Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 6, No. 3, 2023

Research on Convolution Neural Network for Object Detection and Recognition

Tran Anh Tu *, Pham Thi Phuong

Institute of Information Technology, Hanoi Pedagogical University 2, Hanoi, Vietnam

*** Corresponding author:** Tran Anh Tu

Abstract: The state-of-the-art object detection networks for natural images have recently demonstrated impressive performances. However, the complexity of object's shape and orientation exposes the limited capacity of these networks for strip-like rotated assembled object detection which are common in any dataset as well as real im- ages. In this project, I embrace this observation and introduce the Faster Rotated Region-based Convolutional Neural Network (Faster RR-CNN), which can learn and accurately extract features of rotated regions and arbitrary-oriented objects precisely. In comparison with the classic Faster RCNN, Faster RR-CNN has three important new components including a skew non-maximum suppression, a rotated bounding box regression model and a rotated region of interest (RRoI) pooling layer. I conduct experiments using the PASCAL VOC 2012 dataset, demonstrat-ing the potential ability of this novel network in detecting oriented objects.

Keywords: Convolution Neural Network; Object Detection and Recognition; Faster RR-CNN; PASCAL VOC 2012.

1. Introduction

Recently, advances in object detection are driven by progresses of backbone deep convolutional neu-ral networks [7], [15], [5], [6] and [16], and improvements of object detection frameworks including R-CNN [3], Fast R-CNN [2], Faster R-CNN [14], YOLO [13] or SSD [10]. In comparison with image classification, object detection is more complex due to the demand of precise localization. Inorder to satisfy that requirement, the above-mentioned detection framework combined with variousbackbones is based on different feature extraction strategies of regions. Several candidates of this approach could be listed such as Selective Search [17] or Edge boxes [18]. Additionally, in Faster R-CNN, instead of a special region proposal method, a region proposal network (RPN) [14] is used, reducing the running time by sharing convolutional layers with detection network.

Feature extraction strategy of regions is an important difference between these detection frameworks.In R-CNN [3], a special object proposal method (e.g. Selective Search [17]) which typically relies on simple feature and fast evaluation is used. Then on each generated proposal region, convolution operation is carried out directly from images without sharing computation, leading to inefficiency. A more advanced version is Fast R-CNN [2], which performs convolution operation on the entire image and extracts convolution features for each proposal in each feature map by a region of interest(RoI) pooling layer. Compared with R-CNN, Fast R-CNN saves much computation time, exposing proposal generation time as a bottleneck. However, the running time required for the latter is still very long. So as to reduce running time while still preserve the accuracy, Faster R-CNN [14] is introduced. This model uses a regional proposal network, which shares convolution computation with detection network, improving the quality of proposals.

Although these frameworks have shown a promising result, most of them rely on axis-aligned an- notations, and return prediction of horizontal regions. However, in the real-world scenario, a great number of object's regions or object's shapes are not axis-aligned. Applying axis-aligned proposalson such objects could result in poor accuracy prediction, as shown in figure 1 so axis-aligned pro- posal methods cannot be widely applied in every case.

In this project, I come up with a rotation-based approach and an end-to-end object detection for arbitrary-oriented object detection. Particularly, orientations are incorporated so that the detection system can generate proposals for objects with arbitrary orientation. Comparison on properties be-tween previous axis-aligned approach and mine are shown in table 1.

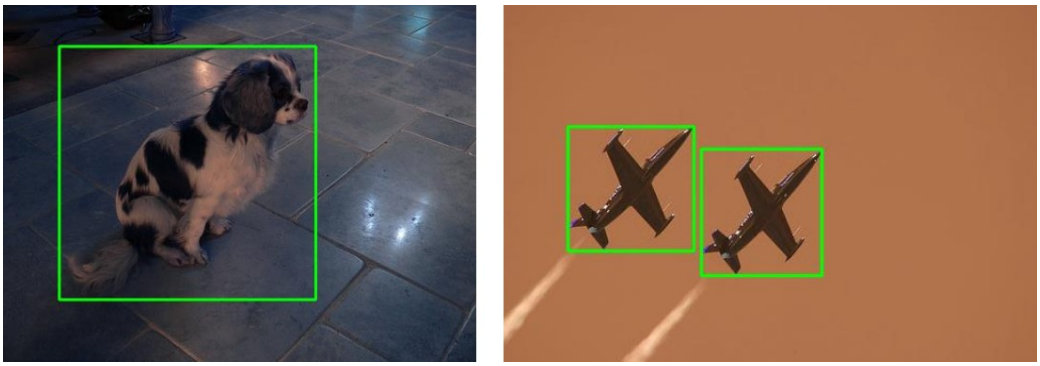

Figure 1. Non-optimal bounding boxes for object detection

Table 1. Properties of networks: feature extraction accurately for rotated regions, sharing convolu- tion computation and end-to-end training.

Network	Rotated region	Sharing computation	End-to-end
R-CNN	No	No	No
Fast R-CNN	No	Yes	No
Faster R-CNN	No	Yes	Yes
Yolo	No	Yes	Yes
SSD	No	Yes	Yes
My model	Yes	Yes	Yes

The main contributions of this work are threefold:

I propose the way to create oriented annotation for every single image. Based on the anno-tation for segmentation, I could create annotation for oriented detection, which is defined as the minimal-area bounding box covering an object.

I propose an efficient method for computing Intersection over Union (IoU) for rotated pro-posals. Based on that, a novel skew non-maximum suppression is applied so as to removeoverlapping rotated proposals efficiently

I introduce the Rotated Region of Interest (RRoI) pooling layer, an auxiliary structure, to extract features of rotated regions.

All of the contributions are incorporated to the Faster R-CNN model to lever the ability of this model: To localize oriented objects precisely. I call the new model as **Faster Rotated Region- based Convolutional Neural Network**, or **Faster RR-CNN**.

2. Related Work

Faster R-CNN [14], inherited from Fast R-CNN mainly focuses on finding region proposals with CNN, and shares features with object classification task followed by region proposals. The insight of Faster R-CNN was that region proposals depended on features of the image that were already cal-culated with the forward pass of the CNN (first step of classification). And if those CNN results forregion proposals are reused instead of running a separate selective search algorithm, the total run- ning time could be reduced significantly, with a low marginal cost for computing proposals. Thanksto these attractive features, Faster R-CNN is widely applied in many object detection applications.

Furthermore, it is also used as a baseline for many other advanced detection models, such as FeaturePyramid Network (FPN) [9], Region-based Fully Convolutional Networks (R-FCN) [8] or the verystate-of-the-art model Mask R-CNN [4]

3. Faster RR-CNN

My new model, called Faster RR-CNN is composed of a backbone network and auxiliary structures.The former could be any classical model such as AlexNet [7], VGG-16 [15] or ResNet [5]. The latterincludes 4 key components: The skew non-maximum suppression (**Skew NMS**, section 3.1), The skew Intersection over Union (**Skew IoU**, section 3.2), the non-maximum suppression multi-task loss function (**NMS multi-task loss function**, section 3.3) and the rotated region of interest poolinglayer (**RROI pooling layer**, section 3.4). The network architecture is illustrated in figure 2

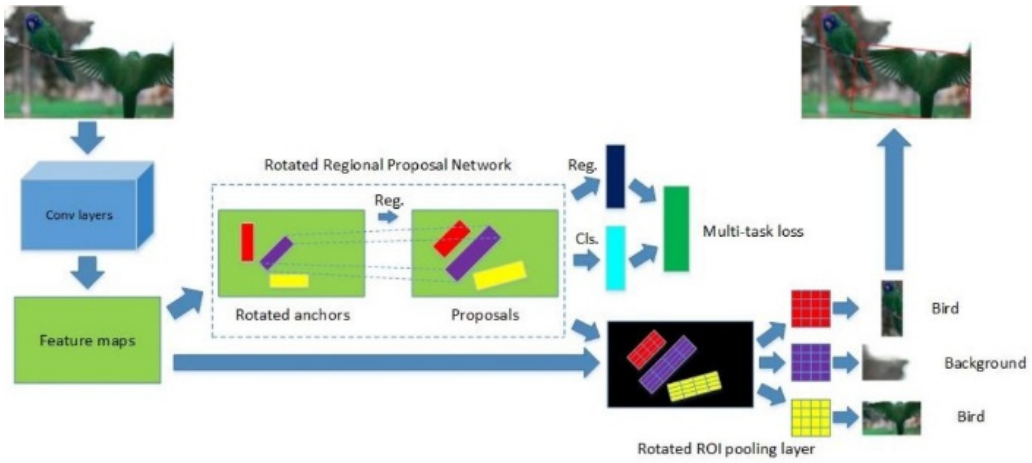

Figure 2. The network architecture of Faster RR-CNN

3.1. Skew Non-Maximum Suppression

Non-maximum suppression is widely seen as an effective way to remove highly overlapping pro- posals, and therefore reduces computational effort as well as running time. In classic Faster R-CNN model, non-maximum suppression's criterion is defined very simple: Skip proposals which overlap more than 70% of any chosen proposals. This module is applied in 2 parts of the Faster R-CNN model: Removing highly overlapping anchors, and removing highly overlapping predicted bound- ing box.

However, in the case of rotated bounding boxes, applying the classic non-maximum suppression arise some problems:

Quality of final list of proposals: Through experiments, as well as stated in [12] , I foundout that in many cases, 2 proposals, though overlap to each other more than 70%, cover quite different areas. The reason is that their orientations are highly different. Therefore, the classic non-maximum suppression [14] when applied on oriented proposals could leadto shortage list of proposals, since there are many useful proposals are skipped.

Computational complexity: The main core of non-maximum suppression module is the Intersection over Union module, in which the overlapping area between 2 sets of

proposalsare calculated. In the case of axis-aligned proposals, this calculation is quite simple, and thesame formula could be applied in every case, since the overlapping always have the same shape. However, in the case of rotated proposals, the shape of overlapping areas could varyin shape, making the task of finding a common way to calculate it to be more challenging.

Therefore, in order to resolve the 2 above-mentioned issues, based on the method proposed in [12],I apply a new Skew-NMS consisting of 2 phases:

Retrieve all proposals having IoU with ground truth larger than 0.75, then take the one withmaximal IoU.

If all proposals have IoU smaller than 0.75, keep the one with minimum angle differencewith ground truth, as long as the overlapped IoU larger than 0.4.

3.2. Skew Intersection over Union

The rotated proposals can be generated in any orientation, so IoU computation on axis-aligned proposals may lead to an inaccurate IoU of skew interactive proposals and further ruins the proposallearning. Generally, the overlapping area between 2 rotated proposals could be a polygon with different number of vertices: 0 (non-overlapped), 3 to 8 (overlapped). Several cases of 2 rotated rectangles' overlapping areas are illustrated in Figure 3.

Theoretically, the procedure described in [12] could be used to find out the overlapping area between 2 arbitrary-oriented rectangles:

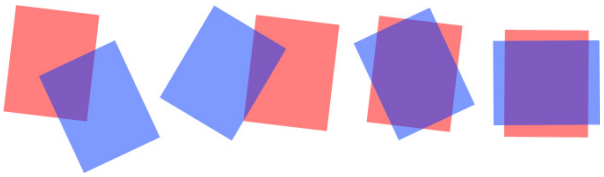

Figure 3. Overlapping areas between 2 arbitrary-oriented rectangles

1) Find all intersection points between 2 rectangles.

2) Find all vertices of the first rectangle inside the second rectangle.

3) Find all vertices of the second rectangle inside the first rectangle.

4) Calculate the area of the polygon constructed from all above found points by summing upthe areas of triangulation.

However, practically, we cannot loop over all pairs of rectangles to perform the above-mentioned procedure, since it could be super time-consuming. In this project, I propose a new way to calculate overlapping areas between 2 **sets** of rectangles **automatically** and **element-wise**. From programming aspect, the second attribute is very important, since it allows the calculation performed on element-wise operation supporting scientific computing packages such as Numpy or Cupy.

For simplicity, here I only analyze the case of calculating overlapping area (intersection) between2 rectangles, but it is straightforward to expand to the general case of 2 sets of rectangles. My algorithm is carried out as follow: Firstly, I define an 1D array with 24 elements, where:

The first 4 elements indicate whether each vertex of the first rectangle is inside the secondrectangle or not.

Elements 5-8 indicate whether each vertex of the second rectangle is inside the first rectan-gle or not.

Elements 9-12 indicate whether the first edge of the first rectangle intersects 4 edges of thesecond rectangle or not.

Elements 13-16 indicate whether the second edge of the first rectangle intersects 4 edgesof the second rectangle or not.

Elements 17-20 indicate whether the third edge of the first rectangle intersects 4 edges ofthe second rectangle or not.

Elements 21-24 indicate whether the last edge of the first rectangle intersects 4 edges of thesecond rectangle or not.

For the first 2 steps, I use the conclusion from following theorem:

Theorem 1. *Given a rectangle ABCD, a point M of coordinates (x,y) is inside the rectangle if andonly if:*

$$0 \leq AM.AB \leq AB.AB \textbf{ and } 0 \leq AM.AD \leq AD.AD$$

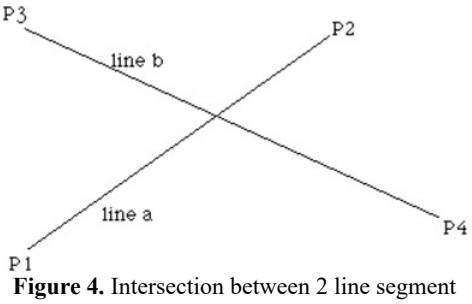

Figure 4. Intersection between 2 line segment

For the steps 3-6, algorithm is more complex. Given 2 line segments: line segment ***a*** with 2 end- points P1 and P2, and line segment ***b*** with 2 end-points P3 and P4 as illustrated in figure 4, we coulddefine the equation for 2 lines as below:

$$P_a = P1 + u_a(P2 - P1) \quad (1)$$
$$P_b = P3 + u_b(P4 - P3) \quad (2)$$

Solving the intersection point where $P_a = P_b$ gives the following 2 equations in 2 unknown variables(u_a and u_b):

$$x_1 + u_a(x_2 - x_1) = x_3 + u_b(x_4 - x_3)$$
$$y_1 + u_a(y_2 - y_1) = y_3 + u_b(y_4 - y_3)$$

Solving these 2 equations gives the following expressions for u_a and u_b:

$$u_a = \frac{(x_4 - x_3)(y_1 - y_3) - (y_4 - y_3)(x_1 - x_3)}{(y_4 - y_3)(x_2 - x_1) - (x_4 - x_3)(y_2 - y_1)} \quad (3)$$

$$u_b = \frac{(x_2 - x_1)(y_1 - y_3) - (y_2 - y_1)(x_1 - x_3)}{(y_4 - y_3)(x_2 - x_1) - (x_4 - x_3)(y_2 - y_1)} \quad (4)$$

Substituting either (3) or (4) into either (1) or (2) gives us the intersection point. The above method is used for calculating intersection between 2 arbitrary lines. But we need to calculate the intersectionbetween 2 **line segments**. One important condition for segment intersection is that:

$$0 \leq u_a \leq 1 \text{ and } 0 \leq u_b \leq 1$$

After performing point-inside-rectangle check and line-segment check between 2 rectangles, the 24-element array will be filled with the coordinates of vertices and intersection points if they belongto the intersection polygon. It should be noted that this array could only contain up to 8 non-zero elements, since the intersection polygon has maximum 8 vertices.

The next step is to sort all vertices in clockwise or anti-clockwise order. The way I use is to sort vertices based on their angle with horizontal axis. Finally, the following formula is used to calculatethe area of any regular or irregular, convex or concave n-vertex polygon:

$$\textbf{area} = \left| \frac{(x_1y_2 - y_1x_2) + (x_2y_3 - y_2x_3) + \ldots + (x_nx_1 - y_ny_1)}{2} \right|$$

The whole method above when applied on set of NxM rectangles will result in an NxM matrix whose elements representing intersection areas. Then combined with the calculation of union areas,which is straightforward, the NxM matrix of IoU is retrieved. Again, it is worth reminding that though there are some existing methods performing IoU calculation (e.g. [12]), to the best of my knowledge, my method is the first one applying the same procedure for all cases of relative positionbetween 2 (sets of) rectangles, which makes it not only theoretically but also practically applicable.

3.3. Non-Maximum Suppression Multi-task loss function

In order to build a regression model for rotated bounding boxes, we need to define a loss function calculated on each proposal. In my project, the loss function is the summation of 2 losses: One for object classification task and one for rotated bounding box regression task. Mathematically, my loss function is defined as below:

$$L\ (p, t) = L_{cls}(p, p^*) + \lambda p^* L_{reg}(t, t^*) \qquad (5)$$

In 5, the classification loss L_{cls} is log loss over M + 1 classes (M different object's classes plus the background class). p and p^* are predicted and ground truth classes respectively. For regression loss,I adopt the smooth L_1 loss as below:

$$L_{reg}(t, t^*) = \sum_{i \in \{x,y,w,h,\theta\}} smooth_{L1}(t_i^* - t_i) \qquad (6)$$

where:

$$smooth_{L1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

x, y, w, and h denote proposals center coordinates and its width and height. θ indicates the angle between the longer side of proposal and x-axis. t and t^* are predicted and ground truth bounding boxes respectively. The specific form of these 10 parameters (5 for predicted and 5 for ground truthbounding boxes) are described in [11]. In 5, the term p^* coming along with regression loss acts as atrigger, activated only for object classes, and disabled for background class. Parameter λ indicates the trade-off between these 2 losses. In my experiments, this parameter is set to 1, resulting in a balancing importance between classification and regression losses.

3.4. Rotated Region of Interest Pooling Layer

As presented in Fast-RCNN [2], region of interest (RoI) pooling layer extracts a fixed-length featurevector from the feature map for each proposal. Each feature vector is then fed into fully connected layers that finally branch into the sibling classification and regression layers. The RoI pooling layer uses max pooling to convert the feature inside any valid RoI into a small feature map with a fixed spatial extent of size h_{roi} x w_{roi}, where h_{roi} x w_{roi} are RoI-independent hyper-parameters.

For the arbitrary-oriented object detection task, the traditional RoI pooling layer becomes insuffi- cient because it can only handle axis-aligned proposals, hence I present the rotated region of interestpooling layer (RRoI pooling layer) which could flexibly handle any arbitrary-oriented proposals. Itshould be mentioned here is that although the idea about this layer was firstly introduced in [12], the way I implement it here is totally different. In [12], the max pooling operation is performed directly on oriented proposals, resulting in slow computation mainly due to the process of extract- ing maximum element in each oriented sub-region inside a proposal. In my approach, firstly each proposal is rotated to axis-aligned direction, then traditional RoI pooling could be applied on that new proposal. From programming perspective, my approach takes advantage of element-wise oper- ations, accelerating computation speed. My RRoI pooling layer is illustrated in figure 5, with the 2hyper-parameters h_{roi} and w_{roi} are set to 7, as suggested in [12].

4. Experiments

4.1. Dataset

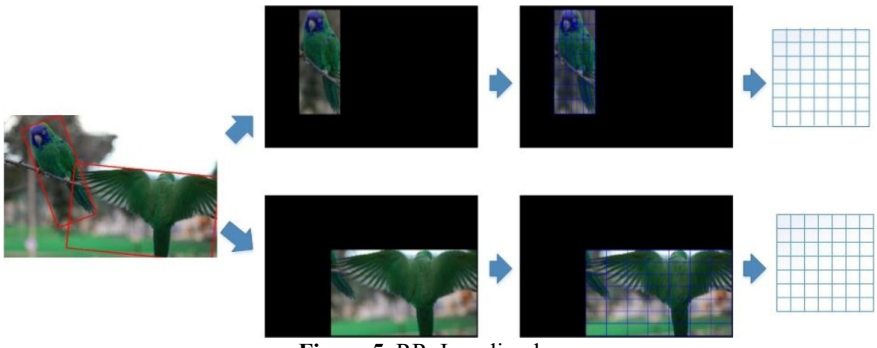

Figure 5. RRoI pooling layer

Figure 6. Axis-aligned ground truth versus arbitrary-oriented ground truth: The left figure displays a ground truth provided in PASCAL VOC 2012 dataset for detection task. The right figure displays a ground truth I create by using ground truth in PASCAL VOC 2012 dataset for segmentation task.

I evaluate my model on the PASCAL VOC 2012 dataset. Because I do not have available ground truth for arbitrary-oriented bounding box for object detection task, I create it myself by using the ground truth for segmentation task. Specifically, by connecting near-by component of objects, I could draw an oriented rectangle which covers any object with minimum area. One example of provided ground truth for detection task, and ground truth I create from segmentation task are shownin figure 6.

Table 2 demonstrates that by using oriented bounding box instead of axis-aligned bounding box, objects could be localized more efficiently by using smaller bounding box but still covering whole object. It should be noted here is that in PASCAL VOC 2012 dataset, there are more ground truth for detection task than segmentation task. For a fair comparison, I only collect images having ground truth in both tasks to train the classic Faster R-CNN model and our Faster RR-CNN model.

Table 2. Comparison between 2 sets of ground truth

	Original ground truth	**Our ground truth**
Number of images	2913	
Number of objects per image*	2.380	
Number of object-related pixel per object*	18957.812	
% of object-related pixels per bounding box*	53.5%	**56.7%**
Bounding boxs area*	35453.926	**33462.425**

*Average value

4.2. Implementation

My network is initialized by a pre-trained VGG16 model. The feature maps from the last convolu-tional layer are used for region proposals and fed into RRoI pooling layer. For generating anchors, apart from 3 values for scales (8,16,32), 3 value for ratios (1:1,2:1,1:2) as suggested in [14], I use3 values $\pi/6$, 0 and $\pi/6$ for angles, resulting in 27 different anchors for each position of center point. Initial learning rate is set to 0.001, and reduced by a half after 50000 iterations as suggested in [1]. Training phase ends at iteration 70000. Both classic Faster R-CNN as well as my Faster RR-CNN are trained with NVIDIA Quadro P6000 24 GB for 12 and 22 hours respectively.

4.3. Results

I use mean Average Precision (mAP) to measure my model's as well as the classic Faster R-CNN'sperformance. There are 20 different classes in PASCAL VOC 2012 dataset. The comparison in performance in all classes as well as in average are shown in Table 3, while some sample results are shown in figure 7. It could be seen from table 3 is that my model outperforms Faster R-CNN in 11/20 classes. On average, my model also performs slightly better than Faster R-CNN, demonstrating the potential capability of using oriented bounding box for better detecting and localizing objects in real applications.

Table 3. Mean Average Precision on Faster R-CNN and Faster RR-CNN on PASCAL VOC 2012 dataset

Class	**Faster R-CNN**	**Faster RR-CNN**
Monitor	67.6	**73.6**
Car	**80.4**	80.1
Cat	**82.0**	79.1
Sheep	68.3	**72.2**
Train	**81.1**	72.2
Horse	79.8	**80.1**
Motorbike	75.0	**77.2**
Cow	**75.3**	73.3
Aeroplane	**70.0**	69.2
Person	76.3	**77.7**
Bus	78.2	**81.7**
Table	**67.2**	67.0
Boat	57.3	**59.1**
Plant	39.1	**41.7**
Bird	**70.1**	68.4
Dog	80.3	**82.3**
Sofa	**67.3**	64.8
Bottle	49.9	**54.6**
Chair	52.2	**53.6**
Bicycle	**80.6**	78.7
Average	70.0	**70.6**

5. Conclusion

I have presented Faster Rotated Region-based CNN (Faster RR-CNN) for efficient and accurate rotated object detection and localization. It shows some initial promising results in comparison withits well-known parent Faster R-CNN on the standard PASCAL VOC 2012 dataset. There are still rooms for improving this model, for example accelerating training speed by approximate instead of absolute rotation of proposals, or using super-pixel for calculating intersection over union. Certainly, there is a trade-off between speed and accuracy. I will experiment these approaches carefully to make sure that the accuracy sacrificed is still within acceptable threshold.

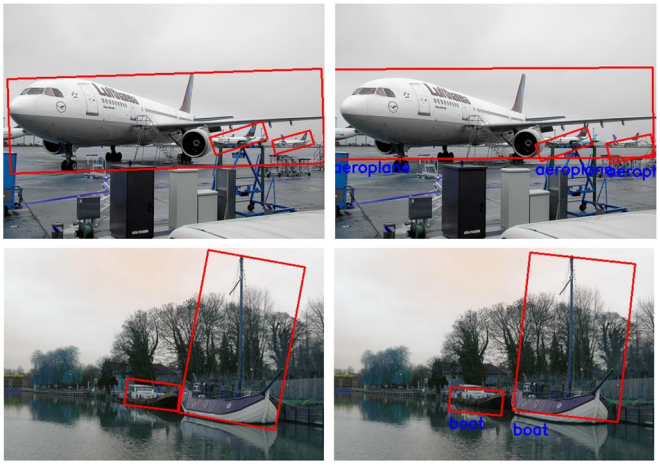

Figure 7. Sample results produced by Faster RR-CNN. The left figures are ground truth. The right figures are predictions from Faster RR-CNN

References

[1] X. Chen and A. Gupta. An implementation of faster rcnn with study for region sampling. arXiv preprint arXiv:1702.02138, 2017.

[2] R. Girshick. Fast r-cnn. In Proceedings of the IEEE international conference on computer vision, pages 1440–1448, 2015.

[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 580–587, 2014.

[4] K. He, G. Gkioxari, P. Dolla´r, and R. Girshick. Mask r-cnn. arXiv preprint arXiv:1703.06870, 2017.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Pro- ceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

[6] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. Densely connected convolutional networks. arXiv preprint arXiv:1608.06993, 2016.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.

[8] Y. Li, K. He, J. Sun, et al. R-fcn: Object detection via region-based fully convolutional net- works. In Advances in Neural Information Processing Systems, pages 379–387, 2016.

[9] T.-Y. Lin, P. Dolla´r, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. arXiv preprint arXiv:1612.03144, 2016.

[10] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In European conference on computer vision, pages 21–37. Springer, 2016.

[11] Z. Liu, J. Hu, L. Weng, and Y. Yang. Rotated region based cnn for ship detection.

[12] J. Ma, W. Shao, H. Ye, L. Wang, H. Wang, Y. Zheng, and X. Xue. Arbitrary-oriented scene text detection via rotation proposals. arXiv preprint arXiv:1703.01086, 2017.

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

[14] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems, pages 91–99, 2015.

[15] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recog- nition. arXiv preprint arXiv:1409.1556, 2014.

[16] D. Stutz. Understanding convolutional neural networks. InSeminar Report, Fakulta¨t fu¨r Math- ematik, Informatik und Naturwissenschaften Lehr-und Forschungsgebiet Informatik VIII Com- puter Vision, 2014.

[17] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. International journal of computer vision, 104(2):154–171, 2013.

[18] C. L. Zitnick and P. Dolla´r. Edge boxes: Locating object proposals from edges. In European Conference on Computer Vision, pages 391–405. Springer, 2014.