BOUNDARY OBJECTNESS NETWORK FOR OBJECT DETECTION AND LOCALIZATION

Juan Wang[1], *Xiaoming Tao*[*1], *Mai Xu*[2], *Jianhua Lu*[1]

[1]Tsinghua National Laboratory for Information Science and Technology (TNList), Beijing, China
[2]School of Electronics and Information Engineering, Beihang University, Beijing, China
Email: taoxm@tsinghua.edu.cn

ABSTRACT

In this paper, we present the boundary objectness network (BON), an effective convolutional neural network (CNN) for object detection. Its core contribution is to accurately localize the objects. Generally, the CNN-based localizers predict four bounding box coordinates by learning a regression function. This method shows a low Intersection-of-Union (IoU) with the ground truth box. In our work, the localization is formulated as a probabilistic problem. Specifically, the deep features inside the candidate proposal are mapped into a row and a column feature vector, which are called boundary objectness. The boundary objectness indicates the existence of an object in the horizontal and vertical direction of the proposal, enabling us to elaborately localize the object. Moreover, the modules of object detection share the common convolutional layers. Meanwhile, a multi-task loss function is designed for joint training strategy. Experimental results on the PASCAL VOC datasets demonstrate the competitive performance of our method. For the VGG16 model, we achieve 77.6 % mAP at a speed of 4 frame per second (FPS), thus having the potential for real-time processing.

Index Terms— convolutional neural network, object detection, localization, boundary objectness.

1. INTRODUCTION

Object detection is a computer vision task that has attracted an immense amount of attention over the last years. Compared with the recognition task, object detection requires not only accurate classification but also precise localization. Efficiency is another important key to determine the performance of object detection, making it practicable in real-time applications.

Over the past decade, the dominant approach to localizing the objects is the sliding window paradigm, based on HOG templates and SVM classifiers [1] [2]. However, sliding windows makes an exhaustive search, which is computationally infeasible. Several works attempt to use segmentation for localization [3] [4]. These methods generate a set of foreground/background segmentations, learn to predict the likelihood that a foreground segment is a complete object, and use this to rank the segmentations. However, these methods are time-consuming. EdgeBoxes [5], simply measuring the number of edges, provides the tradeoff between localization accuracy and efficiency, which is desired in real-time applications.

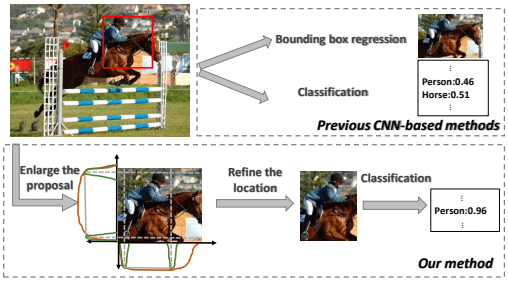


Fig. 1. The comparison of previous CNN-based object detectors and ours.

Important progress for improving both the accuracy and efficiency of object detectors has remarkably advanced with convolutional neural networks (CNNs) [6]. The region-based CNN (R-CNN) [7] and Fast R-CNN [8] are two representative works. They extract candidate proposals through selective search [9] and then apply CNNs to classify these proposals. However, the classification module using CNNs generally takes advantages of GPU, while the localization module is implemented on the CPU. The separate modules make the process inefficiency. To address this problem, Faster-RCNN [10] constructs a region proposal network (RPN) on top of the last convolutional layer of Fast-RCNN, outputing an objectness score and k regressed proposals defined by four coordinates. RPN shares the convolutional layers with Fast R-CNN, thus enabling high-efficiency. However, Faster-RCNN struggles with unsatisfactory mAP (mean Average Precision) with high IoU (Intersection-of-Union) thresholds. To solve this problem, some literature hypothesizes that it is attributed to the coarseness of single deep layer features. Therefore, SSD [11] aims to detect objects on multiple layers of CNN. HyperNet [12] combines deep and coarse information with shallow and fine information into a feature cube. Both two methods are able to improve the performance of RCNN.

In addition to the insufficient features, we consider the localization method as another primary cause. The above CNN-based object detectors apply the bounding box regression for

This work was supported by the National Natural Science Foundation of China (NSFC, 61622110, 61471220, 91538107).

localization. However, this regression method often shows a deviation to the ground truth bounding box, further leading to the failure of classifier, as shown in Fig.1. In our work, we localize the objects by enlarging the candidate proposal, measuring the feature vectors produced from CNN and refining the object location.

Our main contributions are summarized as follows:

(1) We propose a category-agnostic localizer based on the recent advance LocNet [13], which follows a general object detector to refine the category-specific proposals at post-processing stage.

(2) We develop a complete object detection framework, called boundary objectness network (BON), in which all the modules share the convolutional layers.

(3) We introduce a multi-task loss function, which allows for an end-to-end joint training strategy.

2. BOUNDARY OBJECTNESS NETWORK

As illustrated in Fig.2, the framework of our object detector includes four main modules, i.e., the modules of basenet, proposal generation, localization and classification. In the basenet module, we forward the image through a sequence of convolutional layers to extract feature maps of the entire image. In the proposal generation module, we produce a series of objectness [14] maps to give a search guide for candidate proposals. In the localization module, we infer the bounding box of an object by measuring the boundary objectness of candidate proposals. In the classification module, we predict the category for each box. In the following, we introduce each module in detail.

2.1. Basenet module

VGG-16 [15] is used as the basenet, which is pre-trained on ImageNet [16] dataset. Compared with methods using a single layer for recognition or detection, many works have demonstrated that utilizing multi-level features in CNNs through skip-connnections is beneficial to promoting information complementary and localizing all scales of objects [12] [17] [18]. Therefore, we combine the multiple feature maps of different resolutions into a single feature cube, as shown in Fig.2, which is called fused feature maps. In order to match dimensions, we use convolutional kernels with different strides (4, 2 and 1) to convolve with the output of three feature maps, respectively. More details are shown in the right table of Fig.2.

2.2. Proposal generation module

In the proposal generation module, the fused feature maps are as input and a bank of objectness maps are output. Each element on the objectness map corresponds to a specific region in the image. For each region, we predefine K anchors with different sizes and aspect ratios. For each objectness map, it predicts one kind anchor, and thus there are in total K objectness maps. (In the experiments, $K = 9$ is used the same as as [10], i.e., 3 sizes and 3 aspect ratios.) Note that the anchors with larger size have a large reception field, and thus the different kernel sizes (1×1, 3×3 and 5×5) are applied for the anchors with different sizes, which are shown in the right table of Fig.2. After each proposal is scored by objectness, non-maximum suppression (NMS) is adopted to remove highly-overlapped proposals. Then, the top-R ranked proposals are selected.

2.3. Localization module

Given a candidate proposal, we enlarge it by a factor γ. Each boundary of the enlarged proposal is divided into V equal intervals, each of which is assigned two boundary objectness, i.e., in-out and border objectness. If the interval is incorporated in the ground truth bounding box, then the in-out objectness is assigned as 1, and otherwise 0. If the interval is aligned with the boundary of the ground truth bounding box, then the border objectness is assigned as 1, and otherwise 0.

Assuming that the ground truth bounding box is represented as (x_1, y_1, x_2, y_2), where (x_1, y_1) and (x_2, y_2) denote the coordinates of top-left and bottom-right, respectively. Then the formulation of target boundary objectness in the x-axis can be written as:

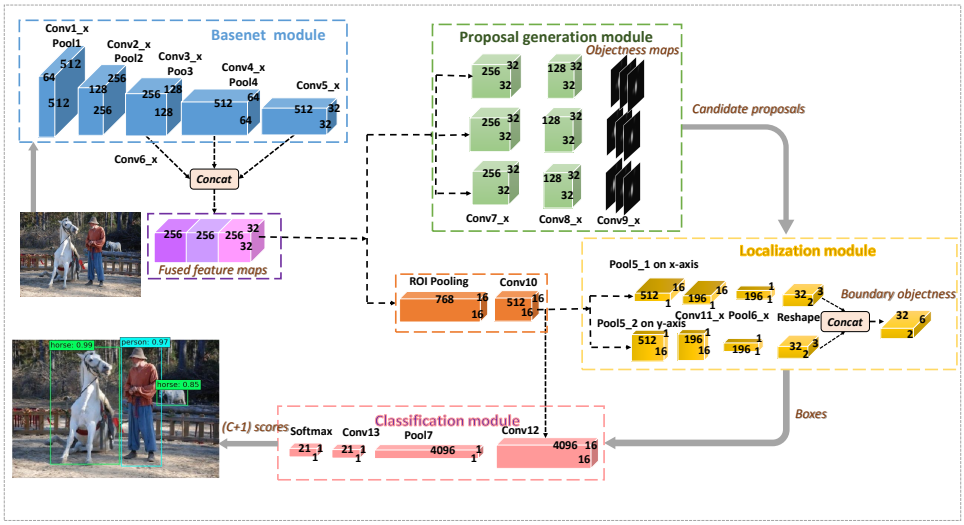
$$\begin{aligned} I_v &= \begin{cases} 1 & \text{if} \quad x_1 \le L(v) \le x_2 \\ 0 & \text{otherwise} \end{cases}, \\ B_v &= \begin{cases} 1 & \text{if} \quad L(v) = x_1 \quad \text{or} \quad x_2 \\ 0 & \text{otherwise} \end{cases}, \end{aligned} \tag{1}$$

where $v \in \{1, ..., V\}$, and $L(v)$ denotes the v-th interval coordinate of the enlarged proposal on the x-axis boundary; I_v and B_v denote the in-out and border objectness of the v-th interval, respectively. The target boundary objectness in the y-axis is the same as that in the x-axis. Note that V decides the elaboration extent of boundary objectness, and thus it has a significant impact on the detection performance of small object, as validated in Sec. 4.

In order to learn the boundary objectness, the localization module branches into two streams, X and Y, each responsible for yielding an axis-boundary objectness. Firstly, the feature maps of the enlarged proposal are extracted from the fused feature maps by the ROI pooling layer [8]. After a convolutional layer, we obtain the feature maps F of a size $16 \times 16 \times 512$, as the input of the localization module. Then, the max pooling layer is applied in F along with the x-axis and y-axis. Consequently, feature maps F_x and F_y are produced, which can be formulated as:

$$F_X(i, l) = \max_j F(i, j, l), \quad F_Y(j, l) = \max_i F(i, j, l), \tag{2}$$

where $F(i, j, l)$ denote the element located at (i, j) in the l-th channel of F. Then, F_X and F_Y are fed into a 1×1 convolutional layer followed by a global average pooling layer. Finally, the two branches are concatenated, and then a sigmoid layer outputs the x-axis and y-axis boundary objectness together.



Layer name	Ker. Size
Conv1_x	[3×3×64]×2
Conv2_x	[3×3×128]×2
Conv3_x	[3×3×256]×2
Conv4_x	[3×3×512]×2
Conv5_x	[3×3×512]×2
Conv6_1	6×6×256
Conv6_2	4×4×256
Conv6_3	1×1×256
Conv7_1	1×1×256
Conv7_2	3×3×256
Conv7_3	5×5×256
Conv8_x	[1×1×128]×3
Conv9_x	[3×3×3]×3
Conv10	3×3×512
Conv11_x	[1×1×196]×2
Conv12	1×1×4096
Conv13	1×1×21
Pool1	2×2
Pool2	2×2
Pool3	2×2
Pool4	2×2
Pool5_1	16×1
Pool5_2	1×16
Pool6	16×1
Pool7	16×16

Fig. 2. The framework of BON for object detection. **Left:** BON overview. **Right:** Kernel sizes used in BON.

2.4. Classification module

In general, the object detection methods have two sibling output layers for each proposal, i.e., bounding box offsets and $(C+1)$ scores (C object classes plus 1 for background). In other words, localization and classification are simultaneously implemented. We argue that the two modules should be cascaded, it is because inaccurate localization may make a shift of feature extraction, leading to the failure of classification. As shown in Fig.1, the candidate proposal covers part of a person and a horse. The previous CNN-based methods directly predict its category, which may lead to the high scores of both categories. In contrast, we firstly refine the location of one object (person), and then the classification module can give a high confidence to it. For this reason, we classify each object after the accurate localization, rather than simultaneously outputing the bounding box and category scores.

3. OPTIMIZATION

As Fig. 2 shows, BON has three output branches of proposal generation, localization and classification, for each of which we define a loss function.

The first task loss L_{obj} is log loss over binary classes ($u=1$ indicates there exists an object, and otherwise no object), which is defined as:

$$L_{obj}(p^{obj}, u) = -\log p_u^{obj}, \tag{3}$$

where p_u^{obj} denotes the objectness score for the class u.

Recall that each boundary of a proposal is divided into V equal intervals, I_v and B_v are the target boundary objectness of the v-th interval, defined in (1). Assuming that p_v^I and p_v^B are the predicted in-out and border boundary objectness, respectively, then the second task loss L_{loc} can be written as:

$$L_{loc}(p^I, p^B, I, B) = -\sum_{v=1}^{V}(I_v \log p_v^I + (1-I_v)\log(1-p_v^I) + \rho^+ B_v \log p_v^B + \rho^-(1-B_v)\log(1-p_v^B)). \tag{4}$$

The former two terms in (4) formulate the loss of the in-out case, while the latter two terms are of the border case. The same as LocNet, we set $\rho^- = 0.5 \cdot \frac{V}{V-1}$ and $\rho^+ = (V-1)\rho^-$ to balance the border and non-border element.

The third task loss L_{cls} is computed by a softmax over $(C+1)$ outputs for each ROI, which is expressed as:

$$L_{cls}(p^{cls}, c) = -log p_c^{cls}, \tag{5}$$

where $c \in (1, .., C+1)$ denotes the object class, and p_c^{cls} represents the confidence for the class c.

Combining (3), (4) and (5), we introduce a multi-task loss function to jointly train for these three branches:

$$L = \frac{\alpha}{N_{obj}} L_{obj}(p^{obj}, u) + \frac{\beta}{N_{loc}} L_{loc}(p^I, p^B, I, B) + \frac{1}{N_{cls}} L_{cls}(p^{cls}, c). \tag{6}$$

The hyper-parameters α and β in (6) control the balance between three task losses. Besides, each loss term is normalized by the corresponding batch size, i.e., N_{obj}, N_{loc} and N_{cls} for the proposal generation, localization and classification task, respectively.

4. EXPERIMENTS

In the experiments, all methods are trained on the union set of PASCAL VOC2007 trainval [19] and VOC2012 trainval and tested on VOC2007 test set (4952 images with 20 categories). In the training stage, we use the proposals generated by selective search [9] to simultaneously train three tasks of BON, which allows for an end-to-end joint training strategy. We set the initial learning rate to 10^{-3}, weight decay to 0.0005 and momentum to 0.9. Moreover, we set $\alpha = 20$ and $\beta = 10$, which are defined in (6). Note that all the hyper-parameters are tuned over the validation. Our implementation uses a single Nvidia Titan GPU on Caffe [20].

4.1. Evaluation on localization Performance

Here we evaluate the localization performance of BON. Following [10] [12] [13], we provide the recall as a function of the IoU thresholds for the proposals. We compare BON-V28 (V=28) and BON-V36 (V=36) with EdgeBoxes, RPN and

Table 1. Comparison of object detection performance.

Method	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Fast R-CNN	70.0	77.0	78.1	69.3	59.4	38.3	81.6	78.6	86.7	42.8	78.8	68.9	84.7	82.0	76.7	69.9	31.8	70.1	74.8	80.4	70.4
Faster R-CNN	73.2	76.5	79.0	70.9	65.5	52.1	83.1	84.7	86.4	52.0	81.9	65.7	84.8	84.6	77.5	76.7	38.8	73.6	73.9	83.0	72.6
HyperNet	76.3	77.4	83.3	75.0	69.1	62.4	83.1	87.4	87.4	57.1	79.8	71.4	85.1	85.1	80.0	79.1	51.2	**79.1**	**75.7**	80.9	76.5
SSD	76.8	**82.4**	84.7	**78.4**	73.8	53.2	86.2	**87.5**	86.0	57.8	83.1	70.2	84.9	85.2	**83.9**	**79.7**	50.3	77.9	73.9	82.5	75.3
BON-V28	77.0	78.2	83.3	76.9	72.4	61.2	86.0	85.8	88.0	**59.3**	**84.4**	71.1	85.7	**85.9**	81.0	78.4	49.6	78.1	74.2	**83.6**	**76.9**
BON-V36	**77.6**	79.8	**84.9**	78.2	**73.9**	**62.5**	**86.9**	86.6	**88.9**	59.1	83.4	**71.9**	**85.8**	85.5	81.9	78.3	**51.8**	78.5	74.8	82.6	76.3

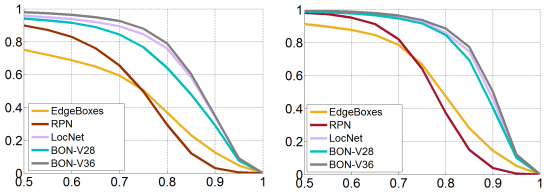


Fig. 3. Recall versus IoU thresholds. **Left**: top-50 proposals. **Right**: top-300 proposals.

LocNet[1]. Then top-50 and top-300 ranked proposals are respectively used based on the confidence generated by these methods.

As Fig.3 shows, both our method and LocNet get good results across a variety of IoU thresholds, which is desirable in practice and plays an important role in object detection. Note that with 50 proposals, BON-V28 performs a little worse than LocNet, we consider this is because LocNet uses a category-specific localizer, while BON works in an category-agnostic way. However, by increasing V to 36, we observe that BON-V36 surpasses LocNet. As we have mentioned before, the parameter V has a significant impact on the localization performance. Larger V divides the boundary into more elaborate intervals, objects especially with small size are more likely to be detected. Nevertheless, when V continues to increase, the performance is nearly saturated. Another observation is that with more proposals, the gap between LocNet and our method is smaller, and V is not the most decisive factor. In Fig.4, we show some localization examples with BON.

4.2. Evaluation on detection performance

We also compare BON to Fast R-CNN, Faster R-CNN, HyperNet and SSD for generic object detection. The performance is measured by mean average precision (mAP) over IoU = 0.5. As shown in Tab.1, BON-V28 is already better than any other method. Furthermore, BON-V36 gets mAP of 77.6 %, 1.3% higher than HyperNet and 0.8% higher than SSD, outperforming Fast R-CNN and Faster R-CNN by a large margin of 7.0% and 4.4%, respectively. Moreover, BON-V36 gets a large improvement in detecting the small objects (e.g., bottle and plant) compared to BON-V28, which could be attributed to the elaborate divide of the boundary. In Fig.4, we show some detection examples with BON.

[1]LocNet is not used independently, which generally follows an object detector. The same as [13], we use LocNet with $V = 28$ after Fast R-CNN in the experiments.

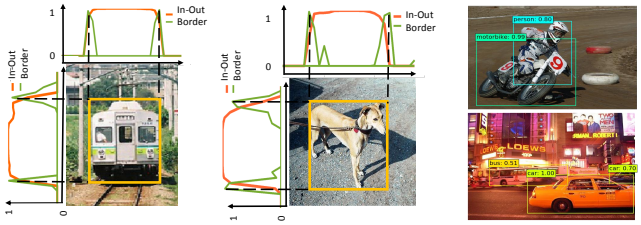


Fig. 4. **Left&middle:** localization examples with BON. **Right:** detection examples with BON.

4.3. Evaluation on accuracy and efficiency

Table 2. Comparison of detection accuracy and running time.

Methods	mAP			time(msec)
IoU	0.5	0.65	0.8	-
Fast R-CNN	74.6	58.7	33.3	1830
Faster R-CNN	73.2	59.9	31.5	**198**
LocNet	77.5	64.8	41.6	2031
BON-V28	77.0	63.2	39.8	227
BON-V36	**77.6**	**66.3**	**43.8**	239

We examine both the accuracy and efficiency, respectively measured by mAP over IoU = 0.5, 0.65, 0.8 and runtime. We compare BON-V28 and BON-V36 to Fast R-CNN, Faster R-CNN and LocNet. The result in Tab.2 shows that both LocNet and our method yield much better mAP results over each IoU, this is due to the superior localization performance. Compared to Fast R-CNN, BON-28 outperforms it both in mAP and runtime. Compared to Faster R-CNN, BON-V28 takes a bit more time, but outperforms a lot in accuracy. Compared to LocNet, BON-28 achieves a little worse mAP, but shows an absolute advantage in speed. Moreover, BON-V36, taking more 12 msec than BON-28, is more accurate and faster than LocNet.

5. CONCLUSION

We have presented BON, a fully trainable deep neural network for objection detection. In contrast to the previous object detection methods, which apply the bounding box regressor to localize the objects, we develop a powerful category-agnostic localizer based on the recently introduced LocNet model. Moreover, the multi-task loss function allows for an end-to-end joint training strategy, and the sharing of the convolutional layers enables efficient detection at inference time. Extensive experiments on the PASCAL VOC datasets demonstrate that our method achieves a remarkable improvement on localization performance, detection accuracy and runtime.

6. REFERENCES

[1] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained partbased models. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.32, no.9, pp.1627C1645, 2010.

[2] T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-SVMs for object detection and beyond. In ICCV. IEEE, 2011.

[3] J. Carreira and C. Sminchisescu. Constrained parametric mincuts for automatic object segmentation. In CVPR, 2010.

[4] I. Endres and D. Hoiem. Category independent object proposals. In ECCV, 2010.

[5] C. L. Zitnick and P. Dollar. Edge boxes: Locating object proposals from edges. In Computer VisionCECCV 2014.

[6] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks. International Conference on Neural Information Processing Systems. Curran Associates Inc. pp. 1097-1105, 2012.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR, 2014.

[8] R. Girshick. Fast r-cnn. In ICCV, 2015.

[9] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. IJCV, 2013.

[10] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In NIPS, 2015.

[11] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. Reed. SSD: Single shot multibox detector. In ECCV, 2016.

[12] T. Kong, A. Yao, Y. Chen, and F. Sun. Hypernet: Towards accurate region proposal generation and joint object detection. In CVPR, 2016.

[13] Gidaris S, Komodakis N. LocNet: Improving localization accuracy for object detection. Computer Vision and Pattern Recognition. IEEE, pp.789-798, 2016.

[14] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. TPAMI, 2012.

[15] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In ICLR, 2015.

[16] J. Deng, A. Berg, S. Satheesh, H. Su, A. Khosla, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Competition 2012 (ILSVRC2012). http://www.image-net.org/challenges/LSVRC/2012/.

[17] Huang G, Liu Z, Weinberger K Q, et al. Densely Connected Convolutional Networks. 2016.

[18] Zeng X, Ouyang W, Yang B, et al. Gated Bi-directional CNN for Object Detection. Computer Vision C ECCV 2016.

[19] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results, 2007.

[20] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. arXiv:1408.5093, 2014.