Atrous Filter Design for Enhancing SSD' s Small Object Detection Capability: A Postprint

Authors: Wen Jiewen, Zhan Yinwei, Li Chuhong, Lu Jianbiao

Date: 2018-05-02T00:00:00+00:00

Abstract

To address the issue of limited small object detection capability in the real-time object detection SSD (Single Shot MultiBox Detector) algorithm, a design strategy for Atrous filters to enhance feature map resolution is proposed. The improved algorithm, built upon the SSD network architecture, concatenates the feature maps generated by the third and fourth convolutional layers after normalization, and then increases the resolution of these feature maps through Atrous convolution operations. These feature maps collectively provide the essential features required for small object detection. Furthermore, the improved SSD algorithm incorporates the SeLU (Scaled Exponential Linear Units) activation function and designs a comprehensive data augmentation scheme in the data preprocessing stage. Experimental results demonstrate that this improved algorithm framework achieves higher detection accuracy, superior robustness, and significantly enhanced performance in small object detection compared to the original SSD algorithm framework.

Full Text

Preamble

Design of Atrous Filter to Strengthen Small Object Detection Capability of SSD

Wen Jiewen, Zhan Yinwei, Li Chuhong, Lu Jianbiao
(School of Computer Science & Technology, Guangdong University of Technology, Guangzhou 510006, China)

Abstract: To address the limited capability of the real-time object detection SSD (Single Shot MultiBox Detector) algorithm in detecting small objects, this paper proposes an Atrous filter design strategy that enhances feature map resolution. The improved algorithm builds upon the SSD network architecture by

concatenating the feature maps generated from the third and fourth convolutional layers after normalization, then increasing the resolution of these feature maps through Atrous convolution operations. These concatenated feature maps collectively provide the necessary features for small object detection. Additionally, the improved SSD algorithm incorporates the SeLU (Scaled Exponential Linear Units) activation function and designs a data augmentation method during the data preprocessing stage. Experiments demonstrate that the proposed algorithm framework achieves higher detection accuracy and better robustness compared to the original SSD algorithm framework, with particularly noticeable improvements in small object detection.

Keywords: SSD; object detection; Atrous

0 Introduction

Object detection is one of the most popular and challenging research topics in computer vision in recent years, with critical real-world applications such as autonomous driving and intelligent surveillance. In particular, the theoretical research and practical applications of deep learning have provided powerful support for computer vision. Object detection algorithms are divided into traditional methods and those combined with deep learning. Traditional object detection algorithms require manual intervention during feature extraction to obtain target-related feature information from raw image inputs. This manual feature extraction heavily depends on the prior knowledge of feature designers, resulting in low efficiency. Moreover, the irreversible loss of useful information during feature extraction increases classification errors during training.

Deep learning-based object detection algorithms are further divided into region proposal-based methods and regression-based methods. Classic region proposal-based algorithms include works such as \cite{3~9}, with candidate generation methods including SS (Selective Search) [**?**], EB (Edge Boxes) [**?**], and RPN (Region Proposal Network) [**?**]. Among these, the RPN algorithm proposed in [**?**] uses convolutional neural networks to directly generate candidate regions, integrating the previously separate candidate region generation and CNN classification into a unified framework. Additionally, the anchor mechanism adopted by RPN can accurately map target bounding box coordinates, thereby reducing localization errors and improving detection accuracy. While this series of region proposal-based algorithms remains mainstream in detection research, their detection speeds generally fail to meet real-time requirements.

To address the speed bottleneck in object detection, regression-based detection algorithms such as YOLO [**?**] and SSD [**?**], along with their improved versions YOLOv2 [**?**] and DSSD [**?**], have emerged. YOLO designs a Darknet neural network structure that takes the entire image as input, converting object detection into a regression problem. The algorithm directly regresses target bounding box coordinates and class probabilities at the network output layer, achieving 45 fps detection speed with GPU support. However, due to its loss function design,

YOLO suffers from large classification and localization errors and weak generalization capability. YOLOv2 makes significant improvements in data input, network structure, and localization methods, and currently represents the best comprehensive performance in terms of detection speed and accuracy.

The SSD algorithm framework studied in this paper offers several advantages. First, in terms of detection speed, it converts the convolutional and fully connected layers of Faster R-CNN into a fully convolutional network structure, greatly improving detection speed. Second, in detection accuracy, it extends Faster R-CNN' s RPN anchor mechanism to feature maps of various scales, where each pixel on a feature map corresponds to several anchors. The network trains these anchors while simultaneously driving feature training, resulting in very high detection precision.

However, research has found that the SSD algorithm framework has limited capability for small object detection. First, SSD is a fully convolutional network-based detection framework that uses different layers of the CNN to detect objects of different sizes. In the network model, earlier feature maps have large spatial dimensions but insufficient contextual semantics, while later feature maps have rich contextual semantics but become very small after multiple pooling operations. In the VGG-16 network structure used by SSD, a 32×32 object corresponds to only a 2×2 feature map after the conv5_3 layer, resulting in significant loss of location information. Therefore, detecting small objects requires both sufficiently large feature maps to provide finer details and dense sampling, as well as rich semantic information to distinguish objects from backgrounds.

Second, the SSD network model is fully convolutional and can accept images of arbitrary sizes without requiring all training and test images to have the same dimensions. For real-time speed considerations, the input image size is fixed, which significantly impacts small object detection in large images. To address these two problems, this paper proposes three solutions: (a) perform feature sampling in relatively shallow but high-resolution convolutional layers while using Atrous filters to increase the size of certain feature maps and improve their resolution to provide more effective features; (b) adopt the SeLU activation function to make model training more robust; and (c) design a data augmentation rule different from the original that allows the network model to accept input images of different sizes.

1 Related Work

Most deep learning-based object detection algorithms or classification models consider two aspects to obtain more effective features: multi-scale strategies and deeper, more effective neural network structures. Regarding multi-scale strategies, there are two approaches. First, combining feature maps generated by multiple convolutional layers of the network model for prediction. ION [**?**] uses L2 normalization to perform feature sampling at different layers of the network structure to generate object proposals. HyperNet [**?**] adopts a similar

method for feature sampling and proposal generation. Feature maps from different layers contain feature information at different hierarchical levels of the input image, which is beneficial for object localization and classification. However, this approach also increases model computation and reduces detection speed.

Second, different convolutional layers can be designed with different receptive field sizes. Large receptive fields are used for predicting large objects, while small receptive fields are used for predicting small objects. Therefore, different scales can be used to predict objects at different layers of the neural network model. MS-CNN [**?**] applies deconvolution at multiple layers of a CNN to increase feature map resolution, then performs candidate region learning and feature sampling on these layers. To better detect small objects, these methods use small, shallow receptive fields and dense feature maps with contextual semantic information.

In terms of deep neural network structures [**?**], deep learning-based object detection frameworks adopt deeper network structures to achieve better classification and detection accuracy. Currently, classic and commonly used network structures include AlexNet [**?**], VGG [**?**], GoogleNet [**?**], and ResNet [**?**]. The SSD algorithm framework adopts the VGG-16 model as its base network structure, using the first five layers of this model, then converting the fully connected layers fc6 and fc7 into two convolutional layers, and adding three additional convolutional layers plus an average pooling layer. As an improved version of SSD, DSSD not only uses deconvolution to increase feature map resolution and enhance contextual semantic learning but also adopts the deeper ResNet-101 network model with better classification accuracy. While DSSD achieves higher detection accuracy than SSD, it loses real-time detection capability.

2 Atrous Filter Design

According to [**?**], Atrous filters were originally used for image processing in wavelet transforms. Atrous can compute convolutional feature responses at any resolution in any layer. First, consider the one-dimensional signal Atrous convolution calculation:

$$y[i] = \sum_{k=1}^{K} x[i + r \cdot k] \cdot w[k]$$

where $x[i]$ is the one-dimensional signal input, $y[i]$ is the output signal after computation, $w[k]$ is the filter, K is the length of the filter, and the rate parameter r is equivalent to the stride when sampling the input signal. [Figure 1: see original paper] illustrates the specific process of one-dimensional signal Atrous convolution.

Figure 1(a) shows the standard convolution process for sparse feature extraction with low-resolution input, while (b) shows the Atrous convolution process. Compared to Figure 1(a), (b) has larger extension (pad=2). Setting the rate to

2 involves inserting zeros between each value in the input feature map matrix before performing convolution operations. Finally, three input signals undergo extension, interpolation, and convolution operations to output five activation values, thereby increasing the feature map size.

Similar operations exist for two-dimensional input images. [Figure 2: see original paper] shows the processes of sparse and dense feature extraction. In Figure 2, the top row represents sparse feature extraction. Given an image, a downsampling operation reduces the image resolution with a stride factor of 2, followed by convolution with a Gaussian kernel of size 7. The resulting feature map is only 1/4 the size of the original image. The bottom row shows dense feature extraction. This paper upsamples the filter with a stride of 1 while interpolating with a rate value of 2. Although the filter size is enlarged, only non-zero filter values are considered, so the filter parameters and operations remain unchanged at each position. This Atrous convolution operation enables controllable feature responses for low-resolution images.

Based on the above description, if the filter size is $k \times k$, then when the rate value is r, $r-1$ zero values need to be inserted into the filter. The filter size will expand to $k + (k-1)(r-1)$. [Figure 3: see original paper] shows the network structure of the SSD algorithm framework. This framework adopts the VGG-16 base network structure, using the first five layers of VGG-16, then converting the fc6 and fc7 layers into two convolutional layers, adding four additional convolutional layers, and removing all Dropout layers and the fc8 layer. Feature maps from different hierarchical levels are used for predicting object bounding box offsets and class scores, with final detection results obtained through NMS (non-maximum suppression).

The smallest-scale object detection in SSD is primarily determined by the feature map generated from the conv4_3 layer, which is relatively early in the network structure. This paper makes the following improvements to the SSD algorithm framework based on the Atrous filter principle:

a) Extract features simultaneously from the conv3_3 and conv4_3 layers of the VGG-16 base network, with stride settings of 4 for conv3_3 and 8 for conv4_3. The feature maps generated by these two convolutional layers are then concatenated after normalization.

b) Use Atrous filters to improve resolution. This paper sets the stride of the pool3 layer from 2 to 1, then interpolates and expands all filters in the conv4 layer with a rate value of 2, thereby increasing feature map resolution.

These two approaches first consider enhancing feature information from lower layers by concatenating the feature maps from conv3_3 and conv4_3 after normalization, then strengthen feature map resolution through Atrous convolution operations at the conv4_3 layer.

3 SeLU Activation Function

The SSD algorithm network structure uses the ReLU activation function. During backpropagation, ReLU reduces the likelihood of gradient vanishing, allowing parameters in earlier layers to update quickly. During forward propagation, ReLU only requires threshold setting, which speeds up computation. However, when training neural network models, ReLU can easily cause training interruptions. A large gradient passing through a ReLU neuron may result in the neuron no longer being activated for any data after parameter updates. Therefore, using ReLU requires setting a relatively small and appropriate learning rate.

In 2017, [**?**] introduced a new activation function called SeLU (scaled exponential linear units), defined as:

$$f(x) = \begin{cases} \lambda x & \text{if } x > 0 \\ \lambda\alpha(e^x - 1) & \text{if } x \leq 0 \end{cases}$$

SeLU introduces self-normalizing properties that automatically converge neuron activations to zero mean and unit variance. Unlike batch normalization which requires explicit normalization, SeLU activations approximate zero mean and unit variance, and even with noise and perturbations, they converge to zero mean and unit variance after forward propagation through many layers. This convergence property enables: (a) training of very deep neural networks, (b) use of strong regularization, and (c) more robust learning. Furthermore, for activations not approximating unit variance, the variance has upper and lower bounds, making gradient vanishing and explosion impossible.

Normalization techniques in deep neural network training are typically perturbed by stochastic gradient descent (SGD), stochastic regularization (such as dropout), and other parameters. This paper aims to automatically shift and rescale neuron activations to achieve zero mean and unit variance without explicit normalization. Based on these excellent properties of the SeLU activation function, this paper considers replacing the ReLU activation function in the SSD algorithm network structure with SeLU to increase network robustness.

4 Image Preprocessing

To make the model more robust to objects of different sizes and shapes, SSD adopts specific rules for data augmentation: (a) use the entire input image; (b) sample images with ratios of 0.1, 0.3, 0.5, 0.7, and 0.9 relative to the original image; (c) randomly sample images where the size ratio to the original is between [0.1, 1] and the aspect ratio is between [0.5, 2]. After sampling, when the ground truth object center falls within the sampled image, SSD retains the overlapping portion and resizes the sampled image to a fixed size.

Research shows that for small object detection, this sampling rule can be better adjusted. In sampled images, the basic object size ratio to the original image

is also in [0.1, 1]. However, the 0.1 ratio is still relatively large—for example, with a 512×512 input image, sampling at ratio 0.1 yields a 51×51 image. Based on this analysis, this paper adjusts the basic object size ratio to original image ratio for small object detection. The improved algorithm uses a ratio range of [1/64, 1], and correspondingly sets the sampling ratios to 1/64, 1/32, 1/16, 1/8, 1/4, 1/2, and 1.

This design aims to: (a) reduce the sampling ratio so that small objects become more prominent when the sampled image is resized to a fixed dimension; (b) increase sensitivity to small object regions—SSD originally had 5 sampling ratios, while this paper uses 7, with 6 ratios smaller than 0.5.

5 Experimental Results and Analysis

5.1 Experimental Setup

The experimental hardware configuration includes an Intel Xeon E5-2620 v2 processor, NVIDIA GTX 980ti GPU, and 64 GB RAM server, with software environment consisting of Ubuntu, GCC, CUDA, OpenCV, and the Caffe framework. GPU acceleration uses CUDA programming, while OpenCV is primarily for image display during testing.

SSD algorithm training and evaluation are conducted primarily on the PASCAL VOC 2007 and PASCAL VOC 2012 datasets. PASCAL VOC is a standard dataset for visual object classification and detection, providing images including 20 object categories. This paper uses the validation and test sets of PASCAL VOC 2007 and the validation set of PASCAL VOC 2012 as the training set, and the test set of PASCAL VOC 2012 for testing.

5.2 Evaluation Metrics

Depending on performance emphasis, object detection has various evaluation metrics such as detection accuracy, efficiency, and localization accuracy. This paper focuses on detection accuracy, using mAP (mean average precision) as the most important evaluation metric. The calculation process is:

a) Compute average precision for each class:

$$AP = \frac{1}{R}\sum_{j=1}^{n}\frac{R_j}{j}\cdot I_j$$

where R is the total number of relevant objects (detected and undetected) in a class, n is the number of objects, I_j is 1 if the j-th object is relevant and 0 otherwise, and R_j is the number of relevant objects among the first j objects.

b) Take the average of average precisions across multiple classes.

mAP values range between 0 and 1, with higher values indicating better detection accuracy. This paper uses frames per second (FPS) to measure detection speed, with 25 fps as the real-time threshold.

5.3 Implementation Details

The algorithm framework modifications are based on the original SSD framework and VGG-16 classification model, with changes made to conv3_3, pool3, and conv4 layers of the SSD network structure, and replacement of ReLU with SeLU activation functions. Implementation uses the Caffe framework with reference to open-source code from SSD and SeLU papers.

The model is pretrained on ImageNet dataset classification and localization tasks, then fine-tuned for detection. This paper uses stochastic gradient descent with an initial learning rate of 0.001, momentum of 0.9, weight decay of 0.005, and batch size of 32. Different datasets use different learning rate schedules. The convolutional network structure uses random initialization. During training, increasing iteration count is necessary. This paper trains with a learning rate of 0.001 for 60,000 iterations, then 0.0001 for 30,000 iterations, and finally 0.00001 for 10,000 iterations.

During training, the algorithm needs to label positive and negative samples, determined by the IOU (intersection-over-union) between ground truth bounding boxes and predicted boxes. An IOU threshold of 0.5 is set as positive samples, otherwise negative.

5.4 Experimental Results and Analysis

This paper conducts three groups of comparative experiments to test and verify the improved algorithm: Section 5.4.1 compares the training effects of various object detection frameworks; Section 5.4.2 compares detection accuracy on the PASCAL VOC 2012 test set; Section 5.4.3 shows the improved algorithm' s effectiveness.

5.4.1 Atrous Experiment Comparison The improved algorithm extracts small object-related features in relatively low network layers. In the original SSD algorithm, features extracted from the conv4_3 layer are sensitive to small object detection. This design concatenates feature maps from conv3_3 and conv4_3 after normalization to jointly provide features for small object detection. Experiments also test other layers' small object feature extraction capabilities. lists the final effects of feature extraction from various SSD layers.

The improved algorithm Ours achieves a 1.2% mAP improvement over SSD300, but due to the added Atrous processing, the computational load increases, reducing real-time performance by 9 fps. Ours512 improves detection accuracy by 0.4% over Ours300 but reduces speed by 7 fps. Compared to SSD512, the proposed algorithm shows very good real-time performance. Relative to YOLOv2

288 and YOLOv2 544, the improved algorithm has relative advantages in real-time performance under high-resolution input. Compared to YOLO, R-CNN Minus R, Fast R-CNN, and Faster R-CNN, the proposed algorithm shows advantages in both detection accuracy and speed.

In theory, earlier convolutional layers can provide rich semantic information for object detection, but this feature extraction capability is not linear. This paper compares the ability of several convolutional layers to provide features for small objects. As shown in , single-layer feature extraction performs best at conv4_3 with 74.3% mAP, which is also the design in the original SSD framework. Next is conv3_3, while conv5_3 performs worst. When concatenating feature maps from multiple layers, conv3_3 and conv4_3 together improve algorithm mAP. Building upon these concatenated feature maps, using Atrous filters to enhance resolution further improves mAP to 75.5%, the best among all comparison experiments. In terms of speed, single-layer or multi-layer feature provision adds no extra computation, maintaining 59 fps, while the Atrous convolution and interpolation operations increase computational load, reducing detection speed but still maintaining real-time performance at 50 fps.

5.4.2 Comparison of Algorithm Frameworks This paper compares several popular object detection frameworks: YOLO, YOLOv2, SSD, R-CNN Minus R, Fast R-CNN, and Faster R-CNN. Training is conducted on PASCAL VOC 2007 and 2012 datasets. compares detection accuracy (mAP) and speed across typical algorithm frameworks.

Analysis of reveals two factors affecting detection accuracy: (a) input image scale —generally, higher input resolution yields better detection accuracy, as evidenced by YOLOv2's five input scales and SSD/Ours's two scales; (b) the Atrous design strategy used in this paper also improves detection accuracy. Correspondingly, factors affecting real-time performance are: (a) higher input resolution requires greater computation, reducing real-time capability; (b) the Atrous filter design strategy also increases computational load, decreasing real-time performance.

5.4.3 PASCAL VOC 2012 Test Comparison Various algorithms are tested and compared on PASCAL VOC 2012 test set. records detection accuracy for each class and average mAP. The comparison shows that the improved algorithm Ours performs excellently not only on large objects but also achieves the best detection results for several small object categories among all compared algorithms. In Ours512, large objects such as aero, boat, bus, dog, mbike, person, and sofa achieve the highest detection accuracy, while small objects like bird, bottle, and plant also achieve the highest accuracy.

Individual detection accuracy is relatively high for Faster R-CNN, SSD, and Ours. R-CNN divides too many region proposals, and using CNN for feature extraction on each ROI results in enormous computational load, affecting detection performance. YOLO' s final two fully connected layers are relatively

inaccurate in regressing localization coordinates, directly affecting YOLO' s localization precision. Faster R-CNN, SSD, and the improved algorithm Ours adopt or improve upon RPN for localization, resulting in smaller localization errors and higher detection accuracy.

5.4.4 Improved Algorithm Effectiveness [Figure 4: see original paper] compares detection results between the original SSD framework and the improved SSD framework. The comparison shows that the improved algorithm Ours can detect more objects than the original SSD. Objects that the original SSD misclassifies as background can be accurately localized and classified by the improved algorithm Ours.

6 Conclusion

This paper studies the real-time object detection SSD algorithm framework and proposes an improved algorithm that uses Atrous filters to increase feature map resolution, designs a data augmentation rule for the image preprocessing stage, and replaces ReLU with SeLU activation functions in the network structure. Experimental results demonstrate that the improved algorithm Ours shows superior performance in small object detection compared to the original SSD algorithm. Future research will combine traditional methods such as wavelet analysis to determine optimal resolution for feature maps processed by Atrous filters.

References

[13] Liu W, Anguelov D, Erhan D, et al. SSD: single shot multibox detector [C]// Proc of European Conference on Computer Vision. Springer International Publishing. 2016: 21-37.

[14] Joseph R, Farhadi A. YOLO9000: better, faster, stronger [J]. arXiv preprint arXiv: 1612. 08242, 2016.

[15] Fu Chengyang, Liu Wei, Ranga A, et al. DSSD: deconvolutional single shot detector [J]. arXiv preprint arXiv: 1701. 06659, 2017.

[16] Chen L C, Papandreou G, Kokkinos I, et al. DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs [J]. arXiv preprint arXiv: 1606. 00915, 2016.

[17] Bell S, Zitnick C L, Bala K, et al. Inside-outside net: detecting objects in context with skip pooling and recurrent neural networks [C]// Proc of IEEE Conference on Computer Vision and Pattern Recognition. 2016: 2874-2883.

[18] Kong Tao, Yao Anbang, Chen Yurong, et al. HyperNet: towards accurate region proposal generation and joint object detection [C]// Proc of IEEE Conference on Computer Vision and Pattern Recognition. 2016: 845-863.

[19] Cai Zhaowei, Fan Quanfu, Rogerio S, et al. A unified multi-scale deep convolutional neural network for fast object detection [C]// Proc of European Conference on Computer Vision. Springer International Publishing. 2016: 354-370.

[20] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks [C]// Advances in Neural Information Processing Systems. 2012: 1097-1105.

[21] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition [J]. arXiv preprint arXiv: 1409. 1556, 2014.

[22] Szegedy C, Liu W, Jia Y, et al. Going deeper with convolutions [C]// Proc of IEEE Conference on Computer Vision and Pattern Recognition. 2015: 1-9.

[23] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition [C]// Proc of IEEE Conference on Computer Vision and Pattern Recognition. 2016: 770-778.

[24] Klambauer G, Unterthiner T, Mayr A, et al. Self-normalizing neural networks [J]. arXiv preprint arXiv: 1706. 02515, 2017.

[25] Lenc K, Vedaldi A. R-CNN minus r [J]. arXiv preprint arXiv: 1506. 06981, 2015.

[26] Zhou Feiyan, Jin Linpeng, Dong Jun. Survey of convolutional neural network research [J]. Chinese Journal of Computers, 2017, 40 (6): 1229-1251.

[1] Felzenszwalb P F, Girshick R B, McAllester D, et al. Object detection with discriminatively trained part-based models [J]. IEEE Trans on Pattern Analysis and Machine Intelligence, 2010, 32 (9): 1627-1645.

[2] Dalal N, Triggs B. Histograms of oriented gradients for human detection [C]// Proc of Computer Vision and Pattern Recognition. 2005: 886-893.

[3] Girshick R, Donahue J, Darrell T, et al. Rich feature hierarchies for accurate object detection and semantic segmentation [C]// Proc of IEEE Conference on Computer Vision and Pattern Recognition. 2014: 580-587.

[4] He K, Zhang X, Ren S, et al. Spatial pyramid pooling in deep convolutional networks for visual recognition [J]. IEEE Trans on Pattern Analysis and Machine Intelligence, 2015, 37 (9): 1904-1916.

[5] Girshick R. Fast R-CNN [C]// Proce of IEEE International Conference on Computer Vision. 2015: 1440-1448.

[6] Ren S, He K, Girshick R, et al. Faster R-CNN: towards real-time object detection with region proposal networks [C]// Advances in Neural Information Processing Systems. 2015: 91-99.

[7] Dai J, Li Y, He K, et al. R-FCN: Object detection via region-based fully convolutional networks [C]// Proc of Neural Information Processing Systems. 2016: 379-387.

[8] Lin T Y, Dollár P, Girshick R, et al. Feature pyramid networks for object detection [J]. arXiv preprint arXiv: 1612. 03144, 2016.

[9] He Kaiming, Gkioxari G, Dollár P, et al. Mask R-CNN [J]. arXiv preprint arXiv: 1703. 06870, 2017.

[10] Uijlings J R R, Van De Sande K E A, Gevers T, et al. Selective search for object recognition [J]. International Journal of Computer Vision, 2013, 104 (2): 154-171.

[11] Zitnick C L, Dollár P. Edge boxes: locating object proposals from edges [C]// Proc of European Conference on Computer Vision. [S. l.]: Springer International Publishing, 2014: 391-405.

[12] Redmon J, Divvala S, Girshick R, et al. You only look once: unified, real-time object detection [C]// Proc of IEEE Conference on Computer Vision and Pattern Recognition. 2016: 779-788.

Note: Figure translations are in progress. See original paper for figures.