Analysis of Object Detection Performance Based on Faster R-CNN

Analysis of Object Detection Performance Based on Faster R-CNN

Wenze Li[1*]

[1]Faculty of Engineering and Information Technology, University of Technology Sydney, Sydney, Australia

*Wenze.Li-1@student.uts.edu.au

Abstract. The related regions with convolutional neural networks (R-CNN) models have been widely used in the field of object detection. Faster R-CNN significantly improves the overall performance by adding RPN, especially in terms of detection speed. However, the application of different pre-training models will result in a great difference in the performance of Faster R-CNN. This paper analyzed the performance of Faster R-CNN models based on different pre-training models and conducted a comprehensive evaluation of the performance of Faster R-CNN. The experimental results showed the accuracy and detection speed of R-CNN, fast R-CNN and faster R-CNN based on three different data sets. They can objectively and comprehensively evaluate the performance of R-CNN, fast R-CNN, and faster R-CNN.

1. Introduction

Object detection is a hot topic of computer vision. The main purpose of object detection is to find objects of interest in images or videos and detect their position and size simultaneously. Object detection is an image segmentation based on the geometric and statistical characteristics of the object, and it combines the segmentation and recognition of the object. The accuracy and real-time performance are important capabilities of the entire system for object detection. Especially in complex scenes, automatic object extraction and recognition are particularly important when multiple objects need to be processed in real-time. In recent years, object detection has been widely used in artificial intelligence, face recognition, unmanned driving, and other fields. The existing object detection algorithms include traditional detection algorithms and detection algorithms based on deep learning. Traditional object detection algorithms are mainly based on sliding window frames or matching based on feature points. Although this method has achieved good results, the lack of pertinence when using sliding windows for region selection leads to high time complexity and window redundancy. In addition, the methods based on manual feature selection are often not very robust. With the development of deep learning technology, object detection algorithms have switched from traditional methods based on manually selected features to detection methods based on deep neural networks. Detection methods based on deep neural networks can be mainly divided into two categories: one is a two-stage object detection algorithm combining region proposal and convolutional neural networks (CNN), such as R-CNN, and the other one is a one-stage algorithm that converts object detection into a regression problem (for example, YOLO).

R-CNN algorithm was first proposed by Girshick et al. in 2014, who applied a large-capacity CNN to bottom-up region proposals [1]. The R-CNN affected all subsequent two-stage algorithms and made CNN-based object detection algorithms gradually become the mainstream. The application of deep learning has improved detection accuracy and detection speed. In 2015, Girshick proposed Fast R-CNN

based on R-CNN, which simplified the spatial pyramid pooling (SPP) layer to the region of interest (ROI) Pooling layer on the basis of SPP-Net, and decomposed the output of the fully connected layer into (SVD) to obtain two output vectors: the classification score of softmax and the window regression of the bounding rectangle of the Boundingbox. This method improvement combines the classification problem with the bounding box regression problem. Fast R-CNN uses softmax instead of singular value decomposition (SVM) and stores all the features in the video memory, which reduces the storage space occupation and greatly accelerates the detection speed [2]. However, neither R-CNN nor Fast R-CNN can solve the following problem: using selective search and similar methods to select region suggestions will generate a large number of invalid regions. It leads to inefficiency and waste of computing power. Ren et al. proposed to use Region Proposal Networks (RPN) to replace the Selective Search algorithm on the basis of Fast R-CNN. RPN can use neural networks to learn its own strategy for generating region proposals and make full use of feature maps. RPN replaces time-consuming selective search and other similar algorithms to make detection faster [3].

The main contributions of this work can be summarized as follows:

1. Faster R-CNN algorithm is implemented based on Pytorch, and VGG16 and ResNet101 are used as pre-training models to train and record the time on the two data sets of Pascal VOC and COCO, respectively.

2. The trained model is tested to get the accuracy rate (mAP) and detection time.

3. The performance of Faster R-CNN is analyzed under different pre-training models and data sets.

The rest of this paper is organized as follows. In Section 2, the network structure of the Faster R-CNN algorithm will be introduced in detail. In Section 3, faster R-CNN test results based on different pre-training models and data sets will be displayed, and this part will analyze the performance of faster R-CNN. Finally, Section 4 concludes the performance analysis results of Faster R-CNN under different data sets and pre-training models.

2. Related works

According to the description of Ren et al. in the original paper, the entire process of Faster R-CNN can be divided into four parts: Conv layers, Region Proposal Networks, Roi Pooling and Classification [3]. First, a $P \times Q$ image of any size is scaled to a fixed size $M \times N$ and sent to Conv layers to extract the feature map. This feature map is shared for subsequent RPN layer and fully connected layer. The feature maps then input to the Region Proposal Network to generate Region of Interest (ROIs). RPN uses softmax to determine whether the anchors are positive or negative, and then corrects the anchors to obtain an accurate proposal through the bounding box regression. The Roi Pooling layer gathers the feature maps and proposals from the input and integrates that information for extracts the proposal feature maps, and then sends them to the fully connected layer to determine the object category. Finally, the Classification part calculates the category of the proposal by proposal feature maps, and it also obtains the final precise position of the detection frame by bounding box regression at the same time. Figure 1 shows the flow chart of Faster R-CNN.

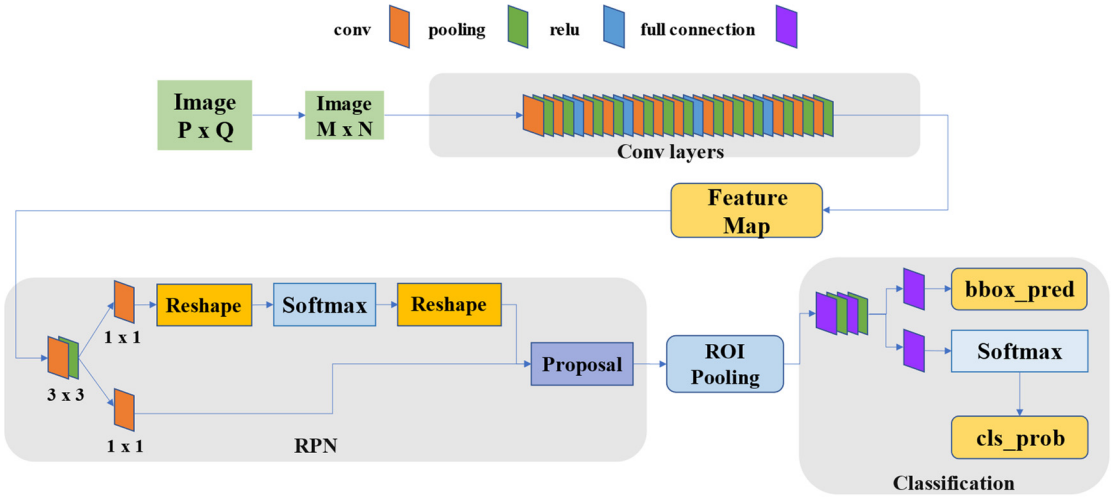

Figure 1. The framework of Faster R-CNN

2.1 Conv layers

The Conv layers contains 13 conv layers, 13 relu layers and 4 pooling layers.

All conv layers are: kernel_size=3, pad=1, stride=1

All pooling layers are: kernel_size=2, pad=0, stride=2

Since all convolutions in Faster R-CNN Conv layers are expanded (pad=1), the original image will fit to $(M+2)\times(N+2)$ size, and then 3×3 convolution to output $M\times N$. This method can ensure that the conv layer in Conv layers does not change the input and output matrix sizes (as shown in Figure 2). Similarly, each $M\times N$ matrix will become to $(M/2)\times(N/2)$ in size when passing through the pooling layer. In the entire Conv layer, the input and output sizes will not be changed by the conv and relu layers, and only the output length and width will be changed to 1/2 of the input by the pool layer.

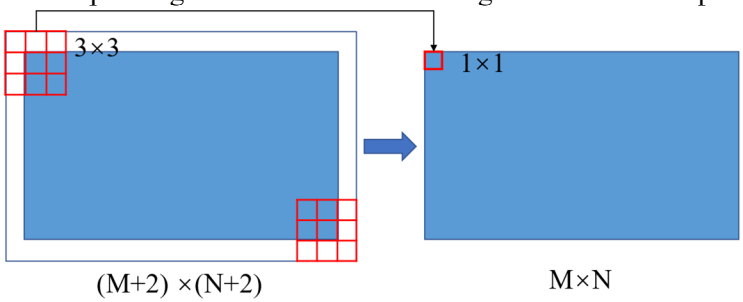

Figure 2. The convolution diagram

2.2 Region Proposal Networks (RPN)

Traditional detection methods generate detection frames are very time-consuming. For example, R-CNN generates detection frames through the selective search (SS) method. Faster R-CNN uses RPN to generate detection frames instead of the traditional sliding window and SS method. This method can greatly increase the speed of detection frame generation, which is a big advantage of R-CNN. RPN is divided into two parts (as shown in Figure 3). One part is to obtain the anchors positive and negative classification through Softmax, and the other part is used to obtain an accurate solution by calculating the anchor point bounding box regression. Finally, the proposal layer synthesizes the positive anchor point and the corresponding bounding box regression offset to obtain recommendations, eliminating the proposals that are too small and out of the boundary at the same time.

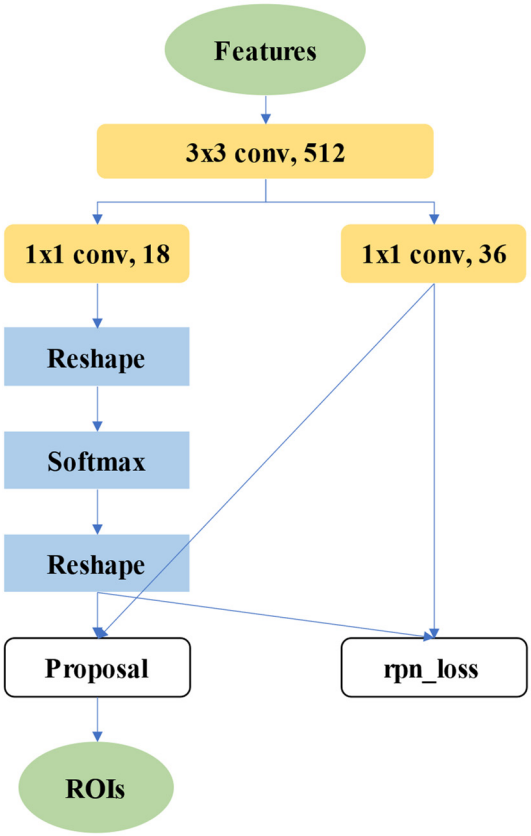

Figure 3. The network structure diagram of RPN

2.2.1 Anchors

At each sliding window position, multiple candidate boxes are predicted where the maximum number of candidate boxes at each position is represented by k. Therefore, the regression layer has 4k output codes and k candidate boxes, and the classification layer has 2k output scores to estimate the probability that each candidate box is a background class. The k candidate frames are parameterized, and they are called anchors. An anchor is essentially the center of the sliding window and is related to the scale and aspect ratio. By default, 3 scales and 3 aspect ratios are used to generate 9 anchors at each position. For a convolution feature map of size $W \times H$, there are a total of $W \times H \times k$ anchors.

2.2.2 Softmax determines the anchors

After using Backbone to extract the feature map, Conv1x1 convolution is used to obtain a $H \times W \times 18$ matrix to indicate whether the 9 anchors at each pixel are positive or negative. Then Softmax is used to determine whether the object is included; it is generally considered that a positive simple includes the object. The reshape layer before Softmax is used to reshape the input $B \times H \times W \times 18$ into $B \times 2 \times 9 \times H \times W$, and the second dimension is used to store the Softmax classification results of 9 anchors at each pixel. The reshape layer is used to adjust it back to $B \times 18 \times H \times W$.

2.2.3 Bounding box regression

Bounding box regression is to fine-tune the position of the detection box because an Anchor cannot contain exactly one object, so it needs to be adjusted. Suppose the window uses a 4-dimensional vector $(x,\ y,\ h,\ w)$ to represent the center point coordinates, height and width of the window. Given anchor $A = (A_x, A_y, A_h, A_w)$ and GroundTruth $G = (G_x, G_y, G_h, G_w)$, the task of the detection frame

regression is to find a transformation $G' = F(A)$ such that $G' \approx G$.

This transformed network is proposed in R-CNN. The transformation F includes four functions $F = \{d_x(A), d_y(A), d_h(A), d_w(A)\}$, and it is used to translate the center point and scale the width and height:

$$G_x^{'} = A_w d_x(A) + A_x \quad (1)$$

$$G_y^{'} = A_h d_y(A) + A_y \quad (2)$$

$$G_w^{'} = A_w \exp(d_w(A_w)) \quad (3)$$

$$G_h^{'} = A_h \exp(d_h(A_h)) \quad (4)$$

The input of this regression function is the feature map and the offset between GroundTruth and the anchor, and the output is $d*(A), * \in \{x, y, w, h\}$. Assuming that the input feature map is $\phi(A)$, then:

$$d_*(A) = W_*^T \phi(A) \quad (5)$$

The loss function is:

$$Loss = \sum_i^N | t_*^i - W_*^T \phi(A) | \quad (6)$$

Among them, t* is the offset between GroundTruth and anchor:

$$t_x = \frac{G_x - A_x}{A_w}, \quad t_y = \frac{G_y - A_y}{A_h} \quad (7)$$

$$t_w = \log(\frac{G_w}{A_w}), \quad t_h = \log(\frac{G_h}{A_h}) \quad (8)$$

The function optimization goals are:

$$\hat{W}_* = \arg\min_{w_*} \sum_i^N \left| t_*^i - W_*^T \phi(A) \right| + \lambda \left| W_* \right| \quad (9)$$

2.2.4 Proposal Layer

The function of the Proposal layer is to calculate an accurate proposal based on the coordinate regression matrix of the bounding box of size $H \times W \times 4k$ and it is sent to the subsequent RoI Pooling Layer.

The specific steps to implement the Proposal layer are:

1. Generate proposals based on RPN predicted offset and positive bounding box;
2. Process the bbox that exceeds the image boundary, and treat the image boundary as the boundary of the bbox that exceeds the boundary;
3. Remove bboxes less than the threshold;
4. Sort all (proposal, score) pairs from high to low according to the value of the score;
5. Take the first pre_nms_topN (6000) values;
6. Perform NMS on the remaining positive proposals, threshold = 0.7;
7. Take the top post_nms_topN (300) after NMS;
8. Return the remaining proposal.

2.3 RoI pooling

RoI Pooling is to change each generated proposal to the size of pooledw×pooledh through pooling. The specific steps are:

1. Since the proposal coordinates output by the Proposal layer are on the left side of the original image, spatial_scale is firstly used to transform the proposal onto the feature map;
2. Divide the proposals on each feature map into a grid of pooledw×pooled;
3. Max pooling is performed on each part of the grid.

2.4 Classification

The Classification part calculates the categories that each proposal belongs to (such as desk, cars, people, etc.) by using the proposal feature maps, full connect layer and softmax, and outputs the cls_prob probability vector. Moreover, Faster R-CNN will use bounding box regression again to obtain the position offset bbox_pred of each proposal at the same time, which is used to return to a more accurate target detection frame.

3. Experiment

3.1 Datasets

The data sets used in this experiment are the three most common public data sets in the field of object detection, which are PASCAL VOC2007, COCO and ILSVRC.

3.1.1 PASCAL VOC2007

The PASCAL VOC2007 data set comes from the Pattern Analysis, Statistical Modelling and Computational Learning project Visual Object Classes (PASCAL VOC) challenge, which is a world-class computer vision challenge. Lots of excellent computer vision models are launched in the PASCAL VOC Challenge (such as classification, positioning, detection, segmentation, action recognition and other models) especially some target detection models, such as R-CNN series, YOLO, SSD. The PASCAL VOC2007 data set is divided into four categories: vehicle, household, animal and person, and twenty subcategories. VOC2007 contains 9963 annotated pictures and is composed of three parts train/val/test, and a total of 24,640 objects are marked.

3.1.2 MS COCO

The full name of MS COCO is Microsoft Common Objects in Context, which is the Microsoft COCO data set that Microsoft funded and annotated in 2014. The COCO dataset is large and rich object detection, segmentation and captioning dataset with a size of more than 25. The main purpose of the COCO data set is to understand the scene, and the pictures in the data set are mainly obtained from complex daily scenes. The target in the image is calibrated through precise segmentation. The data set includes 91 types of targets, 328,000 images and 2.5 million labels. The number of individuals in the entire dataset exceeds 1.5 million, and it has been the largest dataset of semantic segmentation so far.

3.1.3 ILSVRC

The IMAGENET Large-scale Visual Recognition Challenge (ILSVRC) is a data set often used in deep learning and computer vision. Most of the research work is based on this data set, such as image classification, positioning and detection. The size of the ILSVRC data set is about 148G, and it has about 15 million pictures. It covers more than 20,000 categories, more than one million pictures of which have clear category annotations and the location of objects in the images. The ILSVRC data set has detailed documents, and it is maintained by a dedicated team, so it is very convenient to use. The ILSVRC data set is widely used in research papers in the field of computer vision, and it has almost become the current "standard" data set for algorithm performance testing in the field of deep learning images.

Table 1. The data details of PASCAL VOC2007, MS COCO and LSVRC

	Train		Val		Trainval		Test	
	Images	Objects	Images	Objects	Images	Objects	Images	Objects
PASCAL VOC2007	2501	6301	2510	6307	5011	12608	4952	12032
MSCOCO	118287	-	5000	-	35504	-	40670	-
ILSVRC	1,281,167	-	50,000	-	-	-	100,000	-

All experiments are conducted on a computer with a 3.6GHz Intel i9-9900KF processor, 32GB RAM and Nvidia GTX1080Ti under a Microsoft Windows 10 operating system. The program codes of data

preprocessing and graphs modeling are written by Python 3.7.

3.2 Evaluation Metrics

3.2.1 Precision and Recall

Accuracy and recall are two metrics widely used in information retrieval and statistical classification to evaluate the quality of results. Precision and recall can well measure the performance of target detection algorithms or retrieval algorithms. For a binary classification problem, the instances are classified into positive (Positive/+) or negative (Negative/-), but there are four situations when using the classifier for classification:

If an instance is a positive class and is predicted to be a positive class, the result recorded as True Positive TP/T+;

If an instance is a positive class but is predicted to be a negative class, the result recorded as False Negative FN/F-;

If an instance is a negative class but is predicted to be a positive class, the result recorded as False Positive FP/F+;

If an instance is a negative class but is predicted to be a negative class, the result recorded as True Negative TN/F-.

Precision describes how many of the positive examples predicted by the two classifiers are true positive examples:

$$P = \frac{TP}{TP + FP} \tag{10}$$

Recall describes how the binary classifier picked out many real positive examples in the test set:

$$R = \frac{TP}{TP + FN} \tag{11}$$

3.2.2 mAP

Mean Average Precision (mAP) is the average value of Average Precision (AP), which is the main evaluation index of the target detection algorithm. The mAP was first used in PASCAL Visual Objects Classes (VOC) challenge. Speed and accuracy (mAP) are usually used to describe the performance of object detection models. The higher the mAP value is, the better the detection effect of the target detection model on a given data set can be.

$$\text{mAP} = \frac{\sum_{i=1}^{K} AP_i}{K} \tag{12}$$

3.2.3 Speed

Many practical applications of target detection technology have high requirements on accuracy and speed. If the speed performance index is not considered, only the breakthrough of accuracy performance is emphasized, but the cost is higher computational complexity and more memory requirements. Generally speaking, the speed evaluation indicators in target detection are:

(1) FPS, the number of pictures that the detector can process per second

(2) The time required for the detector to process each picture

The speed evaluation index must be carried out on the same hardware, ensuring that its maximum floating-point number of operations per second represents the hardware performance (FLOPS) is the same. Different networks require different floating-point operands (FLOPs) to process each picture, so the smaller the FLOPs required to process the same picture on the same hardware, the more pictures can be processed at the same time.

3.3 Experimental Results

Figure 4 shows the Precision-Recall curve of R-CNN, Fast R-CNN, and Faster R-CNN. It can be seen that Faster R-CNN using RPN network has a higher recall rate than R-CNN and Fast R-CNN using Selective Search. Table 2 shows the accuracy of R-CNN, Fast R-CNN and Faster R-CNN on three different data sets and expressed in mAP. The accuracy of Faster R-CNN is much higher than that of R-CNN and Fast R-CNN. For example, the mAP of Faster R-CNN is 8.2% higher than Fast R-CNN and 20.9% higher than R-CNN on the PASCAL VOC2007 data set. Faster R-CNN not only improves accuracy, but also the detection speed. It can be seen from the data in Table 3 that the test time per image of Faster R-CNN is 10 times than that of Fast R-CNN and 235 times than that of R-CNN.

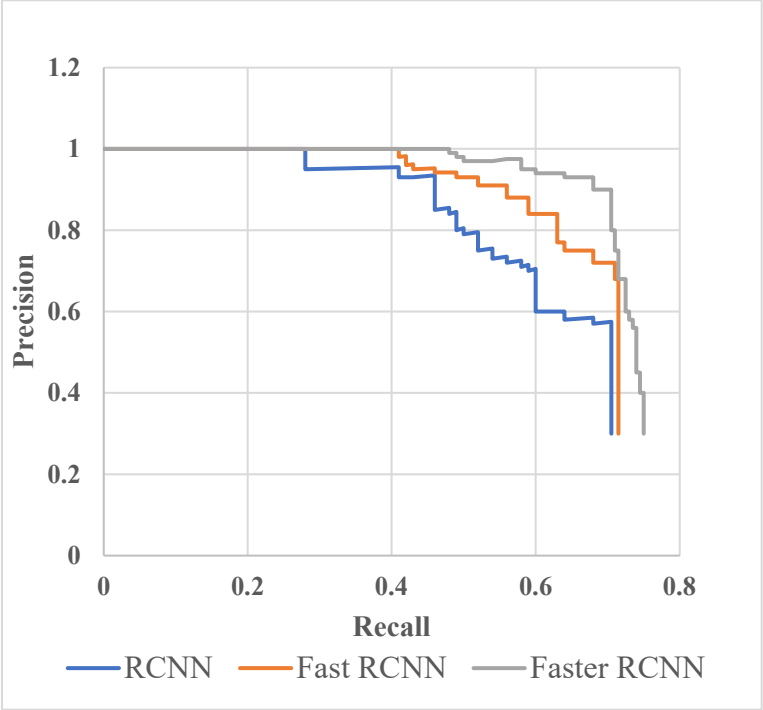

Figure 4. The Precision-Recall curve of R-CNN, Fast R-CNN and Faster R-CNN

Table 2. The mAP(%) of R-CNN, Fast R-CNN and Faster R-CNN on three datasets

Datasets Metrics	PASCAL VOC 2007	COCO	ILSVRC
R-CNN	54.2	24.6	31.4
Fast R-CNN	66.9	35.9	24.9
Faster R-CNN	75.1	42.5	46.9

Table 3 The test time per image of R-CNN, Fast R-CNN and Faster R-CNN

	R-CNN	Fast R-CNN	Faster R-CNN
Test time per image	47 seconds	2 seconds	0.2 seconds
Speed up	1×	23.5×	235×

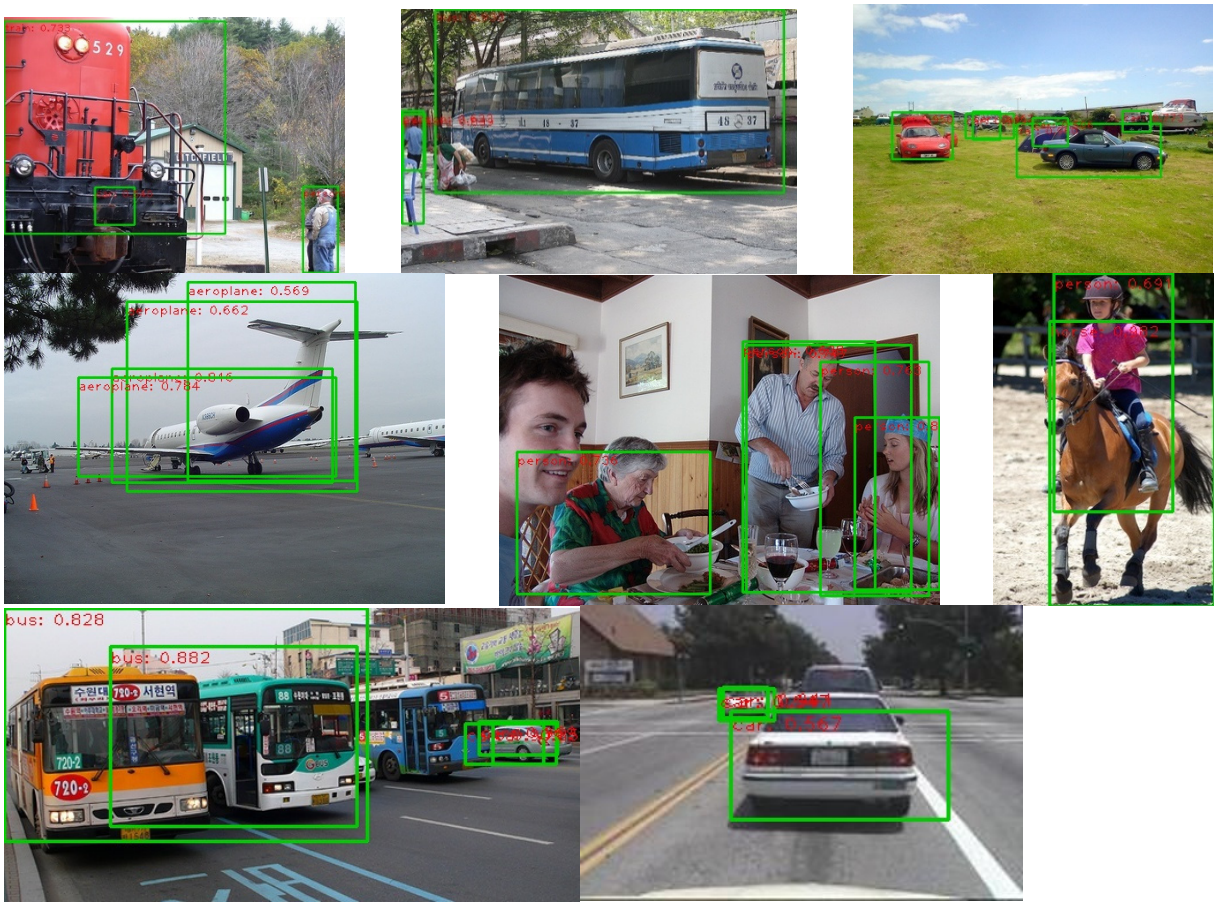

Figure 5. The detection results of Faster R-CNN

4. Conclusion

This article reviews the development process of the target detection algorithm and the R-CNN series of algorithms and analyzes the algorithm structure of Faster R-CNN in detail. According to the experimental results, the performance of Faster R-CNN is analyzed, which includes the accuracy and speed of Faster R-CNN when dealing with target detection problems. The experimental results demonstrate that the accuracy and detection speed of Faster R-CNN are greatly improved compared to R-CNN and Fast R-CNN because it uses RPN to replace the time-consuming Selective Search.

In the future, more in-depth testing and analysis can be conducted on the robustness of Faster R-CNN based on current work. For example, the defence capabilities of Faster R-CNN against adversarial samples can be tested through experiments. In addition, the testing of small data sets is also important to work directly in the future. At present, Faster R-CNN uses the transfer learning method to train in the existing dataset and fine-tune the trained model. If the object is not in the dataset used for pre-training, Faster R-CNN may not achieve good results.

References

[1] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," *2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus*, OH, pp. 580-587, 2014.

[2] R. Girshick, "Fast R-CNN," *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, pp. 1440-1448, 2015.

[3] S. Ren, K. He, R. 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, 2017.

[4] M. A. E. Muhammed, A. A. Ahmed and T. A. Khalid, "Benchmark analysis of popular ImageNet classification deep CNN architectures," *2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon)*, Bangalore, pp. 902-907, 2017.
[5] G. Huang, Z. Liu, L. Van Der Maaten and K. Q. Weinberger, "Densely Connected Convolutional Networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, pp. 2261-2269, 2017.
[6] X. Wang, A. Shrivastava and A. Gupta, "A-Fast-R-CNN: Hard Positive Generation via Adversary for Object Detection," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, pp. 3039-3048, 2017.
[7] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, pp. 779-788, 2016.
[8] B. Liu, W. Zhao and Q. Sun, "Study of object detection based on Faster R-CNN," *2017 Chinese Automation Congress (CAC)*, Jinan pp. 6233-6236, 2017.
[9] K. He, X. Zhang, S. Ren and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904-1916, 2015.
[10] I. Buzcu and A. A. Alatan, "Fisher-selective search for object detection," *2016 IEEE International Conference on Image Processing (ICIP)*, Phoenix, AZ, pp. 3633-3637, 2016.
[11] A. Pathak, M. Pandey and S. Rautaray, "Application of Deep Learning for Object Detection", *Procedia Computer Science*, vol. 132, pp. 1706-1717, 2018.
[12] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu and M. Lew, "Deep learning for visual understanding: A review", *Neurocomputing*, vol. 187, pp. 27-48, 2016.
[13] F. Sultana, A. Sufian and P. Dutta, "A Review of Object Detection Models Based on Convolutional Neural Network", *Intelligent Computing: Image Processing Based Applications*, pp. 1–16, 2020.
[14] K. E. A. van de Sande, J. R. R. Uijlings, T. Gevers and A. W. M. Smeulders, "Segmentation as selective search for object recognition," *2011 International Conference on Computer Vision, Barcelona*, pp. 1879-1886, 2011.
[15] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *International Conference on Learning Representations (ICLR)*, 2014.