsterdam, Netherlands, February 2018.

[2] E. Sun, A. Nieto, and Z. Li, "GPS and Google Earth based 3D assisted driving system for trucks in surface mines," *International Journal of Mining Science and Technology*, vol. 20, pp. 138–142, 2016.

[3] Y.-S. Chen, S.-C. Chiu, and S.-S. Hsiau, "Safe technology with a novel rear collision avoidance system of vehicles," *International Journal of Automotive Technology*, vol. 20, no. 4, pp. 693–699, 2019.

[4] H. Kim, J. Kim, and Y. Kim, "Design of network threat detection and classification based on machine learning on cloud computing," *Cluster Computing*, vol. 22, pp. 1–10, 2019.

[5] H. Xing, G. Zhang, and M. Shang, "Deep learning," *International Journal Of Semantic Computing*, vol. 10, pp. 417–439, 2016.

[6] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *In*, IEEE, in *Proceedings of the IEEE Computer Society Conference On Computer Vision And Pattern Recognition*, June 2005.

[7] M. M. Adankon and M. Cheriet, "Support vector machine," *Computer Science.*vol. 1, pp. 1–28, 2002.

[8] N. Faris, S. Ali, and M. Z. A. Ilham, "Image-based feature extraction technique for inclined crack quantification using pulsed eddy current," *Chinese Journal of Mechanical Engineering*, vol. 32, pp. 19–34, 2019.

[9] K. Phil, *MATLAB Deep Learning*, pp. 103–121, Berkeley, New York, NY, USA, 2017.

[10] Y. L. Cun, B. Boser, J. S. Denker, D. Henderson, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," *Advances in Neural Information Processing Systems*, vol. 2, pp. 396–404, 1997.

[11] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proceedings of the International Conference On Neural Information Processing Systems*, IEEE, Lake Tahoe, NV, USA, December 2012.

[12] R. Girshick, J. Donahue, T. Darrel, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the CVPR*, IEEE, Columbus, Ohio, USA, June 2014.

[13] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE International Conference On Computer Vision*, IEEE, Santiago, Chile, December 2015.

[14] S. Ren, K. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[15] K. M. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE International Conference On Computer Vision*, IEEE, Venice, Italy, October 2017.

[16] T. Y. Lin, M. Maire, S. Belongie et al., "Microsoft COCO: common objects in context," in *Proceedings of the European Conference On Computer Vision*, Springer, Zurich, Switzerland, September 2014.

[17] V. Bianco, P. L. Mazzeo, M. Paturzo, C. Distante, and P. Ferraro, "Deep learning assisted portable IR active imaging sensor spots and identifies live humans through fire," *Optics and Lasers in Engineering*, vol. 124, 2020.

[18] P. L. Mazzeo, A. Argentieri, F. D. Luca et al., "Convolutional neural networks for recognition and segmentation of aluminum profiles," *Multimodal Sensing: Technologies and Applications*, vol. 11059, 2019.

[19] Q. C. Pan and H. H Zhang, "Key algorithms of video target detection and recognition in intelligent transportation systems," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 4, pp. 142–162, 2019.

[20] D. W. Du, Y. K. Qi, and H. Y. Yu, "The unmanned aerial vehicle benchmark: object detection and tracking," in

Proceedings of the European Conference On Computer Vision, Springer, Munich, Germany, September 2018.

[21] W. Ge, S. Yang, and Y. Yu, "Multi-evidence filtering and fusion for multi-label classification, object detection and semantic segmentation based on weakly supervised learning," in *Proceedings of the CVPR*, IEEE, Salt lake city, UT, USA, August 2018.

[22] S. Xi, R. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the CVPR*, IEEE, Honolulu, HI, USA, July 2017.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference On Computer Vision & Pattern Recognition*, IEEE, Las Vegas, NV, USA, June 2016.

[24] T. Y. Lin, P. Dollár, R. Girshick, K. He, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the CVPR*, IEEE, Honolulu, HI, USA, July 2017.

[25] A. K. Fattal, M. Karg, C. Scharfenberger, and J. Adamy, "Saliency-guided region proposal network for cnn based object detection," in *Proceedings of the IEEE International Conference On Intelligent Transportation Systems*, IEEE, Yokohama, Japan, October 2017.

[26] Z.-b. Zhao, L. Zhang, Y.-c. Qi, and Y.-y. Shi, "A generation method of insulator region proposals based on edge boxes," *Optoelectronics Letters*, vol. 13, no. 6, pp. 466–470, 2017.

[27] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013.

[28] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 39, pp. 640–665, 2014.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *Proceedings of the European Conference On Computer Vision*, Springer, Zurich, Switzerland, September 2014.

[30] A. Bulat, J. Yang, and G. Tzimiropoulos, "To learn image super-resolution, use a GAN to learn how to do image degradation first," in *Proceedings of the European Conference On Computer Vision*, Springer, Munich, Germany, September 2018.

[31] B. H. Chen, Y. G. Wang, and G. Wei, "End-to-End trained sparse coding network with spatial pyramid pooling for image classification," *Neural Processing Letters*, vol. 1, pp. 1–16, Jan. 2019.

[32] H. Zhang, V. Sindagi, and V. M. Patel, "Multi-scale single image dehazing using perceptual pyramid deep Network," in *Proceedings of the CVPR*, IEEE, Salt lake city, UT, USA, August 2018.

[33] M. E. Paoletti, J. M. Haut, R. Fernandez-Beltran, J. Plaza, A. J. Plaza, and F. Pla, "Deep pyramidal residual networks for spectral-spatial hyperspectral image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 2, pp. 740–754, 2019.

[34] R. Gail, M. Paular, R. Gary, E. Mary, and K. S. Heidi, "Top-down and bottom-up treatment approaches compared in a single case of pure alexia," *Aphasiology*, vol. 1, pp. 1–14, 2019.

[35] J. Chen, Z. Cai, J. Lai, and X. H. Xie, "Efficient segmentation-based patchmatch for large displacement optical flow estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 99, p. 1, 2018.

[36] F. Dornaika, K. Hammoudi, M. Melkemi, and T. D. A. Phan, "An efficient pyramid multi-level image descriptor: application to image-based parking lot monitoring," *Signal Image and Video Processing*, vol. 11, pp. 1–7, 2019.

[37] S. Liu, L. Qi, H. F. Qin, "J. Shi, and J. Jia., "Path aggregation network for instance segmentation," in *Proceedings of the CVPR*, IEEE, Salt lake city, UT, USA, August 2018.

[38] G. Ghiasi, T. Lin, R. Pang, and V. Le, "Nas-fpn: learning scalable feature pyramid architecture for object detection," in *Proceedings of the Conference On Computer Vision And Pattern Recognition (CVPR)*, IEEE, Los Angeles, CL, USA, June 2019.

[39] S. Woo, S. Hwang, H.-D. Jang, and I. S. Kweon, "Gated bi-directional feature pyramid network for accurate one-shot detection," *Machine Vision and Applications*, vol. 30, no. 4, pp. 543–555, 2019.

[40] J. Hosang, R. Benenson, and B. Schiele, "A convnet for non-maximum suppression," in *Proceedings of the German Conference On Pattern Recognition*, Springer International Publishing, Stuttgart, Germany, September 2016.

[41] X. Le, L. Xiang, and Z. Hua, "Vehicle Detection on Road Frontage Based on SSD," *Software Guide*, vol. 5, pp. 27–36, 2019.

[42] F. K. Zhang, Y. Feng, and L. I. Ce, "Fast vehicle detection method based on improved YOLOv3," *Computer Engineering and Applications*, vol. 3, pp. 133–139, 2019.

[43] J. Sang, P. Guo, and Z. Xiang, "Vehicle detection based on faster-RCNN," *Chongqing Daxue Xuebao/Journal of Chongqing University*, vol. 40, pp. 32–36, june. 2017.

[44] D. Yang, J. Huang, J. Zhang, and R. Zhang, "Cascaded superpixel pedestrian object segmentation algorithm," *Control and decision making in China (CCDC)*, vol. 5, pp. 41–48, 2018.

[45] C. Zhou and J. Yuan, "Bi-box regression for pedestrian detection and occlusion estimation," in *Proceedings of the European Conference On Computer Vision*, Springer, Munich, Germany, September 2018.