

# Object Detection using Faster R-CNN

Xin Li, Sihan Wang, Yawen Zhao, Chen Zhao  
{xil222, siw003, yaz005, c8zhao@eng.ucsd.edu}

**Abstract**—This project is aiming at solving object detection problems on images with large varieties in fields of size, angle, posture. Instead of using inefficient and traditional region-based convolutional neural networks(R-CNNs), with producing candidate bounding box with CPU and passing CNN one by one, we choose Faster-RCNN, a method approaching real-time rates, ignoring the time spent on region proposals and thus realizing the speed improvement. We will test this method on Pascal VOC dataset and the results will be shown in this report.

**Index Terms**—Object Detection, RCNN, Fast-RCNN, Faster-RCNN, Pascal VOC

## I. INTRODUCTION

Object detection is currently a popular topics which has been applied to lots of fields like facial recognition, image retrieval and video object segmentation. The basic idea of object detection is to detect objects in a given image with precise positioned-bounding boxes and correct class labels. The difficulty of solving this problem is that the varieties of size, angle, posture of objects and classes. Besides, the object can be shown in any position of this image with either complete or not complete features of it, and thus will make the problem challenging. In order to deal with such large data variability, we need an algorithm with high computational speed. Fortunately, the success of region proposal methods and region-based convolutional neural networks(R-CNNs) realize the high-speed development in object detection.

When solving the problem concerning object detection, a naive idea was firstly brought up, which take different regions of interest from the image, and use a CNN to realize the classification of the object within that region. The drawback of this method is that the objects of interest might have different spatial locations in the image and different aspect ratios. In this case, we need to choose a large number of regions and the computational burden is pretty high. Hence, R-CNN [1] has been proposed with region proposal algorithms, like Selective Search [2], which extract a certain number of regions from the huge number of regions to solve the computational problem.

However, region proposal methods rely on inexpensive features and economical inference schemes. Even Selective Search, one of the most popular methods which greedily merges superpixels based on engineered low-level features, still consumes much more running time than the efficient detection network. Besides, most of the region proposal algorithm are fixed algorithms where no learning process is happening at that stage. In this case, some bad region proposals may be generated and lead to worse performance. To improved R-CNN, Fast R-CNN [3] is proposed. Instead of feeding the region proposals to the CNN every time, Fast

R-CNN feeds the input image to the CNN to generate a convolutional feature map one time per image and thus is significant faster in training and testing sessions over R-CNN. However, we still need to identify the region of proposals from the convolutional feature map, which slows down the algorithm significantly. Thus, Faster R-CNN [4] is introduced to solve the bottleneck caused by region proposals. A separate network, Region Proposal Networks (RPNs), is used to predict the region proposals. As RPNs share convolutional layers with object detection networks, Faster R-CNN can be much more efficient than Fast R-CNN and even can realize real-time object detection.

In our project, we are going to realize Faster R-CNN to solve object detection.

## II. DATASET

We use the datasets from Pascal VOC [5] challenge. This dataset keeps updated till 2012. We choose the training dataset from the Pascal VOC 2012 and testing dataset from the Pascal VOC 2007.

### A. Description

TABLE I  
SIZE OF PASCAL VOC

	train		val		test	
	images	objects	images	objects	images	objects
2007	2501	6301	2510	6307	4952	12032
2012	5717	13609	5823	13841	11540	27540

The PASCAL VOC datasets come from VOC challenges, which are short for Visual Object Classes. The PASCAL VOC project provides image data sets for object class recognition and also provides a common set of tools for accessing the data sets and annotations. In VOC2007, all annotations are available (i.e. for training, validation and test data), but since then the organizers have not made the test annotations available. Therefore, we use the "trainval" (training + validation) set VOC2012(the latest one) to train and fine tune our model and use the test set of VOC2007 version to test our model.

There are twenty classes, four categories in the dataset.

- Person: person
- Animal: bird, cat, cow, dog, horse, sheep
- Vehicle: aeroplane, bicycle, boat, bus, car, motorbike, train
- Indoor: bottle, chair, dining table, potted plant, sofa, tv/monitor.

### III. RELATED WORK

Throughout the history of realizing object detection, there were some related models in realizing this goals: RCNN, Fast-RCNN.

1) *R-CNN*: R-CNN mainly consists of four steps: first using selective search to initialize bounding box from the image; then pass selected images into CNN network to extract features. Third step is using SVM to classify whether object in the bounding box. Then we use the label as well as bounding box location for training.

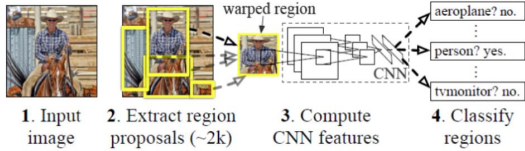


Fig. 1. RCNN architecture

2) *Fast-RCNN*: Fast-RCNN made progress on the R-CNN model, instead of using Selective Search which is an inefficient way of generating bounding box, and passing images to CNN one by one. It combines the SVM classification and bounding box regression into one part. Besides, after validating the region of proposals from the convolutional feature map, a ROI(Region of Interest) pooling layer is realized to reshape them into a fixed size so that they can be put into a fully connected layer.

### IV. METHODS

Compared to previous R-CNN and Fast-RCNN, Faster-RCNN model is more efficient as it implements the Regional Proposal Network which utilizes neural network to take care of the generating bounding box process.

#### A. Terms

1) *IoU*: Intersection over Union is a measure of the accuracy of detecting a corresponding object in a particular data set. We can see many of the practices of using this standard in many object detection challenges.

Usually we use this method to detect its performance in Convolutional Neural Network detectors. IoU is a simple measurement standard. Any task can be measured with IoU as long as there is a bounding box in the output. In order for IoU to be used to measure objects of any size and shape, we need the ground-truth bounding boxes and the range of results from our algorithm.

That is to say, this standard is used to measure the correlation between real data and prediction. The higher the correlation, the higher the value. IoU is equivalent to the result of dividing the overlap of two regions by the collection of the two regions. In general, when  $IoU > 0.5$  can be considered a good result.

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

2) *Anchor*: To fully consider all the possible region proposals, Faster-RCNN uses a sliding-window method which is called the anchor-based method. The sliding windows are on a feature map instead of the original image. At each location on the feature map, we predict k different bounding boxes centered at current sliding position, and those proposals are called anchors.

An anchor is centered at the sliding window, and is associated with 2 attributes: scale and aspect ratio.

In generating potential bounding box, at each anchor, we have three scales 8, 16, 32, three ratios 0.5, 1, 2. For a feature map with dimension  $W * H$ , we have  $W * H$  anchors, and  $3 * 3$  bounding boxes for each anchor, overall  $9 * W * H$ .

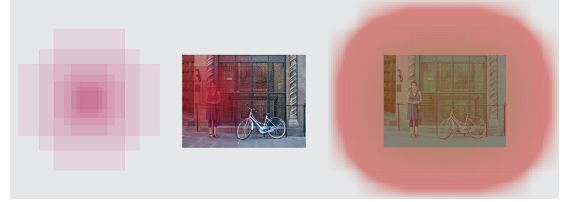


Fig. 2. Left: Anchors, Center: Anchor for a single point, Right: All anchors

#### B. Models

Faster RCNN consists of mainly four parts:

1) *Conv Layers*: As a CNN network target detection method, Faster RCNN firstly uses a set of basic Conv+ReLU+pooling layers to extract image feature maps. The feature maps are shared for subsequent RPN layers and fully connected layers.

2) *Region Proposal Network*: The RPN network is used to generate region proposals. This layer uses softmax to determine that the anchors belong to the foreground or background, and then use the bounding box regression to correct the anchors to obtain accurate proposals.

3) *ROI Pooling*: This layer collects the input feature maps and proposals, synthesizes the information and extracts the proper feature maps, and sends them to the subsequent fully connected layer to determine the target category.

4) *Classification*: Finally it uses the proposal feature maps to calculate the class of the proposal, and at the same time obtain the final precise position of the detection frame with bounding boxes.

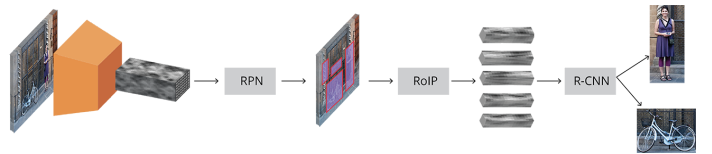


Fig. 3. Faster RCNN module

Each convolution layer uses the previous network information to generate an abstract description. The first layers generally learn edge edges information, and the second layer learns pattern patterns in edge edges to learn more complex

shapes and other information. Finally, the convolution features can be obtained. The spatial dimension is much smaller than the original image, but deeper. The width and height of the feature map are reduced due to the pooling layer between the convolutional layers, and the depth is increased by the number of filters learned by the convolutional layer.

The classic detection method generates a detection frame that is very time consuming. For example, R-CNN uses the Selective Search method to generate a detection frame. The Faster RCNN discards the traditional sliding window and SS method, directly uses the RPN to generate the detection frame. This is also a huge advantage of the Faster R-CNN, which can greatly improve the detection frame generation speed.

The RPN network is actually divided into two lines, one for the foreground and background by the softmax classification anchors, and the other for calculating the bounding box offset for the anchors to obtain the precise proposal. The final proposal layer is responsible for synthesizing foreground anchors and bounding box traction offsets to acquire proposals, while dropping proposals that are too small or out of bounds. In fact, the entire network to the Proposal Layer here, complete the equivalent of the target positioning function.

After RPN processing, we can get a bunch of object proposals without class score. The problem now is how to use these bounding boxes bounding boxes and classify them. One of the easiest ways is to crop each proposal, feed it into a pre-trained base network, and extract the features; then, extract the features to train the classifier. However, this requires calculations for all 2000 proposals, low efficiency and slow speed. The computational efficiency is accelerated by reusing the conv feature map, that is, a fixed-size feature map is extracted for each proposal using RoI (region of interest) Pooling.

The RoI Pooling layer is responsible for collecting the proposal and calculating the proposal feature maps for delivery to the subsequent network. The RoI pooling layer has 2 inputs: original feature maps and proposal boxes for RPN output (different sizes).

The Classification uses the proposal feature maps, through the full connect layer and softmax function, to calculate which class each proposal belongs, and outputs the probability vector. At the same time, the bounding box regression is used again to obtain each proposal. The position offset, in order to return a more accurate object detection frame.

### C. Post processing

Similar to RPN, we end up with a bunch of objects with classes assigned which need further processing before returning them. In order to adjust the bounding box, we need to consider the proposals of the category with the highest probability. The maximum value of the ignore probability is the proposal of the background class. When the final objects are obtained, and the result predicted as background is ignored, the class-based NMS is used, mainly by grouping the objects according to the class, then sorting according to the probability, applying NMS processing for each independent

group, and finally put it together. For our final list of objects, we also can set a probability threshold and a limit on the number of objects for each class.

### D. Training

In the paper, Faster R-CNN was trained using a multi-step approach, training parts independently and merging the trained weights before a final full training approach. Since then, the joint training of end-to-end has been found to give better results. When the complete model, 4 different losses are obtained, 2 for RPN, and 2 for R-CNN. The training of the base network depends on the objects to be learned and the available computing power. If the new data is similar to the original data set of the base basic network training, it is not necessary to train unless you want to try different performances. The training of the network is relatively high in comparison time and hardware consumption, and needs to be adapted to the gradient calculation.

Four different losses are organized in a weighted sum. We can set weights for the classification loss and regression loss as needed, or set different weights for R-CNN and RPNs.

Similar to the detection network, feature maps are still extracted using Conv Layers. The Loss used throughout the network is as follows:

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*)$$

In the above formula,  $i$  denotes the anchors index,  $p_i$  denotes the foreground softmax probability, and  $p_i^*$  represents the corresponding GT predict probability (i.e., when the IoU between the  $i^{th}$  anchor and GT > 0.7, it is considered that the anchor is Foreground,  $p_i^*=1$ ; otherwise, when IoU < 0.3, it is considered that the anchor is background,  $p_i^*=0$ ; as for those anchors with  $0.3 < \text{IoU} < 0.7$ , they are not involved in training.  $t$  stands for predict bounding box, and  $i^*$  represents the GT box corresponding to the foreground anchor.

### E. Evaluation

The evaluation is done using the standard Mean Average Precision at some specific IoU threshold. mAP is a metric that comes from information retrieval, and is commonly used for calculating the error in ranking problems and for evaluating object detection problems.

## V. RESULTS

We take the implementation version of fast-rcnn from github [6] as our training and testing code base. We trained VGG16 [7] and ResNet101 [8] model on pascal 2012 dataset using 1080Ti[3] from scratch. Due to GPU limit, we trained with batch size of 1. In training, ResNet101 takes nearly 2 hours per epoch and VGG16 takes around 1.5 hrs per epoch. Since the annotated test set for VOC 2012 has not been released, we are using VOC 2007 test set for benchmark. The advantage is that we can compare our result to the result trained on VOC 2007 training set. Here is the parameters for our training models:

Due to the GPU limit, we choose batch size of 1 for training both models. In terms of learning rate, we choose 1e-3 because

TABLE II  
TRAINING PARAMETERS

	GPU	Batch Size	Lr	Epoch	Time
VGG16	1	1	1e-3	12	18 hrs
ResNet101	1	1	1e-3	7	13 hrs

a large number may cause overfitting, a too many number will slow down the training process and sometimes lead to training process get stuck in local min. We use SGD optimizer as it is an efficient optimizer for training in this problem. We set 12 epoches for VGG16 and 7 epoches for ResNet101, because from figure4, figure6, we can see it achieves similar result. We use this as the benchmark for demoing the detection result on more complexed pictures.

1) *VGG16*: Figure2 and figure3 represents the detection result of trained VGG16 model on two different images. Of the two, figure2 is relatively simple. There are only two classes: person and dog. VGG16 model is able to correctly identify both of them, but if we look closely, we can see the nose of the dog is a bit outside of the bounding box, which implies the bounding box is a bit off from the ground truth. Then take a look at the figure3, there are lots of chairs and different objects inside this image, some of them were not found. Meanwhile, we notice there is misclassification on the monitor at the right corner.

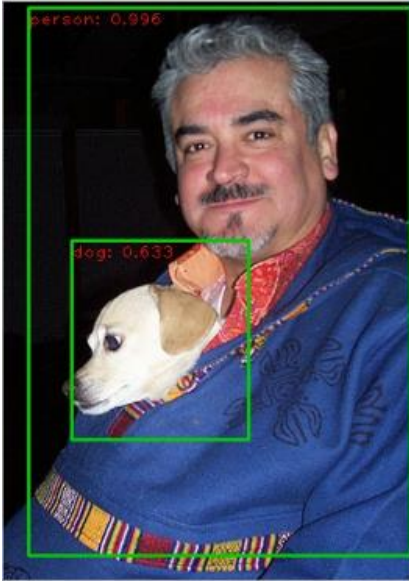


Fig. 4. Detected object of Image 1 of VGG16

2) *ResNet101*: Figure4 and figure5 represents the detection result ResNet101 on the two different images. Regarding the figure 4, ResNet101 model is able to correctly identify both of them, and the bounding box is more tighted compared to the box generated for by VGG16, which implies the bounding box is more closer to the ground truth. Then take a look at the figure5, there is misclassification on the closet; however, this model is able to detect more objects in the picture.



Fig. 5. Detected object of Image 2 of VGG16

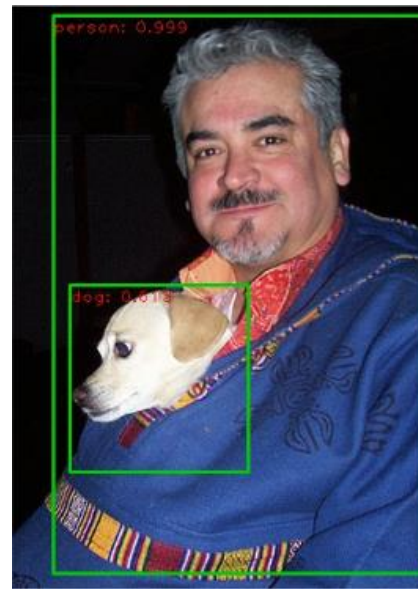


Fig. 6. Detected object of Image 1 of ResNet101

3) *Comparison*: Above are object detection results of our trained models on two different images. We use these two set of parameters for comparison because they have similar result in the first image. Both trained models are able to detect most of the objects. For VGG16, it has misclassification on the television, and for ResNet101, it has a misclassification at the top. But comparing the bounding box location vs the real object, it is not hard to see ResNet101 is slightly better in precisely detecting project. This observation is also reflected in mean average precision(mAP) of VOC 2007 test set. The resulted mAP on both models are better than those trained on VOC 2007 training set VGG16(0.724 vs 0.701), ResNet101(0.769 vs 0.75) which makes sense because VOC 2012 is a more complexed dataset. Overall, the mAP reaches the state of the art as we want, but according to observation, there are still potentials to improve.



Fig. 7. Detected object of Image 2 of ResNet101

## VI. CONCLUSION

In this project, we are using Faster-RCNN to realize object detections among Pascal VOC. Two different CNN models, VGG16 and ResNet101, are trained and tested. It shows that these two models reach similar results in object detection. However, ResNet101 with more layers, takes longer time to train and achieves better results which is reflected by observation and also shows in mAP. Besides, there are some drawbacks for our current algorithm. Some objects can not be detected or will be misclassified, and the mAP for some objects with unusual features will reach pretty low value. For our future work, if we have more resources for training, we may train more models with different parameters to see if there will be any improvement. Besides, there are some popular directions to improve Faster-RCNN recently. First, improving feature extraction network, such as using IncRes v2, ResNet. Second, improving RPN, reduce the number of proposal and improve the accuracy. Third, improving ROI Pooling, with multi-task benefits algorithm, MASK R-CNN and multi-layer roi-pooling, DeepText.

## REFERENCES

- [1] Girshick, Ross B. et al. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. 2014 IEEE Conference on Computer Vision and Pattern Recognition (2014): 580-587.
- [2] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. IJCV, 2013.
- [3] Ross Girshick, Fast R-CNN, The IEEE International Conference on Computer Vision(2015): 1440-1448
- [4] Ren, Shaoqing, Kaiming He, Ross B. Girshick and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence 39 (2015): 1137-1149.
- [5] The PASCAL Visual Object Classes Homepage(2005–2012) <http://host.robots.ox.ac.uk/pascal/VOC/>
- [6] <https://github.com/jwyang/faster-rcnn.pytorch/tree/pytorch-1.0>
- [7] Karen Simonyan, Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:1409.1556
- [8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition. arXiv:1512.03385