RR-FCN: Rotational Region-Based Fully Convolutional Networks for Object Detection

Dingqian Zhang[1,3](✉), Hui Zhang[2], Haichang Li[1], and Xiaohui Hu[1]

[1] Institute of Software Chinese Academy of Sciences, Beijing, China
{haichang,hxh}@iscas.ac.cn
[2] Beijing IrisKing Co., Ltd., Beijing, China
zhanghui@irisking.com
[3] University of Chinese Academy of Sciences, Beijing, China
zhangdingqian15@mails.ucas.edu.cn

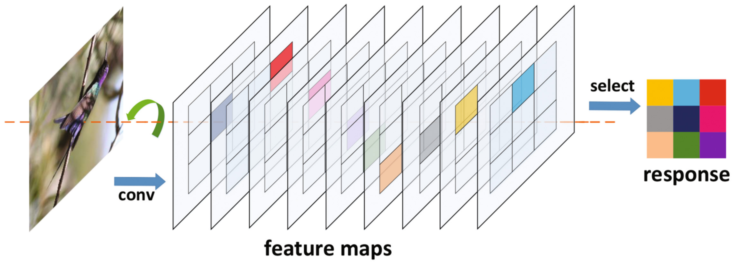
Abstract. In this paper, we present rotational region-based fully convolutional networks (RR-FCN) for object detection. In contrast to previous detectors that do not consider rotation, our region-based detector incorporates rotational invariance into networks efficiently and generate more appropriate features according to the rotation angle. Specifically, we propose component-sensitive feature maps, rotational RoI pooling and interceptive back propagation which make RR-FCN learn rotation situations without extra supervision information. Using the 101-layer ResNet model, our method achieves state-of-the-art detection accuracy on PASCAL VOC 2007 and 2012. Moreover, since the feature maps in our network are component-sensitive, RR-FCN can find out objects with various postures, even those appear rarely in the training set. So our RR-FCN has better performance in the real world.

Keywords: Object detection · Rotational invariance
Fully convolutional network

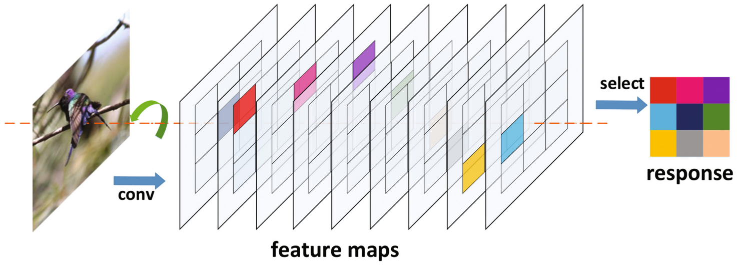
1 Introduction

Recently, a series of deep network object detection methods have been proposed, such as fast R-CNN [5], faster R-CNN [15] and R-FCN [2]. These methods divide deep networks into two parts with Region of Interest (RoI) Pooling layer [5]: (i) a CNN-based part for extracting features from the whole image, and (ii) a part that is associated with RoI for classifying each proposal generated by RoI pooling layer. Because the second part gets location-aware proposals, the networks have the ability to detect objects in different places. However, object detection also requires rotational invariance. Rotational invariance means if an object rotates around its geometrical center, the prediction should be the same.

This work is supported by the Natural Science Foundation of China (U1435220) (61503365).



(a)



(b)

Fig. 1. Illustrations of our idea. Our network selects features from different locations on the feature maps, depending on the object under detecting. The images are the same in (a) and (b), but the rotation angles are different. RR-FCN gives them different feature selection methods. The colored bins mean they are selected.

Previous detectors train their parameters without considering rotational situations, which are called "*rotation tolerating*" in this paper. Instead of tolerating rotation, our network is aimed to understand rotation and select appropriate features. We call our process mode "*rotation handling*".

In this paper, we introduce a framework called Rotational Region-based Fully Convolutional Networks (RR-FCN) which can conveniently incorporate rotational invariance into deep networks for object detection. The idea is shown in Fig. 1. We use convolution operation to construct a group of component-sensitive feature maps. Each of these feature maps is sensitive to a specific component of an object, rather than a fixed part of region proposal. Then we use rotational RoI pooling proposed by us to select appropriate pieces of these feature maps to form the responses. Finally, we pick one of these responses as the output feature. Through this way, RR-FCN can deal rotation situations *without extra supervision information or parameters*. Moreover, since RR-FCN has component-sensitive feature maps, it can detect objects in various postures as long as the relative positions of components remain unchanged. So RR-FCN is more competitive in the real world.

In order to preserve the spatial information, we construct our network as a fully convolutional network. Recently, deep fully convolutional networks for computer vision become popular (e.g., [2,12,14]). Besides preserving spatial information of images, they also have fewer parameters and high computational speed. In [2], Dai et al. have solved the problem of translation variance fading in deep fully convolution network. In this paper, we figure out the detailed requests of translation invariance in the structure of fully convolutional networks. Further, we design our network according to these requests to preserve both translation variance and rotational invariance in our network.

Taking the 101-layer Residual Net (ResNet-101) [8] as our backbone and training it with interceptive back propagation developed by us, our RR-FCN has better robustness when the objects appear with rare postures. In simple terms, although people are barely reversed in the dataset, RR-FCN can still find an upside down person. Additionally, RR-FCN achieves state-of-the-art results. Our code will be made publicly available.

In conclusion, our main contributions are:

1. We propose a fully convolutional network, which can handle the rotation invariance of the targets with little extra computation time.

2. We preserve translation variance in the network by analysing and carefully designing the network structure.

3. We improve the robustness of the detection network. Our network can work normally even under the situations rarely appear in training set.

2 Related Works

In this section, we will discuss the previous works related to our work, covering the development of object detection and researches on rotational convolution networks.

2.1 Object Detection

Object detection networks can be divided into two kinds of frameworks: (i) detectors with region proposals [5,10,15], and (ii) detectors without region proposals [11,13,14]. In this paper, we focus on the first kind. Region-based object detection networks start using specialised pooling method from SPP-Net [7] (Spatial Pyramid Pooling) and Fast R-CNN [5] (Region of Interest pooling). Instead of resizing images into a fixed size, they pool the region proposals into a fixed size (e.g., 7×7). Then the fixed size outputs will be sent into various detectors to classify the related region proposals. R-FCN abandons fully-connected detector and shares the computation of the whole image for the first time. Without fully connected layers and deep RoI-wise sub-network, the detection precision becomes much lower than we expect. R-FCN uses position-sensitive RoI pooling to improve the detection result by solving translation variance problem in object detection, inspired by which we construct our networks. However, these methods all use the representation capacity and the powerful generalization ability to tolerate rotations of objects. We handle the rotational invariance problem in object detection which is also significantly important.

2.2 Rotational Convolution Networks

There is a research [1] indicating max pooling in CNN is beneficial to accept rotation. Max pooling ensures the output is constant when a few pixels on feature maps change their relative locations. However, it only works in the small region and ignores much useful information.

Several approaches have considered the rotation problems in convolutional networks. The most representative ones are Spatial Transformer Networks (STN) [9] and Deformable Convolutional Networks (DCN) [3]. STN adds a small network in parallel on the normal CNNs construction. This small network is capable of learning the affine transformation matrixes of images, in this way, it can accomplish translation, rotation, scale, etc. Although STN is successful in MNIST dataset, it fails to achieve desirable performance in object detection. This is due to the situation that detection datasets are relatively complex compared to MNIST. DCN also adds several convolution layers in parallel to guide the selection of sampling locations in convolutional operations. They both use bilinear interpolation to ensure back propagation can run during training. It is a success in semantic segmentation and improves the mAP in detection. However, DCN only selects the direction and distance of the expansion in convolution operations. Although its deformable position-sensitive Rol pooling is different from the previous pooling methods in the last sampling area, it still uses the relative position of the region proposal as a prior condition (e.g., the top-left part of a proposal). It cannot share rotational information across different feature maps. So the DCN focuses on deformation as its name indicates.

3 Our Approach

Figure 2 shows the basic architecture of our network. RR-FCN is an object detection network considering the rotation problem. Moreover, it incorporates rotational invariance into the network with no (or little) extra computation time. In this section, we describe the RR-FCN from three parts in detail.

3.1 Component-Sensitive Feature Maps and Rotational RoI Pooling

To make the feature maps in RR-FCN component sensitive, we develop a novel pooling method called rotational RoI pooling which pools feature maps clockwise. Figure 3 shows the relationship between component-sensitive feature maps and rotational RoI pooling. Each component-sensitive feature map is sensitive to a particular component of an object (e.g., head of a bird, leg of a dog). To distinguish different components, we divide every region proposal into $k \times k$ equal parts and each part has its corresponding feature maps. So we need to combine the $k \times k$ feature maps to get the entire detection result for one class. In our experiment, we make these components class-aware. That means each proposal has $k \times k \times (C + 1)$ feature maps. The C is the total number of categories in a dataset.

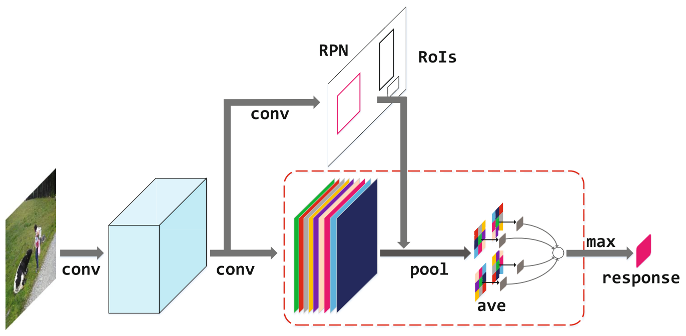


Fig. 2. The basic architecture of RR-FCN. We use different pooling methods with rotation information to get different responses from the same feature maps. We classify objects with the highest response. The proposed component-sensitive feature maps and rotational RoI pooling are marked with a red box. They can be used in parallel to obtain more specific information of rotation. Note that some back propagations in RR-FCN are not normal. (Color figure online)

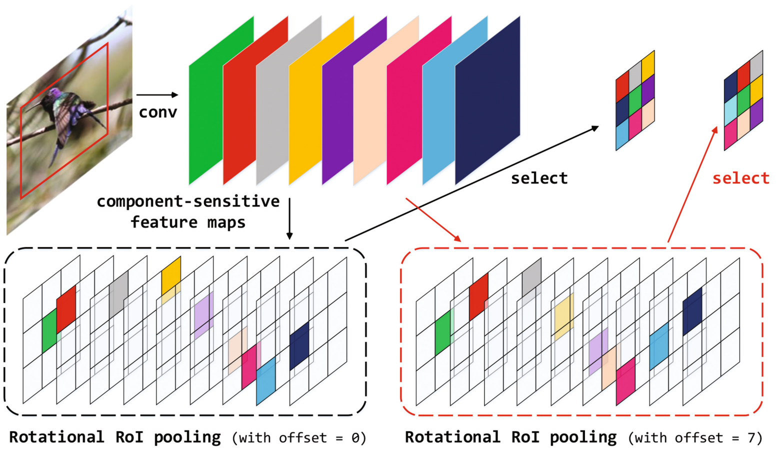

conv
component-sensitive feature maps
select
select
Rotational RoI pooling (with offset = 0)
Rotational RoI pooling (with offset = 7)

Fig. 3. Illustration of the component-sensitive feature maps and rotational RoI pooling. The feature maps may have multiple pooling methods with different offset f. So there are more than one output coming from a blank of feature maps. We take 0 and 7 offset for example.

Rotational RoI pooling layers pool component-sensitive feature maps into $k \times k$ bins with given rotation offset f which means the angle the features need to be rotated. f ranges from 0 to $4 \times (k - 1)$ and stands for the distance of outermost output bins moving. Although some methods can make f continuous and differentiable [3,9], we still use discrete f. Because f represents a channel choice, and the channel is an integer. Besides this, discrete f can simplify the computation.

For convenient coding and rigorous expression, instead of calculating the location on a feature map after rotation, we calculate which channel is selected at a specific position. And we mark it as $\varphi(i, j \mid f)$. Figure 4(a) is the situation of feature map channel selection for rotational RoI Pooling when offset f equals to 0 and k equals to 5 ($\varphi(i, j \mid 0)$).

9	10	11	12	13
24	1	2	3	14
23	8	0	4	15
22	7	6	5	16
21	20	19	18	17

(a)

15	16	17	18	19
14	4	5	6	20
13	3	0	7	21
12	2	1	8	22
11	10	9	24	23

(b)

Fig. 4. The selected channel of the (i, j)-th bin. The digit of each block is the serial number of a component-sensitive feature map. Different rotations correspond to different selections. Here, we give the examples with offset f equals to 0 (a) and 6 (b) when $k = 5$.

With the definition of $\varphi(i, j \mid 0)$, coupled with the use of mathematical knowledge such as polar coordinates, it is not difficult to derive $\varphi(i, j \mid f)$ which is:

$$\begin{aligned}\varphi(i, j \mid f) = (&\varphi(i, j \mid 0) - (2t-1)^2 + round(\frac{2tf}{k-1}))\%n(t) \\ &+ (2t-1)^2.\end{aligned} \tag{1}$$

in which

$$t = floor((\sqrt{\varphi(i, j \mid 0)} + 1)/2). \tag{2}$$

and

$$n(t) = \begin{cases} 1, & if \quad t = 0, \\ 8t, & otherwise. \end{cases} \tag{3}$$

And a rotational RoI pooling operation in the (i, j)-th bin ($0 \leq i, j \leq k-1$) is:

$$r_c(i, j \mid f, \Theta) = \sum_{(x,y) \in bin(i,j)} z_{\varphi(i,j \mid f),c}(x + x_0, y + y_0 \mid \Theta)/n. \tag{4}$$

We follow some definitions in [2] here. $r_c(i, j)$ is the response in the (i, j)-th bin for the c-th category, $z_{\varphi(i,j \mid f),c}$ is one of component feature maps, (x_0, y_0) is the coordinate of left-top corner of an RoI, $\lfloor i\frac{w}{k} \rfloor \leq x < \lceil (i+1)\frac{w}{k} \rceil$, $\lfloor j\frac{h}{k} \rfloor \leq y < \lceil (j+1)\frac{h}{k} \rceil$ (w is the width of an RoI and h is the height of an RoI) and n is the total number of pixels in the (i, j)-th bin.

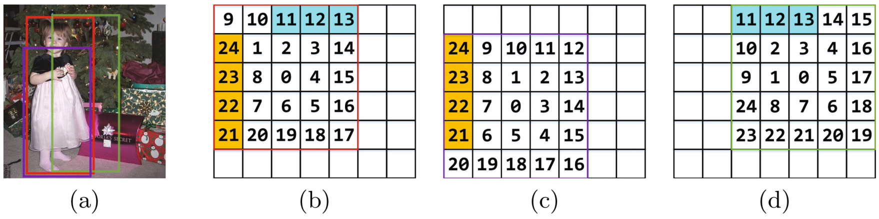
The $k \times k$ rotation-sensitive scores then give the final response of the RoI. In this paper we simply vote by averaging the scores, producing a $(C+1)$-dimensional vector for each RoI:

$$r_c(f, \Theta) = \sum_{i,j} r_c(i, j \mid f, \Theta)/(k \times k) \tag{5}$$

Here, we consider objects possess axial symmetry. Therefore, we do not take mirror operation and the rotation of mirror images into consideration. Note that in this paper, all the rotations are anticlockwise. We give the result of offset 6 when $k = 5$ (Fig. 4(b)).

3.2 Parallel Rotational Feature Extraction Modules

In RR-FCN, we choose the maximum of all rotational RoI pooling output as the final response. And we find that the feature extraction of all rotation angles on the same set of feature maps will cause translation variance fading away. Region proposals in different locations should get different responses. However, since we adopt rotational RoI pooling, the responses may be similar or partly same by rotating. Here are the analysis and solution.



(b)

9	10	11	12	13		
24	1	2	3	14		
23	8	0	4	15		
22	7	6	5	16		
21	20	19	18	17		

(c)

24	9	10	11	12		
23	8	1	2	13		
22	7	0	3	14		
21	6	5	4	15		
20	19	18	17	16		

(d)

		11	12	13	14	15
		10	2	3	4	16
		9	1	0	5	17
		24	8	7	6	18
		23	22	21	20	19

Fig. 5. (a) is the image to be detected with region proposals of red, blue and purple. (b) (c) and (d) are possible responses to the rotational RoI pooling of three region proposals, respectively. Here, $f_b = 0$, $f_c = 15$, $f_d = 2$. We mark out the same response in pure color.

Conflict with Translational Variance. Figure 5 is a simplified region-based detection situation with only three proposals. We can see if proposals are close to each other, they may get partly same outputs. And since we select the maximum response, the probability of this situation is high. So the total number of conflicts is larger even though every two proposals have a little overlap. Therefore, if we train a network with rotation information on the same feature maps, the output bins with different predicted results but same response value will conflict with themselves and lead to the unstable loss oscillation. However, if the difference of two offsets is a multiple of $(k-1)$, they will share no response. That means there are *up to four* kinds of rotational RoI pooling on a blank of component-sensitive feature maps. Thus we use rotational RoI pooling with discrete offset f to ensure that the difference is fixed. By this method, the translation variance can be kept in our network.

Parallel Rotational Feature Extraction Modules. As is analysed above, the available offsets are limited on a blank of component-sensitive feature maps. Thus we make the feature maps and rotational RoI pooling into a feature extraction module (shown in Fig. 2) which can be added in parallel easily. Each module has an initial offset f_i $(0 \leq i < k)$. Because the difference of every two offsets should be a multiple of $(k-1)$, the offsets of four rotational RoI pooling layers are f_i, $f_i + (k-1)$, $f_i + 2(k-1)$ and $f_i + 3(k-1)$. Therefore, there are four rotation-sensitive outputs in a module, and we select the maximum of these four as the ouput of this module. The four RoI pooling layers here share feature maps and thus come free of cost which is quite different from maxout [6]. In RR-FCN, we have several feature extraction modules in parallel with different initial f_i to cover more rotational situations and select the max response as the output, too. Therefore, the final rotation-sensitive response in RR-RCN is:

$$r_c(\Theta) = \max_{f} \; r_c(f, \Theta) \tag{6}$$

Then we use softmax function to normalize the responses across categories.

Interceptive Back Propagation. As we will introduce in Sect. 3.3, we train our networks with only one offset (equals to 0) first and with rotational feature extraction modules in the next step. In order to make our feature maps component-sensitive, we make changes to back propagation in extraction module. We make the network focus on rare situations rather than ones. Table 1 shows the back propagation selections in feature extraction modules. When a region proposal is an object (e.g a dog) and it chooses offset 0 in extraction modules, RR-FCN will ignore this back propagation. In other cases, they run as usual.

Table 1. The back propagation selections in feature extraction module. Note that objects here are class-aware.

	Class-aware object (yes)	Class-aware object (no)
Offset (0)	×	✓
Offset (otherwise)	✓	✓

3.3 Rotational Region-Based Fully Convolutional Networks

Architecture. We take ResNet-101 as our backbone. Since RR-FCN shares the same physical significance with R-FCN [2] when offset f equals to 0, we refer to its structure and make our transformation (we choose the python version code py-R-FCN[1]). As shown in Fig. 2, we use parallel rotational feature extraction modules to get rotation-sensitive responses. However, the bounding box regression method is different. Because bounding boxes are more relative to features without correcting rotation, we use position-sensitive RoI pooling [2] for bounding boxes.

[1] https://github.com/YuwenXiong/py-R-FCN.

Training. As is in [5], we define a multi-task loss function on each RoI as $\mathcal{L} = \mathcal{L}_{cls} + \mathcal{L}_{reg}$. $\mathcal{L}_{cls}$ is the cross-entropy loss and $\mathcal{L}_{cls}$ is the smooth L1 loss defined in [5]. Firstly, we set the hyper-parameters the same as [2] and use the model pre-trained on ImageNet [16] to train our network with only a single rotational RoI pooling layer whose offset f equals to 0 for 110k iterations. Then we train RR-FCN with rotational feature extraction modules based on the model we get previously. If there are more than one rotational feature extraction module, we make convolution parameters of their component-sensitive feature maps all initialized from the previous model. We use a learning rate of 0.0001, a weight decay of 0.0005 and a momentum of 0.9 to train our model 20k iterations. It is worth mentioning, because we do not know which proposal is rotated and every proposal is important for our network equally, we do not use online hard example mining (OHEM) [17]. Instead of OHEM, we use the interceptive back propagation (Sect. 3.2).

4 Experiments

4.1 Detection Results

We evaluate our methods on PASCAL VOC datasets, and the results are shown in Table 2. We give the detection results of R-FCN [2] model we used in the first step. Besides, we conduct two experiments with different number of rotation offsets (i.e. different number of parallel rotational feature extraction modules). We note the RR-FCN with one rotational feature extraction module as RR-FCN$_4$ (initial offset: 0) and the RR-FCN with two modules as RR-FCN$_8$ (initial offsets: 0 and 1).

As is shown in Table 2, our method achieves state-of-the-art accuracy. And more importantly, feature maps in RR-FCN become component-sensitive. It makes our networks figure out objects' structures and detect successfully under special circumstances (e.g., rotation).

Table 2. Comparisons on PASCAL VOC 2007 and 2012 using ResNet-101. Timing is evaluated on a single NVIDIA TITAN XP, 300 RoIs per image. †: http://host.robots.ox.ac.uk:8080/anonymous/EAVGYV.html ‡: http://host.robots.ox.ac.uk:8080/anonymous/LU6RUU.html

	mAP(%) (VOC07)	test time (sec/img) (VOC07)	mAP(%) (VOC12)	test time (sec/img) (VOC12)
R-FCN	78.83	0.093	74.50	0.097
RR-FCN$_4$	77.75	0.093	73.31†	0.097
RR-FCN$_8$	77.26	0.102	73.12‡	0.102

4.2 Analysis

Robustness Analysis. Since we construct several blanks of component-sensitive maps in RR-FCN, our model is more robust in real world. Instead of matching directly, RR-FCN can adjust the feature extraction method according to the responses and give more robust predictions.

In order to better illustrate the robustness of the RR-FCN, we detect on the freestyle motorbikes [4] test set[2]. It contains a hundred pictures which contain 128 motorbikes with planar rotations. The test results are shown in Table 3. The RR-FCN$_4$ improves AP by 13.7% which shows that the RR-FCN has better stability under abnormal conditions. Here we give some of detection results (Fig. 6). In these pictures, objects are not in their usual position.

Table 3. Test results on the freestyle test set. All of these networks here are still trained on PASCAL VOC 2007 and 2012.

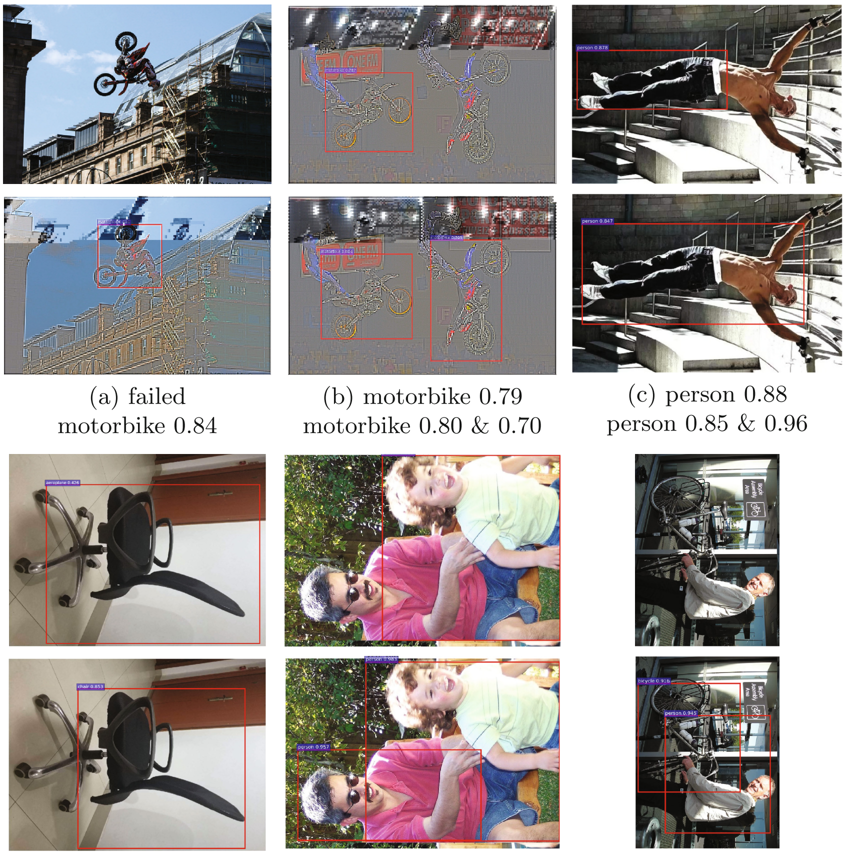
	R-FCN	RR-FCN$_4$	RR-FCN$_8$
AP(%)	62.4	76.1	74.4

As we can see in Fig. 6, RR-FCN is more accurate under rotation operations. Even these postures are not usual in training set, RR-FCN can still deal it well. That means our network is more robust and it can *comprehend* the object structure.

Feasibility Analysis. RR-FCN learns rotation information from dataset without extra supervision information. We further analyse the convergence of our model. We introduce the ability of RR-FCN to detect objects in rare postures in previous section. The "*rare*" is opposite to "*normal*". For example, we are used to seeing people standing, thus we know what handstand looks like even though we rarely see it. A more mathematical statement is: If I rotate myself x degrees and find a "*normal*" object, in fact, the object is in a "*rare*" posture by rotating itself x degrees. In our experiment, we make our RR-FCN learn what is "*normal*" on tens of thousands of images from PASCAL VOC datasets and then get the rotation information through component-sensitive feature map and rotational RoI pooling. Moreover, we develop interceptive back propagation which can stop our network tolerating rotations.

Necessity Analysis. In the test phase, we may use the previous networks to achieve similar detection results by rotating images. But we argue that it is not appropriate. Firstly, it costs several times of RR-FCN computation. Secondly, because we do not know the rotation angles of the images, the data preprocessing must be unpersuasive. More seriously, there might be wrong results when

[2] http://www.iri.upc.edu/people/mvillami/files/iri_freestyle_motocross_dataset_v1.1.zip#opennewwindow.


(a) failed
motorbike 0.84

(b) motorbike 0.79
motorbike 0.80 & 0.70

(c) person 0.88
person 0.85 & 0.96

(d) aeroplane 0.43
chair 0.85

(e) person 0.98
person 0.85

(f) failed
person 0.9 & bike 0.92

Fig. 6. Some detection results for explaining the robustness of the proposed model. We choose objects in unusual positions and angles to detect. In each group of pictures, the top is the detection result of R-FCN, and the bottom is the detection result of RR-FCN. (a) and (b) are come from freestyle motorbike dataset, (c) and (d) are shot normally, (e) and (f) are come from PASCAL VOC but rotated. We can find RR-FCN shows advantages in detecting objects without the usual position or angle assumption.

detecting rotated images, so we cannot merge the detection results coming from rotated images ideally.

We further analyse the necessity of training. RR-FCN guides the training according to the responses of feature maps. The basic assumptions of our experiments is that the network has been trained well with offset 0 and the most objects appear with usual postures. Under these two assumptions, the trained feature maps in the first step perhaps can detect objects in rare postures with rotational RoI pooling layers. We construct the network with this idea, however, the experiments show that the results are not good. We also test the accuracy of the network (Table 4) and the result is not good, either.

Because the large capacity of deep networks, even the rare objects may have large response in normal pooling method. So we must train the network to make the feature maps component-sensitive.

Table 4. Detection results on PASCAL VOC 2007 using ResNet-101. RR-FCN$_1$ is the network trained only with offset equals to 0, RR-FCN$_{4test}$ is still the network RR-FCN$_1$ but using 4 offsets (0, 6, 12, 18) to test it.

	RR-FCN$_1$	RR-FCN$_{4test}$
mAP(%)	78.83	70.99

5 Conclusion and Future Work

This paper presents a Rotational Region-based Fully Convolutional Network, which is a robust detection network incorporating rotational invariance. We develop a novel pooling method which can share rotational information across different feature maps. To train our networks with the purpose, we propose interceptive back propagation. Moreover, we figure out the specific requests for constructing fully convolution detection networks. In this way, our RR-FCN can learn rotational invariance of objects and detect well even under the situations rarely appear in dataset. That means, It will work better in real world.

We will specify the rotation angle to guide detection and other computer vision task in our future work.

References

1. Boureau, Y., Ponce, J., Lecun, Y.: A theoretical analysis of feature pooling in visual recognition, pp. 111–118 (2010)
2. Dai, J., Li, Y., He, K., Sun, J.: R-fcn: object detection via region-based fully convolutional networks. In: Advances in Neural Information Processing Systems, pp. 379–387 (2016)
3. Dai, J., et al.: Deformable convolutional networks (2017)
4. Fergus, R., Perona, P., Zisserman, A.: Object class recognition by unsupervised scale-invariant learning. In: 2003 Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, pp. II-264–II-271 (2003)
5. Girshick, R.: Fast r-cnn. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1440–1448 (2015)
6. Goodfellow, I.J., Warde-Farley, D., Mirza, M., Courville, A., Bengio, Y.: Maxout networks. arXiv preprint arXiv:1302.4389 (2013)
7. He, K., Zhang, X., Ren, S., Sun, J.: Spatial pyramid pooling in deep convolutional networks for visual recognition. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8691, pp. 346–361. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10578-9_23

8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
9. Jaderberg, M., Simonyan, K., Zisserman, A., Kavukcuoglu, K.: Spatial transformer networks. In: Neural Information Processing Systems, pp. 2017–2025 (2015)
10. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. arXiv preprint arXiv:1612.03144 (2016)
11. Liu, W., et al.: SSD: single shot multibox detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9905, pp. 21–37. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_2
12. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Computer Vision and Pattern Recognition, pp. 3431–3440 (2015)
13. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 779–788 (2016)
14. Redmon, J., Farhadi, A.: Yolo9000: better, faster, stronger. arXiv preprint arXiv:1612.08242 (2016)
15. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems, pp. 91–99 (2015)
16. Russakovsky, O., et al.: Imagenet large scale visual recognition challenge. Int. J. Comput. Vis. **115**(3), 211–252 (2015)
17. Shrivastava, A., Gupta, A., Girshick, R.: Training region-based object detectors with online hard example mining. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 761–769 (2016)