Revisiting Faster R-CNN: A Deeper Look at Region Proposal Network

Guangxing Han, Xuan Zhang(✉), and Chongrong Li

Tsinghua National Laboratory for Information Science and Technology (TNList),
Institute for Network Sciences and Cyberspace (INSC),
Tsinghua University, Beijing 100084, China
hgx14@mails.tsinghua.edu.cn, {zhangx,licr}@cernet.edu.cn

Abstract. Currently, state-of-the-art object detectors are based on Faster R-CNN. We firstly revisit Faster R-CNN and explore problems in it, e.g., coarseness of feature maps for accurate localization, fixed-window feature extraction in RPN and insensitivity for small scale objects. Then a novel object detection network is proposed to address these problems. Specifically, we utilize a two-stage cascade multi-scale proposal generation network to get high accurate proposals: an original RPN is adopted to initially generate coarse proposals, then another network with multi-layer features and RoI pooling layer are introduced to refine these proposals. We also generate small scale proposals in the second stage simultaneously. After that, a detection network with multi-layer features further classifies and refines proposals. A novel 3-step joint training algorithm is introduced to optimize our model. Experiments on PASCAL VOC 2007 and 2012 demonstrate the effectiveness of our network.

Keywords: Faster R-CNN · General object detection · Multi-scale object proposal · Multi-layer feature aggregation

1 Introduction

Object detection is one of the most fundamental problems in computer vision. Currently top leading results on PASCAL VOC [1], MS COCO [2] object detection challenges all utilize Faster R-CNN [3] framework. The pioneering R-CNN [4] decomposes object detection into two primary tasks. Firstly thousands of candidate object locations are generated by traditional object proposal methods (e.g., Selective Search [5]). Then these candidates are classified and further refined by deep convolutional neural network (DCNN [6,7]). Based on R-CNN, Faster R-CNN introduces region proposal network (RPN [3]) to generate high quality proposals and adopts Fast R-CNN [8] to perform RoI-wise classification and refinement. Both shared convolutional layers and end-to-end joint training push object detection to real-time and state-of-art accuracy.

In Faster R-CNN, RPN is built on top of high-level convolutional feature layer (e.g., layer "conv5_3" in VGG-16 [7]) which is shared with Fast R-CNN.

© Springer International Publishing AG 2017
D. Liu et al. (Eds.): ICONIP 2017, Part III, LNCS 10636, pp. 14–24, 2017.
https://doi.org/10.1007/978-3-319-70090-8_2

Specifically, a small network is slided over the feature layer. Each sliding window takes as input an 3×3 spatial window of the convolutional feature maps. Then it is mapped to a lower-dimensional feature (e.g., 512-d for VGG-16). After that, this feature is fed into two sibling fully-connected layers: a box-regression layer (reg) and a box-classification layer (cls) to generate object proposals. Using multiple reg and cls layers, RPN can simultaneously predict multiple object proposals centered at each sliding window with different scales and aspect ratios. Then top scored proposals are selected. Finally, Fast R-CNN, also built on top of high-level convolutional features (e.g., "conv5_3" in VGG16), are followed to further classify and refine the proposals.

However, three issues emerge in such an approach. Firstly, as we know, higher layer convolutional features capture high-level semantic knowledge. Lower layer features, on the other hand, contain rich low-level visual information. [3] only focuses on high-level features, which is too coarse for accurate object detection. Secondly, multiple proposals produced by one sliding window in RPN share the same features which is calculated by the fixed 3×3 spatial window of the DCNN feature. However, these proposals differ much in scales and aspect ratios. We expect RoI pooling features would improve localization accuracy. Thirdly, the receptive field of higher layers is relatively large, and therefore not suitable for small scale object detection. For example in VGG-16, the receptive fields of layer "conv3_3", "conv4_3" and "conv5_3" are 40, 92 and 192 respectively [9]. Higher layers with large receptive field are more suitable for large object detection. In contrast, low-level layers such as "conv4_3" are better matched to small scale object. We can detect different scale objects on different layers.

For the first problem, [10,11] adopt skip-layer connections to directly aggregate multi-layer convolutional features. We follow this with a top-down feature aggregation. Then we propose a novel parallel top-down cascade to refine proposals generated in higher layers. This is different from [12,13] which only use higher layer features for multi-stage refining. For the second problem, HyperNet [10] extracts features for candidate boxes by RoI pooling layer. But approximately 70% of the forward time is consumed by the proposal generation network. HyperNet-SP simplifies this network with detection accuracy loss. We expect careful cascade design can achieve better speed/accuracy trade-offs. As for the third problem, [14,15] employ RPNs on multiple layers to compute multi-scale proposals. Each layer focuses on objects within certain scale ranges. However, lower layers alone do not have strong semantic knowledge for detection [16]. We expect top-down feature aggregation would alleviate the problem.

To summarise, we propose an object detection network with region proposals generated in a top-down cascade manner. Instead of treating DCNNs as black-box feature extractors in [12,13], our approach explores the inherent hierarchies of DCNNs. Concretely, we first generate coarse object proposals on high-level convolutional layer using original RPN. Ablation experiments demonstrate excellent performance of RPN. Then another network, denoted as R-RPN, uses multi-layer features and RoI pooling layer to refine the top scored 2k proposals. Small scale object proposals are generated simultaneously in this stage. Using

the two-stage cascade multi-scale proposal generation network, we achieve high recall with only a few proposals. Meanwhile we also introduce a 3-step method to optimize the whole network. Our main contributions are in two folds:

- We propose a novel two-stage cascade multi-scale proposal generation network. It achieves 98.2% recall at IoU = 0.5 with 300 proposals on PASCAL VOC 2007 test which is comparable to HyperNet, and outperforms RPN by a large margin (needs 2k proposals). Moreover, our method is 12× faster than HyperNet on proposal generation due to cascade design.
- Using the 3-step joint training algorithm, we achieve 76.8% and 73.2% mAP on PASCAL VOC 2007 and 2012 test respectively. Moreover, our method runs at 3.3 fps on TITAN X GPU. Both accuracy and running speed are competitive compared to other object detection networks.

2 Object Detection with Two-Stage Cascade Multi-scale Proposal Generation

2.1 Basic Proposal Generation Networks

In this section, we investigate into three basic proposal generation networks RPN, R-RPN and FRCN. In RPN, fixed-window convolutional feature extraction is adopted for fast computation. In FRCN, RoI pooling and two successive fully connected layers (fc) is adopted to extract accurate features for proposals. While R-RPN is a compromise of FRCN and RPN. In R-RPN, the ultimate 256-d feature for each box is obtained through RoI pooling layer and a single fc layer. The difference between R-RPN and RPN lies in that feature is extracted using a fixed window or RoI pooling layer. While R-RPN differs with FRCN on whether or not using more discriminative but computationally expensive fc layers.

2.2 Our Approach

The detailed architecture of our object detection network is shown in Fig. 1. It mainly consists of three parts. Firstly, we generate hierarchical feature maps through convolutional layers. Then a two-stage proposal generation network is followed to produce high quality proposals. Finally, object proposals are further classified and refined by a detection network. In the rest of our paper we use conv3, conv4 and conv5 to represent for layer conv3_3, conv4_3 and conv5_3 respectively. Conv45 and conv345 are the combination of corresponding layers.

Hierarchical convolutional layers feedforward. We use VGG16 as our backbone network. Initially, we resize the input image such that its shorter side is 600 pixels and keep the aspect ratio. Then we feed forward the re-scaled image through hierarchical convolutional layers. Specifically, we mainly focus on features from layers conv3, conv4 and conv5.

Two-stage cascade multi-scale proposal generation. Multi-scale proposals are generated in a two-stage cascade manner. Firstly, we directly build a RPN

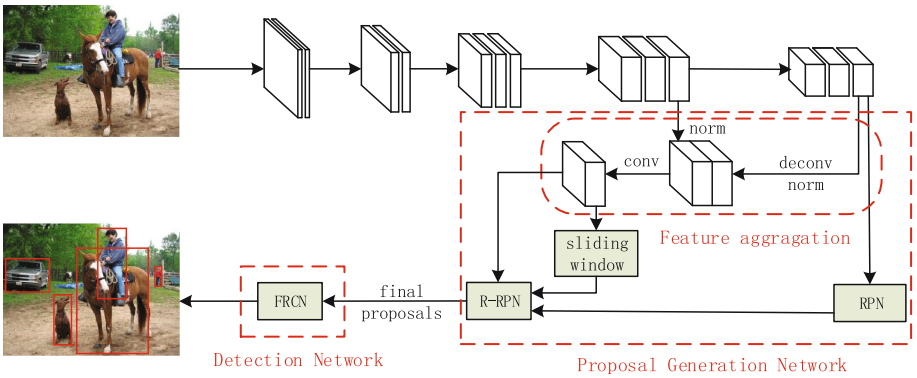

Fig. 1. Proposed object detection architecture. We firstly generate hierarchical feature maps which are shared by proposal generation network and detection network. Then a two-stage cascade multi-scale proposal generation network is introduced to produce proposals. Finally, proposals are further classified and refined by the detection network. Our model is jointly trained in a 3-step manner.

module on top of layer conv5. Roughly 20k anchors are generated as in [3]. We also adopt NMS with threshold 0.7 to suppress most of the redundancy. After this, top 2k ranked proposals are selected and they achieve very high recall, e.g., approximately 98% at IoU = 0.5 on PASCAL VOC 2007 test dataset.

Then a R-RPN module follows. In R-RPN, we combine features from layer conv4 and conv5 to serve as the input feature maps. To concatenate multi-layer features, we firstly use a deconvolutional layer upsampling conv5 to achieve the same resolution with conv4, then L2-normalise each layer per spatial location, re-scale it with a learnable "scale layer" initialized to 20 and concat them together. At last, we reduce the dimension of combined feature to 256 channels with 1×1 convolutional layer. The input boxes can be divided into two parts. One comes from the generated 2k proposals in RPN, the other is the small scale boxes generated by sliding window on conv45 resolution (stride: 2) with single scale of 64^2 pixels and aspect ratio 1:1. We generate small scale object proposals on conv45 rather than conv5 to better fit the small object size and utilize fine-grained features. Only one aspect ratio is adopted for no obvious drop in recall when it comes to small scale object. Finally, top 300 proposals are selected after the same NMS operation in RPN before the final detection network.

Final object detection. The final detection network employs a core FRCN module. The input RoIs come from the proposal generation network. Input feature maps are derived from layer conv3, conv4 and conv5. Different from the multi-layer combination strategy used in R-RPN, we combine multi-layer features for each proposal individually. Firstly, we extract fixed feature descriptors from multiple layers simultaneously for each proposal, then perform L2-normalization, concatenation, and feature reduction in turn to produce fixed representation of size $7 \times 7 \times 512$. Different from method in [17], we again

L2-normalise the feature per spatial location and re-scale (learnable "scale layer" initialized to 130) it back to match the activation amplitudes in VGG16 fc6 layer. After obtaining final feature for each RoI, we conduct accurate RoI-wise classification and refinement. At the end of detection, we adopt class specific NMS with threshold 0.3 to suppress redundancy.

2.3 3-Step Joint Training

Unlike [3,10] use 4-step alternating training mechanism, we introduce a carefully designed 3-step training algorithm. Firstly we train the proposal generation network and detection network separately. Then combining two sub-networks, we optimize the whole pipeline end-to-end.

In the first step, we train the two-stage proposal generation network alone. As for RPN, We mainly follow the hyper-parameters in [3]. For R-RPN, we firstly generate boxes by sliding window on conv4 resolution (stride: 2) with 4 scales (64^2, 128^2, 256^2 and 512^2 pixels) and 3 aspect ratios (1:1, 1:2 and 2:1). The positive and negative labels are determined as in [3]. Moreover, we keep the ratios of positive and negative boxes at most 1:2. This small trick both improves the training speed and model accuracy. Moreover, unsampled boxes are neglected in R-RPN while RPN still forwards them due to its implementation by convolution. We jointly optimize two tasks with the loss function defined as:

$$L = l(c_1, c_1^*, b_1, b_1^*) + l(c_2, c_2^*, b_2, b_2^*) \tag{1}$$

$$l(c, c^*, b, b^*) = L_{cls}(c, c^*) + \lambda L_{reg}(b, b^*) \tag{2}$$

where c_1 and c_2 are predicted labels for RPN and R-RPN, c_1^* and c_2^* are corresponding true labels. Similarly, b_1, b_2 and b_1^*, b_2^* are predicted and true boxes for RPN and R-RPN respectively. The L_{cls} and L_{reg} losses are calculated as in [8]. Our model is initialized by VGG16 model. Newly added layers adopt "Xavier" initialization. By default $\lambda = 1$. We train the network with learning rate 10^{-3} on the first 50k iterations and 10^{-4} for the next 30k iterations.

Then in the second step, we train the FRCN together with a simple RPN. Specifically, we adopt the approximate joint training and mainly follow the hyper-parameters in [3]. The loss function is defined as:

$$L = l(c_1, c_1^*, b_1, b_1^*) + l(c_3, c_3^*, b_3, b_3^*) \tag{3}$$

where c_3 and c_3^* are predicted and true labels for FRCN, b_3 and b_3^* are predicted and true boxes for FRCN.

Thus so far proposal generation network and detection network are separately trained. In the third step, we jointly train two networks with shared convolutional layers. Firstly, parameters of the shared convolutional layers and FRCN are inherited from the trained model in step two, while parameters in RPN and R-RPN comes from the model in step one. Different from step one, the input boxes of R-RPN come from top 2k proposals in RPN together with ∼2k small

scale proposals generated by sliding window. We optimize the whole pipeline with approximate joint training and the loss function is defined as:

$$L = l(c_1, c_1^*, b_1, b_1^*) + l(c_2, c_2^*, b_2, b_2^*) + l(c_3, c_3^*, b_3, b_3^*) \quad (4)$$

We use relatively low learning rate in this step, 10^{-4} for the first 40k iterations, then 10^{-5} and 10^{-6} for the next 40k and 20k iterations.

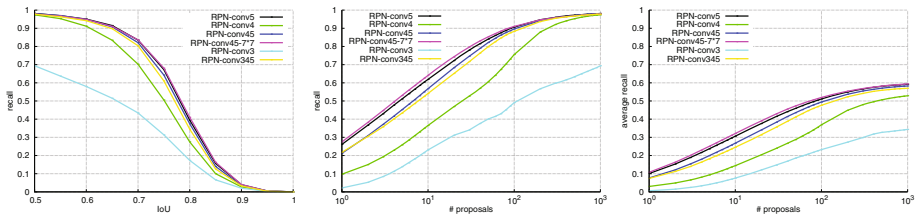

Fig. 2. RPN evaluation on PASCAL VOC 2007 test. **Left**: Recall versus IoU threshold with 1000 proposals. **Middle**: Recall versus number of proposals at IoU = 0.5. **Right**: Average Recall versus number of proposals. "7 * 7": Spatial window size for extracting features in RPN.

3 Experimental Results

3.1 Ablation Experiments

We conduct several ablation experiments to evaluate our design on PASCAL VOC 2007 test. The training set combines VOC 2007 trainval and VOC 2012 trainval dataset. There are about 16.5k trainval images and 5k test images over 20 categories. By default, all networks share the same VGG16 backbone network.

Is RPN with conv5 Feature Good Enough?
We evaluate RPN on a variety of convolutional layers including multi-layer combination. We mainly follow the hyper-parameters in [3]. By default, we use 3×3 spatial window to extract feature maps for each anchor.

The evaluation result is illustrated in Fig. 2. It shows that RPN with conv5 feature maps outperforms methods either using conv4 or conv3 feature maps by a large margin. When using multi-layer features such as conv45 and conv345, the performance significantly improves with the help of conv5, thus indicating that conv5 contains more semantics which is suit for proposal generation. However the combined features are still inferior to conv5 feature alone. The reason is that on conv3 or conv4 resolution, a 3×3 spatial window only has an receptive field of 48 and 108 respectively. It may not generate the exact feature description covering most of the object. If we enlarge the feature window to 7×7 on conv4 resolution, we achieve slightly better proposal recall than conv5 due to the help of fine-grained details of lower layers. But the improvement is minor considering

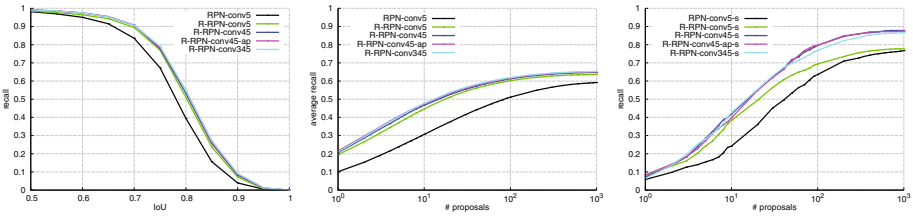

Fig. 3. R-RPN evaluation on PASCAL VOC 2007 test. **Left**: Recall versus IoU threshold with 1000 proposals. **Middle**: Average Recall versus number of proposals. **Right**: Small scale object ($<32^2$ pixels) recall versus number of proposals at IoU = 0.5. "ap": Multi-layer feature combination after pooling. "s": Small scale object proposals.

additional time consuming. We also notice that 2k proposals for RPN on conv5 already achieve 98.5% recall at IoU = 0.5. So it is pretty efficient for RPN on conv5 alone to generate coarse region proposals in the first-stage.

How Much Does the 2nd Stage Proposal Network Help?
We comprehensively evaluate the design choices of the 2nd stage proposal generation network. In practice, we use the training scheme of the first step in Sect. 2.3. By default, multi-layer features are concatenated before R-RPN.

The evaluation result is illustrated in Fig. 3. We can find that all two-stage cascade methods significantly outperform RPN alone. RPN gets 59.2% average recall (AR) with 1000 proposals, while RPN cascaded with R-RPN on conv5, conv45 and conv345 achieve 63.6%, 64.7% and 65.3% AR respectively. For higher IoU (e.g., 0.8), cascaded methods outperform RPN by more than 10 points. This indicates that RoI pooling extracts fine-grained feature which obtains more accurate object localization.

However, it seems R-RPN on conv5, conv45 or conv345 achieves similar performance. Now let's take a deeper look at small scale object (area $< 32^2$ pixels) in Fig. 3. For small scale object, R-RPN on conv5 only gets 77.7% recall with 1000 proposals at IoU = 0.5, which is similar to RPN alone. But R-RPN on conv45 achieves 87.8% recall, similar to conv345, which is substantially higher than conv5. This implies that features on conv4/conv3 resolution are more suitable for small scale objects due to its suitable receptive field and large resolution feature maps. In addition, R-RPN on conv45 is more competitive compared to conv345 for less time-consuming and similar performance.

Furthermore, we also compare different feature combination strategies. We explore on whether to combine multi-layer features before RoI pooling or individually for each box. Results in Fig. 3 indicate that the two methods achieve almost the same performance. However in practice, the later method consumes nearly twice as much time as the former. The reason is that R-RPN needs to process ∼4k boxes and a large amount of boxes are highly overlapped. If we manipulate each box individually, large computation resources will be wasted. So we directly combine conv4 and conv5 before RoI pooling in R-RPN.

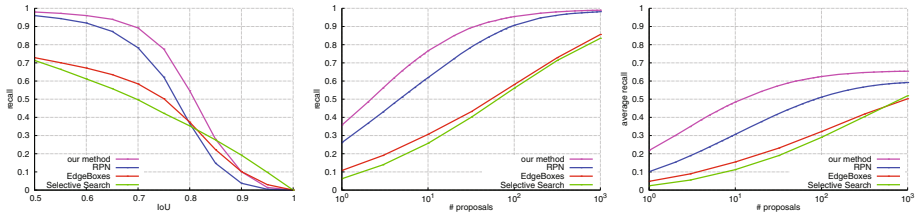

Fig. 4. Proposal evaluation on PASCAL VOC 2007 test. **Left**: Recall versus IoU threshold with 300 proposals. **Middle**: Recall versus number of proposals at IoU=0.5. **Right**: Average Recall versus number of proposals.

Table 1. Object detection mAP on VOC 2007 and 2012 for different features.

Feature	mAP on VOC 2007	mAP on VOC 2012
conv5	75.7	72.2
conv45	76.5	72.9
conv345	**76.8**	**73.2**

How Important Are Multi-layer Features?
We evaluate the effect of multi-layer features on detection network. By default, proposals come from two-stage proposal generation network and multi-layer features are combined individually for each proposal with skip-pooling.

The evaluation result is shown in Table 1. We achieve similar results as [17]. Features with conv345 achieve the best result both in VOC2007 and VOC2012 test dataset. We adopt skip-pooling here for two reasons. Firstly, normalization just before layer fc6 match the activation amplitudes of the original VGG16 fc6 layer, which helps for training. Secondly, only few proposals (e.g., 300 proposals) are evaluated in the final detection network. In practice we observe lower time consumption compared to layer combination before RoI pooling.

3.2 Proposal Evaluation

In this section, we evaluate the proposal generation network on PASCAL VOC 2007 test dataset. The training data consists of VOC 2007 trainval and VOC 2012 trainval. Evaluation result is shown in Fig. 4. Using only 300 proposals, our method achieves 64.7% AR, outperforming RPN by 5.5 points and also 25 points higher than Selective Search as well as EdgeBoxes. To compare with HyperNet and DeepProposal, [10] shows that HyperNet and HyperNet-SP needs 20 and 30 proposals respectively to achieve 75% recall at IoU = 0.7. DeepProposal [18] needs 540 proposals. Our method requires 25 proposals. However, HyperNet takes nearly 810 ms for proposal generation owing to the enormous number of input RoIs, while our method only costs 70 ms to achieve comparable results due to the efficient cascade design.

In addition, we observe that our proposal network also benefits a lot from multi-task joint training of proposal generation network and detection network. For example, before joint training, our proposal network needs 1000 proposals to achieve 64.7% AR, while it only needs 300 proposals to achieve similar results after joint training.

3.3 Object Detection Evaluation

We comprehensively compare our method with other VGG16 based object detection networks on PASCAL VOC 2007 and 2012 test dataset.

Table 2. PASCAL VOC 2007 & 2012 test detection results. For VOC2007 test, the training data is 2007trainval + 2012trainval. For VOC2012 test, the training data is 2007trainvaltest + 2012trainval. "#100": 100 proposals. "#300": 300 proposals.

Approach	FPS	mAP on VOC 2007	mAP on VOC 2012
Faster R-CNN	7	73.2	70.4
CRAFT	-	75.7	71.3
HyperNet-SP	5	74.8	71.3
HyperNet	0.9	76.3	71.4
Ours-#100	4	76.3	72.3[a]
Ours-#300	3.3	**76.8**	**73.2**[b]

[a] http://host.robots.ox.ac.uk:8080/anonymous/NMMVBD.html.
[b] http://host.robots.ox.ac.uk:8080/anonymous/ZMWN3P.html

PASCAL VOC 2007 results. The comparison result is shown in Table 2. Using 100 proposals, our method achieves 76.3% mean average precision (mAP) on VOC 2007 similar to HyperNet. While using 300 proposals, our method achieves 76.8% mAP, 0.5 points higher than HyperNet and 2 points higher than HyperNet-SP. Moreover, our method has a frame rate of 3.3 fps which is comparable with HyperNet-SP and 3.7× faster than HyperNet, which demonstrates the effectiveness of our cascade design. Our method also achieves 1.1 points higher than CRAFT, which proves the effectiveness of the top-down cascade multi-scale proposal generation and our 3-step joint training. Moreover, our method is especially good at small scale object detection. For example our method achieves 66.9% mAP on "bottle", which is 15 points higher than Faster R-CNN and 4.5 points higher than HyperNet.

PASCAL VOC 2012 results. We observe similar results on PASCAL VOC 2012 test dataset in Table 2. Using 100 proposals, our method achieves 72.3% mAP on VOC 2012 test, 0.9 points higher than HyperNet and CRAFT. Using 300 proposals, our method achieves 73.2% mAP, 1.8 points higher than HyperNet and CRAFT. We still outperforms other methods significantly on small scale objects such as "bottle" and "plant".

Our implementation adopt the publicly available code of [3] in python version. All of our networks are trained and tested on a single TITAN X GPU.

4 Conclusion

We propose an object detection network with a novel proposal generation subnetwork attempting to solve problems in Faster R-CNN. Two-stage cascade multi-scale proposal generation network is carefully designed for fast and accurate proposals generation. Specifically, initial multi-scale proposals are generated in RPN, then R-RPN is adopted to further refine those proposals. We also conduct several ablation experiments to evaluate our design. Experiments on PASCAL VOC 2007 and 2012 demonstrate the effectiveness of our method both on speed and accuracy compared to other object detection networks.

References

1. Everingham, M., Eslami, S.A., Van Gool, L., et al.: The Pascal visual object classes challenge: a retrospective. Int. J. Comput. Vis. **111**, 98–136 (2015). LNCS. Springer
2. Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common Objects in Context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 740–755. Springer, Cham (2014). doi:10.1007/978-3-319-10602-1_48
3. Ren, S., He, K., Girshick, R., et al.: Faster R-CNN: towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems 28, pp. 91–99. Curran Associates, Montréal (2015)
4. Girshick, R., Donahue, J., Darrell, T., et al.: Region-based convolutional networks for accurate object detection and segmentation. In: IEEE Computer Vision and Pattern Recognition, pp. 580–587. IEEE Press, Columbus (2014)
5. Uijlings, J.R., Van De Sande, K.E., Gevers, T., et al.: Selective search for object recognition. Int. J. Comput. Vis. **104**, 154–171 (2013)
6. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems 25, pp. 1097–1105. Curran Associates, South Lake Tahoe (2012)
7. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
8. Girshick, R.: Fast R-CNN. In: IEEE International Conference on Computer Vision, pp. 1440–1448. IEEE Press, Santiago (2015)
9. Yu, W., Yang, K., Bai, Y., et al.: Visualizing and comparing convolutional neural networks. arXiv preprint arXiv:1412.6631 (2014)
10. Kong, T., Yao, A., Chen, Y., et al.: HyperNet: towards accurate region proposal generation and joint object detection. In: IEEE Computer Vision and Pattern Recognition, pp. 845–853. IEEE Press, Las Vegas (2016)
11. Zhang, L., Lin, L., Liang, X., He, K.: Is Faster R-CNN doing well for pedestrian detection? In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9906, pp. 443–457. Springer, Cham (2016). doi:10.1007/978-3-319-46475-6_28
12. Yang, B., Yan, J., Lei, Z., et al.: Craft objects from images. In: IEEE Computer Vision and Pattern Recognition, pp. 6043–6051. IEEE Press, Las Vegas (2016)

13. Gidaris, S., Komodakis, N.: Attend refine repeat: active box proposal generation via in-out localization. arXiv preprint arXiv:1606.04446 (2016)
14. Cai, Z., Fan, Q., Feris, R.S., Vasconcelos, N.: A unified multi-scale deep convolutional neural network for fast object detection. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9908, pp. 354–370. Springer, Cham (2016). doi:10.1007/978-3-319-46493-0_22
15. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., Berg, A.C.: 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). doi:10.1007/978-3-319-46448-0_2
16. Lin, T.Y., Dollár, P., Girshick, R., et al.: Feature pyramid networks for object detection. arXiv preprint arXiv:1612.03144 (2016)
17. Bell, S., Lawrence Zitnick, C., Bala, K., et al.: Inside-outside net: detecting objects in context with skip pooling and recurrent neural networks. In: IEEE Computer Vision and Pattern Recognition, pp. 2874–2883. IEEE Press, Las Vegas (2016)
18. Ghodrati, A., Diba, A., Pedersoli, M., et al.: DeepProposal: hunting objects by cascading deep convolutional layers. In: IEEE International Conference on Computer Vision, pp. 2578–2586. IEEE Press, Santiago (2015)

http://www.springer.com/978-3-319-70089-2

Neural Information Processing
24th International Conference, ICONIP 2017,
Guangzhou, China, November 14-18, 2017,
Proceedings, Part III
Liu, D.; Xie, S.; Li, Y.; Zhao, D.; El-Alfy, E.-S.M. (Eds.)
2017, XVIII, 938 p. 405 illus., Softcover
ISBN: 978-3-319-70089-2