Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun

Abstract—State-of-the-art object detection networks depend on region proposal algorithms to hypothesize object locations. Advances like SPPnet [1] and Fast R-CNN [2] have reduced the running time of these detection networks, exposing region proposal computation as a bottleneck. In this work, we introduce a *Region Proposal Network* (RPN) that shares full-image convolutional features with the detection network, thus enabling nearly cost-free region proposals. An RPN is a fully convolutional network that simultaneously predicts object bounds and objectness scores at each position. The RPN is trained end-to-end to generate high-quality region proposals, which are used by Fast R-CNN for detection. We further merge RPN and Fast R-CNN into a single network by sharing their convolutional features—using the recently popular terminology of neural networks with "attention" mechanisms, the RPN component tells the unified network where to look. For the very deep VGG-16 model [3], our detection system has a frame rate of 5fps (*including all steps*) on a GPU, while achieving state-of-the-art object detection accuracy on PASCAL VOC 2007, 2012, and MS COCO datasets with only 300 proposals per image. In ILSVRC and COCO 2015 competitions, Faster R-CNN and RPN are the foundations of the 1st-place winning entries in several tracks. Code has been made publicly available.

Index Terms—Object Detection, Region Proposal, Convolutional Neural Network.

arXiv:1506.01497v3 [cs.CV] 6 Jan 2016

1 Introduction

Recent advances in object detection are driven by the success of region proposal methods (*e.g.*, [4]) and region-based convolutional neural networks (R-CNNs) [5]. Although region-based CNNs were computationally expensive as originally developed in [5], their cost has been drastically reduced thanks to sharing convolutions across proposals [1], [2]. The latest incarnation, Fast R-CNN [2], achieves near real-time rates using very deep networks [3], *when ignoring the time spent on region proposals*. Now, proposals are the test-time computational bottleneck in state-of-the-art detection systems.

Region proposal methods typically rely on inexpensive features and economical inference schemes. Selective Search [4], one of the most popular methods, greedily merges superpixels based on engineered low-level features. Yet when compared to efficient detection networks [2], Selective Search is an order of magnitude slower, at 2 seconds per image in a CPU implementation. EdgeBoxes [6] currently provides the best tradeoff between proposal quality and speed, at 0.2 seconds per image. Nevertheless, the region proposal step still consumes as much running time as the detection network.

- *S. Ren is with University of Science and Technology of China, Hefei, China. This work was done when S. Ren was an intern at Microsoft Research. Email: sqren@mail.ustc.edu.cn*
- *K. He and J. Sun are with Visual Computing Group, Microsoft Research. E-mail: {kahe,jiansun}@microsoft.com*
- *R. Girshick is with Facebook AI Research. The majority of this work was done when R. Girshick was with Microsoft Research. E-mail: rbg@fb.com*

One may note that fast region-based CNNs take advantage of GPUs, while the region proposal methods used in research are implemented on the CPU, making such runtime comparisons inequitable. An obvious way to accelerate proposal computation is to re-implement it for the GPU. This may be an effective engineering solution, but re-implementation ignores the down-stream detection network and therefore misses important opportunities for sharing computation.

In this paper, we show that an algorithmic change—computing proposals with a deep convolutional neural network—leads to an elegant and effective solution where proposal computation is nearly cost-free given the detection network's computation. To this end, we introduce novel *Region Proposal Networks* (RPNs) that share convolutional layers with state-of-the-art object detection networks [1], [2]. By sharing convolutions at test-time, the marginal cost for computing proposals is small (*e.g.*, 10ms per image).

Our observation is that the convolutional feature maps used by region-based detectors, like Fast R-CNN, can also be used for generating region proposals. On top of these convolutional features, we construct an RPN by adding a few additional convolutional layers that simultaneously regress region bounds and objectness scores at each location on a regular grid. The RPN is thus a kind of fully convolutional network (FCN) [7] and can be trained end-to-end specifically for the task for generating detection proposals.

RPNs are designed to efficiently predict region proposals with a wide range of scales and aspect ratios. In contrast to prevalent methods [8], [9], [1], [2] that use

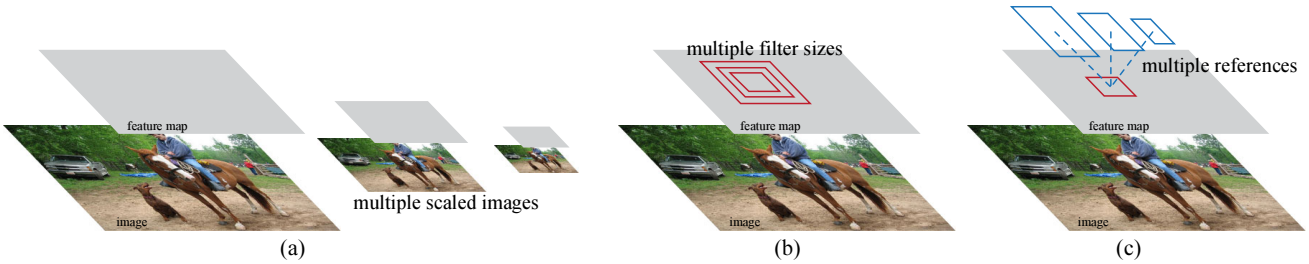


Figure 1: Different schemes for addressing multiple scales and sizes. (a) Pyramids of images and feature maps are built, and the classifier is run at all scales. (b) Pyramids of filters with multiple scales/sizes are run on the feature map. (c) We use pyramids of reference boxes in the regression functions.

pyramids of images (Figure 1, a) or pyramids of filters (Figure 1, b), we introduce novel "anchor" boxes that serve as references at multiple scales and aspect ratios. Our scheme can be thought of as a pyramid of regression references (Figure 1, c), which avoids enumerating images or filters of multiple scales or aspect ratios. This model performs well when trained and tested using single-scale images and thus benefits running speed.

To unify RPNs with Fast R-CNN [2] object detection networks, we propose a training scheme that alternates between fine-tuning for the region proposal task and then fine-tuning for object detection, while keeping the proposals fixed. This scheme converges quickly and produces a unified network with convolutional features that are shared between both tasks.[1]

We comprehensively evaluate our method on the PASCAL VOC detection benchmarks [11] where RPNs with Fast R-CNNs produce detection accuracy better than the strong baseline of Selective Search with Fast R-CNNs. Meanwhile, our method waives nearly all computational burdens of Selective Search at test-time—the effective running time for proposals is just 10 milliseconds. Using the expensive very deep models of [3], our detection method still has a frame rate of 5fps (*including all steps*) on a GPU, and thus is a practical object detection system in terms of both speed and accuracy. We also report results on the MS COCO dataset [12] and investigate the improvements on PASCAL VOC using the COCO data. Code has been made publicly available at `https://github.com/shaoqingren/faster_rcnn` (in MATLAB) and `https://github.com/rbgirshick/py-faster-rcnn` (in Python).

A preliminary version of this manuscript was published previously [10]. Since then, the frameworks of RPN and Faster R-CNN have been adopted and generalized to other methods, such as 3D object detection [13], part-based detection [14], instance segmentation [15], and image captioning [16]. Our fast and effective object detection system has also been built in commercial systems such as at Pinterests [17], with user engagement improvements reported.

In ILSVRC and COCO 2015 competitions, Faster R-CNN and RPN are the basis of several 1st-place entries [18] in the tracks of ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation. RPNs completely learn to propose regions from data, and thus can easily benefit from deeper and more expressive features (such as the 101-layer residual nets adopted in [18]). Faster R-CNN and RPN are also used by several other leading entries in these competitions[2]. These results suggest that our method is not only a cost-efficient solution for practical usage, but also an effective way of improving object detection accuracy.

2 Related Work

Object Proposals. There is a large literature on object proposal methods. Comprehensive surveys and comparisons of object proposal methods can be found in [19], [20], [21]. Widely used object proposal methods include those based on grouping super-pixels (*e.g.*, Selective Search [4], CPMC [22], MCG [23]) and those based on sliding windows (*e.g.*, objectness in windows [24], EdgeBoxes [6]). Object proposal methods were adopted as external modules independent of the detectors (*e.g.*, Selective Search [4] object detectors, R-CNN [5], and Fast R-CNN [2]).

Deep Networks for Object Detection. The R-CNN method [5] trains CNNs end-to-end to classify the proposal regions into object categories or background. R-CNN mainly plays as a classifier, and it does not predict object bounds (except for refining by bounding box regression). Its accuracy depends on the performance of the region proposal module (see comparisons in [20]). Several papers have proposed ways of using deep networks for predicting object bounding boxes [25], [9], [26], [27]. In the OverFeat method [9], a fully-connected layer is trained to predict the box coordinates for the localization task that assumes a single object. The fully-connected layer is then turned

1. Since the publication of the conference version of this paper [10], we have also found that RPNs can be trained jointly with Fast R-CNN networks leading to less training time.

2. http://image-net.org/challenges/LSVRC/2015/results

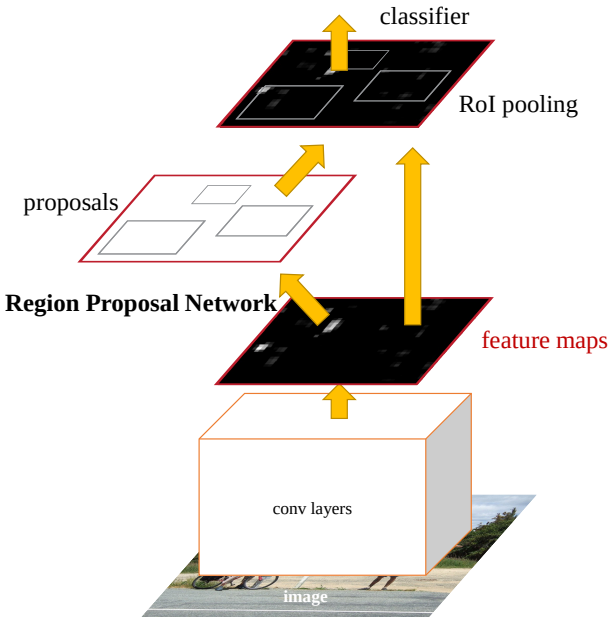


Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

into a convolutional layer for detecting multiple class-specific objects. The MultiBox methods [26], [27] generate region proposals from a network whose last fully-connected layer simultaneously predicts multiple class-agnostic boxes, generalizing the "single-box" fashion of OverFeat. These class-agnostic boxes are used as proposals for R-CNN [5]. The MultiBox proposal network is applied on a single image crop or multiple large image crops (*e.g.*, 224×224), in contrast to our fully convolutional scheme. MultiBox does not share features between the proposal and detection networks. We discuss OverFeat and MultiBox in more depth later in context with our method. Concurrent with our work, the DeepMask method [28] is developed for learning segmentation proposals.

Shared computation of convolutions [9], [1], [29], [7], [2] has been attracting increasing attention for efficient, yet accurate, visual recognition. The OverFeat paper [9] computes convolutional features from an image pyramid for classification, localization, and detection. Adaptively-sized pooling (SPP) [1] on shared convolutional feature maps is developed for efficient region-based object detection [1], [30] and semantic segmentation [29]. Fast R-CNN [2] enables end-to-end detector training on shared convolutional features and shows compelling accuracy and speed.

3 Faster R-CNN

Our object detection system, called Faster R-CNN, is composed of two modules. The first module is a deep fully convolutional network that proposes regions, and the second module is the Fast R-CNN detector [2] that uses the proposed regions. The entire system is a single, unified network for object detection (Figure 2). Using the recently popular terminology of neural networks with 'attention' [31] mechanisms, the RPN module tells the Fast R-CNN module where to look. In Section 3.1 we introduce the designs and properties of the network for region proposal. In Section 3.2 we develop algorithms for training both modules with features shared.

3.1 Region Proposal Networks

A Region Proposal Network (RPN) takes an image (of any size) as input and outputs a set of rectangular object proposals, each with an objectness score.[3] We model this process with a fully convolutional network [7], which we describe in this section. Because our ultimate goal is to share computation with a Fast R-CNN object detection network [2], we assume that both nets share a common set of convolutional layers. In our experiments, we investigate the Zeiler and Fergus model [32] (ZF), which has 5 shareable convolutional layers and the Simonyan and Zisserman model [3] (VGG-16), which has 13 shareable convolutional layers.

To generate region proposals, we slide a small network over the convolutional feature map output by the last shared convolutional layer. This small network takes as input an $n \times n$ spatial window of the input convolutional feature map. Each sliding window is mapped to a lower-dimensional feature (256-d for ZF and 512-d for VGG, with ReLU [33] following). This feature is fed into two sibling fully-connected layers—a box-regression layer (*reg*) and a box-classification layer (*cls*). We use $n = 3$ in this paper, noting that the effective receptive field on the input image is large (171 and 228 pixels for ZF and VGG, respectively). This mini-network is illustrated at a single position in Figure 3 (left). Note that because the mini-network operates in a sliding-window fashion, the fully-connected layers are shared across all spatial locations. This architecture is naturally implemented with an $n \times n$ convolutional layer followed by two sibling 1×1 convolutional layers (for *reg* and *cls*, respectively).

3.1.1 Anchors

At each sliding-window location, we simultaneously predict multiple region proposals, where the number of maximum possible proposals for each location is denoted as k. So the *reg* layer has $4k$ outputs encoding the coordinates of k boxes, and the *cls* layer outputs $2k$ scores that estimate probability of object or not object for each proposal[4]. The k proposals are parameterized *relative* to k reference boxes, which we call

3. "Region" is a generic term and in this paper we only consider *rectangular* regions, as is common for many methods (*e.g.*, [27], [4], [6]). "Objectness" measures membership to a set of object classes *vs.* background.

4. For simplicity we implement the *cls* layer as a two-class softmax layer. Alternatively, one may use logistic regression to produce k scores.

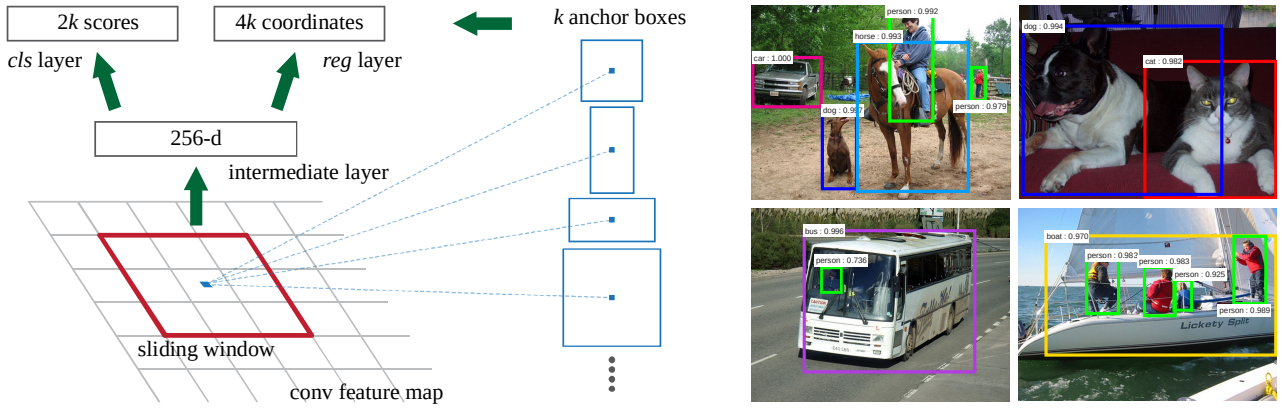


Figure 3: **Left**: Region Proposal Network (RPN). **Right**: Example detections using RPN proposals on PASCAL VOC 2007 test. Our method detects objects in a wide range of scales and aspect ratios.

anchors. An anchor is centered at the sliding window in question, and is associated with a scale and aspect ratio (Figure 3, left). By default we use 3 scales and 3 aspect ratios, yielding $k = 9$ anchors at each sliding position. For a convolutional feature map of a size $W \times H$ (typically ~2,400), there are WHk anchors in total.

Translation-Invariant Anchors

An important property of our approach is that it is *translation invariant*, both in terms of the anchors and the functions that compute proposals relative to the anchors. If one translates an object in an image, the proposal should translate and the same function should be able to predict the proposal in either location. This translation-invariant property is guaranteed by our method[5]. As a comparison, the MultiBox method [27] uses k-means to generate 800 anchors, which are *not* translation invariant. So MultiBox does not guarantee that the same proposal is generated if an object is translated.

The translation-invariant property also reduces the model size. MultiBox has a $(4+1) \times 800$-dimensional fully-connected output layer, whereas our method has a $(4+2) \times 9$-dimensional convolutional output layer in the case of $k = 9$ anchors. As a result, our output layer has 2.8×10^4 parameters ($512 \times (4+2) \times 9$ for VGG-16), two orders of magnitude fewer than MultiBox's output layer that has 6.1×10^6 parameters ($1536 \times (4+1) \times 800$ for GoogleNet [34] in MultiBox [27]). If considering the feature projection layers, our proposal layers still have an order of magnitude fewer parameters than MultiBox[6]. We expect our method to have less risk of overfitting on small datasets, like PASCAL VOC.

5. As is the case of FCNs [7], our network is translation invariant up to the network's total stride.

6. Considering the feature projection layers, our proposal layers' parameter count is $3 \times 3 \times 512 \times 512 + 512 \times 6 \times 9 = 2.4 \times 10^6$; MultiBox's proposal layers' parameter count is $7 \times 7 \times (64+96+64+64) \times 1536 + 1536 \times 5 \times 800 = 27 \times 10^6$.

Multi-Scale Anchors as Regression References

Our design of anchors presents a novel scheme for addressing multiple scales (and aspect ratios). As shown in Figure 1, there have been two popular ways for multi-scale predictions. The first way is based on image/feature pyramids, *e.g.*, in DPM [8] and CNN-based methods [9], [1], [2]. The images are resized at multiple scales, and feature maps (HOG [8] or deep convolutional features [9], [1], [2]) are computed for each scale (Figure 1(a)). This way is often useful but is time-consuming. The second way is to use sliding windows of multiple scales (and/or aspect ratios) on the feature maps. For example, in DPM [8], models of different aspect ratios are trained separately using different filter sizes (such as 5×7 and 7×5). If this way is used to address multiple scales, it can be thought of as a "pyramid of filters" (Figure 1(b)). The second way is usually adopted jointly with the first way [8].

As a comparison, our anchor-based method is built on *a pyramid of anchors*, which is more cost-efficient. Our method classifies and regresses bounding boxes with reference to anchor boxes of multiple scales and aspect ratios. It only relies on images and feature maps of a single scale, and uses filters (sliding windows on the feature map) of a single size. We show by experiments the effects of this scheme for addressing multiple scales and sizes (Table 8).

Because of this multi-scale design based on anchors, we can simply use the convolutional features computed on a single-scale image, as is also done by the Fast R-CNN detector [2]. The design of multi-scale anchors is a key component for sharing features without extra cost for addressing scales.

3.1.2 Loss Function

For training RPNs, we assign a binary class label (of being an object or not) to each anchor. We assign a positive label to two kinds of anchors: (i) the anchor/anchors with the highest Intersection-over-Union (IoU) overlap with a ground-truth box, *or* (ii) an anchor that has an IoU overlap higher than 0.7 with

any ground-truth box. Note that a single ground-truth box may assign positive labels to multiple anchors. Usually the second condition is sufficient to determine the positive samples; but we still adopt the first condition for the reason that in some rare cases the second condition may find no positive sample. We assign a negative label to a non-positive anchor if its IoU ratio is lower than 0.3 for all ground-truth boxes. Anchors that are neither positive nor negative do not contribute to the training objective.

With these definitions, we minimize an objective function following the multi-task loss in Fast R-CNN [2]. Our loss function for an image is defined as:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i, p_i^*) + \lambda\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i, t_i^*). \quad (1)$$

Here, i is the index of an anchor in a mini-batch and p_i is the predicted probability of anchor i being an object. The ground-truth label p_i^* is 1 if the anchor is positive, and is 0 if the anchor is negative. t_i is a vector representing the 4 parameterized coordinates of the predicted bounding box, and t_i^* is that of the ground-truth box associated with a positive anchor. The classification loss L_{cls} is log loss over two classes (object *vs*. not object). For the regression loss, we use $L_{reg}(t_i, t_i^*) = R(t_i - t_i^*)$ where R is the robust loss function (smooth L_1) defined in [2]. The term $p_i^* L_{reg}$ means the regression loss is activated only for positive anchors ($p_i^* = 1$) and is disabled otherwise ($p_i^* = 0$). The outputs of the *cls* and *reg* layers consist of $\{p_i\}$ and $\{t_i\}$ respectively.

The two terms are normalized by N_{cls} and N_{reg} and weighted by a balancing parameter λ. In our current implementation (as in the released code), the cls term in Eqn.(1) is normalized by the mini-batch size (*i.e.*, $N_{cls} = 256$) and the reg term is normalized by the number of anchor locations (*i.e.*, $N_{reg} \sim 2,400$). By default we set $\lambda = 10$, and thus both *cls* and *reg* terms are roughly equally weighted. We show by experiments that the results are insensitive to the values of λ in a wide range (Table 9). We also note that the normalization as above is not required and could be simplified.

For bounding box regression, we adopt the parameterizations of the 4 coordinates following [5]:

$$\begin{aligned} t_{\mathrm{x}} &= (x - x_{\mathrm{a}})/w_{\mathrm{a}}, \quad t_{\mathrm{y}} = (y - y_{\mathrm{a}})/h_{\mathrm{a}}, \\ t_{\mathrm{w}} &= \log(w/w_{\mathrm{a}}), \quad t_{\mathrm{h}} = \log(h/h_{\mathrm{a}}), \\ t_{\mathrm{x}}^* &= (x^* - x_{\mathrm{a}})/w_{\mathrm{a}}, \quad t_{\mathrm{y}}^* = (y^* - y_{\mathrm{a}})/h_{\mathrm{a}}, \\ t_{\mathrm{w}}^* &= \log(w^*/w_{\mathrm{a}}), \quad t_{\mathrm{h}}^* = \log(h^*/h_{\mathrm{a}}), \end{aligned} \quad (2)$$

where x, y, w, and h denote the box's center coordinates and its width and height. Variables x, x_{a}, and x^* are for the predicted box, anchor box, and ground-truth box respectively (likewise for y, w, h). This can be thought of as bounding-box regression from an anchor box to a nearby ground-truth box.

Nevertheless, our method achieves bounding-box regression by a different manner from previous RoI-based (Region of Interest) methods [1], [2]. In [1], [2], bounding-box regression is performed on features pooled from *arbitrarily* sized RoIs, and the regression weights are *shared* by all region sizes. In our formulation, the features used for regression are of the *same* spatial size (3×3) on the feature maps. To account for varying sizes, a set of k bounding-box regressors are learned. Each regressor is responsible for one scale and one aspect ratio, and the k regressors do *not* share weights. As such, it is still possible to predict boxes of various sizes even though the features are of a fixed size/scale, thanks to the design of anchors.

3.1.3 Training RPNs

The RPN can be trained end-to-end by back-propagation and stochastic gradient descent (SGD) [35]. We follow the "image-centric" sampling strategy from [2] to train this network. Each mini-batch arises from a single image that contains many positive and negative example anchors. It is possible to optimize for the loss functions of all anchors, but this will bias towards negative samples as they are dominate. Instead, we randomly sample 256 anchors in an image to compute the loss function of a mini-batch, where the sampled positive and negative anchors have a ratio of *up to* 1:1. If there are fewer than 128 positive samples in an image, we pad the mini-batch with negative ones.

We randomly initialize all new layers by drawing weights from a zero-mean Gaussian distribution with standard deviation 0.01. All other layers (*i.e.*, the shared convolutional layers) are initialized by pre-training a model for ImageNet classification [36], as is standard practice [5]. We tune all layers of the ZF net, and conv3_1 and up for the VGG net to conserve memory [2]. We use a learning rate of 0.001 for 60k mini-batches, and 0.0001 for the next 20k mini-batches on the PASCAL VOC dataset. We use a momentum of 0.9 and a weight decay of 0.0005 [37]. Our implementation uses Caffe [38].

3.2 Sharing Features for RPN and Fast R-CNN

Thus far we have described how to train a network for region proposal generation, without considering the region-based object detection CNN that will utilize these proposals. For the detection network, we adopt Fast R-CNN [2]. Next we describe algorithms that learn a unified network composed of RPN and Fast R-CNN with shared convolutional layers (Figure 2).

Both RPN and Fast R-CNN, trained independently, will modify their convolutional layers in different ways. We therefore need to develop a technique that allows for sharing convolutional layers between the

Table 1: the learned average proposal size for each anchor using the ZF net (numbers for $s = 600$).

anchor	128^2, 2:1	128^2, 1:1	128^2, 1:2	256^2, 2:1	256^2, 1:1	256^2, 1:2	512^2, 2:1	512^2, 1:1	512^2, 1:2
proposal	188×111	113×114	70×92	416×229	261×284	174×332	768×437	499×501	355×715

two networks, rather than learning two separate networks. We discuss three ways for training networks with features shared:

(i) *Alternating training*. In this solution, we first train RPN, and use the proposals to train Fast R-CNN. The network tuned by Fast R-CNN is then used to initialize RPN, and this process is iterated. This is the solution that is used in all experiments in this paper.

(ii) *Approximate joint training*. In this solution, the RPN and Fast R-CNN networks are merged into one network during training as in Figure 2. In each SGD iteration, the forward pass generates region proposals which are treated just like fixed, pre-computed proposals when training a Fast R-CNN detector. The backward propagation takes place as usual, where for the shared layers the backward propagated signals from both the RPN loss and the Fast R-CNN loss are combined. This solution is easy to implement. But this solution ignores the derivative w.r.t. the proposal boxes' coordinates that are also network responses, so is approximate. In our experiments, we have empirically found this solver produces close results, yet reduces the training time by about 25-50% comparing with alternating training. This solver is included in our released Python code.

(iii) *Non-approximate joint training*. As discussed above, the bounding boxes predicted by RPN are also functions of the input. The RoI pooling layer [2] in Fast R-CNN accepts the convolutional features and also the predicted bounding boxes as input, so a theoretically valid backpropagation solver should also involve gradients w.r.t. the box coordinates. These gradients are ignored in the above approximate joint training. In a non-approximate joint training solution, we need an RoI pooling layer that is differentiable w.r.t. the box coordinates. This is a nontrivial problem and a solution can be given by an "RoI warping" layer as developed in [15], which is beyond the scope of this paper.

4-Step Alternating Training. In this paper, we adopt a pragmatic 4-step training algorithm to learn shared features via alternating optimization. In the first step, we train the RPN as described in Section 3.1.3. This network is initialized with an ImageNet-pre-trained model and fine-tuned end-to-end for the region proposal task. In the second step, we train a separate detection network by Fast R-CNN using the proposals generated by the step-1 RPN. This detection network is also initialized by the ImageNet-pre-trained model. At this point the two networks do not share convolutional layers. In the third step, we use the detector network to initialize RPN training, but we fix the shared convolutional layers and only fine-tune the layers unique to RPN. Now the two networks share convolutional layers. Finally, keeping the shared convolutional layers fixed, we fine-tune the unique layers of Fast R-CNN. As such, both networks share the same convolutional layers and form a unified network. A similar alternating training can be run for more iterations, but we have observed negligible improvements.

3.3 Implementation Details

We train and test both region proposal and object detection networks on images of a single scale [1], [2]. We re-scale the images such that their shorter side is $s = 600$ pixels [2]. Multi-scale feature extraction (using an image pyramid) may improve accuracy but does not exhibit a good speed-accuracy trade-off [2]. On the re-scaled images, the total stride for both ZF and VGG nets on the last convolutional layer is 16 pixels, and thus is ∼10 pixels on a typical PASCAL image before resizing (∼500×375). Even such a large stride provides good results, though accuracy may be further improved with a smaller stride.

For anchors, we use 3 scales with box areas of 128^2, 256^2, and 512^2 pixels, and 3 aspect ratios of 1:1, 1:2, and 2:1. These hyper-parameters are *not* carefully chosen for a particular dataset, and we provide ablation experiments on their effects in the next section. As discussed, our solution does not need an image pyramid or filter pyramid to predict regions of multiple scales, saving considerable running time. Figure 3 (right) shows the capability of our method for a wide range of scales and aspect ratios. Table 1 shows the learned average proposal size for each anchor using the ZF net. We note that our algorithm allows predictions that are larger than the underlying receptive field. Such predictions are not impossible—one may still roughly infer the extent of an object if only the middle of the object is visible.

The anchor boxes that cross image boundaries need to be handled with care. During training, we ignore all cross-boundary anchors so they do not contribute to the loss. For a typical 1000×600 image, there will be roughly 20000 ($\approx 60 \times 40 \times 9$) anchors in total. With the cross-boundary anchors ignored, there are about 6000 anchors per image for training. If the boundary-crossing outliers are not ignored in training, they introduce large, difficult to correct error terms in the objective, and training does not converge. During testing, however, we still apply the fully convolutional RPN to the entire image. This may generate cross-boundary proposal boxes, which we clip to the image boundary.

Table 2: Detection results on **PASCAL VOC 2007 test set** (trained on VOC 2007 trainval). The detectors are Fast R-CNN with ZF, but using various proposal methods for training and testing.

train-time region proposals		**test-time region proposals**		
method	# boxes	method	# proposals	mAP (%)
SS	2000	SS	2000	58.7
EB	2000	EB	2000	58.6
RPN+ZF, shared	2000	RPN+ZF, shared	300	**59.9**
ablation experiments follow below				
RPN+ZF, unshared	2000	RPN+ZF, unshared	300	58.7
SS	2000	RPN+ZF	100	55.1
SS	2000	RPN+ZF	300	56.8
SS	2000	RPN+ZF	1000	56.3
SS	2000	RPN+ZF (no NMS)	6000	55.2
SS	2000	RPN+ZF (no *cls*)	100	44.6
SS	2000	RPN+ZF (no *cls*)	300	51.4
SS	2000	RPN+ZF (no *cls*)	1000	55.8
SS	2000	RPN+ZF (no *reg*)	300	52.1
SS	2000	RPN+ZF (no *reg*)	1000	51.3
SS	2000	RPN+VGG	300	59.2

Some RPN proposals highly overlap with each other. To reduce redundancy, we adopt non-maximum suppression (NMS) on the proposal regions based on their *cls* scores. We fix the IoU threshold for NMS at 0.7, which leaves us about 2000 proposal regions per image. As we will show, NMS does not harm the ultimate detection accuracy, but substantially reduces the number of proposals. After NMS, we use the top-N ranked proposal regions for detection. In the following, we train Fast R-CNN using 2000 RPN proposals, but evaluate different numbers of proposals at test-time.

4 Experiments

4.1 Experiments on PASCAL VOC

We comprehensively evaluate our method on the PASCAL VOC 2007 detection benchmark [11]. This dataset consists of about 5k trainval images and 5k test images over 20 object categories. We also provide results on the PASCAL VOC 2012 benchmark for a few models. For the ImageNet pre-trained network, we use the "fast" version of ZF net [32] that has 5 convolutional layers and 3 fully-connected layers, and the public VGG-16 model[7] [3] that has 13 convolutional layers and 3 fully-connected layers. We primarily evaluate detection mean Average Precision (mAP), because this is the actual metric for object detection (rather than focusing on object proposal proxy metrics).

Table 2 (top) shows Fast R-CNN results when trained and tested using various region proposal methods. These results use the ZF net. For Selective Search (SS) [4], we generate about 2000 proposals by the "fast" mode. For EdgeBoxes (EB) [6], we generate the proposals by the default EB setting tuned for 0.7 IoU. SS has an mAP of 58.7% and EB has an mAP of 58.6% under the Fast R-CNN framework. RPN with Fast R-CNN achieves competitive results, with an mAP of 59.9% while using *up to* 300 proposals[8]. Using RPN yields a much faster detection system than using either SS or EB because of shared convolutional computations; the fewer proposals also reduce the region-wise fully-connected layers' cost (Table 5).

Ablation Experiments on RPN. To investigate the behavior of RPNs as a proposal method, we conducted several ablation studies. First, we show the effect of sharing convolutional layers between the RPN and Fast R-CNN detection network. To do this, we stop after the second step in the 4-step training process. Using separate networks reduces the result slightly to 58.7% (RPN+ZF, unshared, Table 2). We observe that this is because in the third step when the detector-tuned features are used to fine-tune the RPN, the proposal quality is improved.

Next, we disentangle the RPN's influence on training the Fast R-CNN detection network. For this purpose, we train a Fast R-CNN model by using the 2000 SS proposals and ZF net. We fix this detector and evaluate the detection mAP by changing the proposal regions used at test-time. In these ablation experiments, the RPN does not share features with the detector.

Replacing SS with 300 RPN proposals at test-time leads to an mAP of 56.8%. The loss in mAP is because of the inconsistency between the training/testing proposals. This result serves as the baseline for the following comparisons.

Somewhat surprisingly, the RPN still leads to a competitive result (55.1%) when using the top-ranked

7. www.robots.ox.ac.uk/~vgg/research/very_deep/

8. For RPN, the number of proposals (*e.g.*, 300) is the maximum number for an image. RPN may produce fewer proposals after NMS, and thus the average number of proposals is smaller.

Table 3: Detection results on **PASCAL VOC 2007 test set**. The detector is Fast R-CNN and VGG-16. Training data: "07": VOC 2007 trainval, "07+12": union set of VOC 2007 trainval and VOC 2012 trainval. For RPN, the train-time proposals for Fast R-CNN are 2000. †: this number was reported in [2]; using the repository provided by this paper, this result is higher (68.1).

method	# proposals	data	mAP (%)
SS	2000	07	66.9†
SS	2000	07+12	70.0
RPN+VGG, unshared	300	07	68.5
RPN+VGG, shared	300	07	69.9
RPN+VGG, shared	300	07+12	**73.2**
RPN+VGG, shared	300	COCO+07+12	**78.8**

Table 4: Detection results on **PASCAL VOC 2012 test set**. The detector is Fast R-CNN and VGG-16. Training data: "07": VOC 2007 trainval, "07++12": union set of VOC 2007 trainval+test and VOC 2012 trainval. For RPN, the train-time proposals for Fast R-CNN are 2000. †: http://host.robots.ox.ac.uk:8080/anonymous/HZJTQA.html. ‡: http://host.robots.ox.ac.uk:8080/anonymous/YNPLXB.html. §: http://host.robots.ox.ac.uk:8080/anonymous/XEDH10.html.

method	# proposals	data	mAP (%)
SS	2000	12	65.7
SS	2000	07++12	68.4
RPN+VGG, shared†	300	12	67.0
RPN+VGG, shared‡	300	07++12	**70.4**
RPN+VGG, shared§	300	COCO+07++12	**75.9**

Table 5: **Timing** (ms) on a K40 GPU, except SS proposal is evaluated in a CPU. "Region-wise" includes NMS, pooling, fully-connected, and softmax layers. See our released code for the profiling of running time.

model	system	conv	proposal	region-wise	total	rate
VGG	SS + Fast R-CNN	146	1510	174	1830	0.5 fps
VGG	RPN + Fast R-CNN	141	**10**	47	**198**	**5 fps**
ZF	RPN + Fast R-CNN	31	**3**	25	**59**	**17 fps**

100 proposals at test-time, indicating that the top-ranked RPN proposals are accurate. On the other extreme, using the top-ranked 6000 RPN proposals (without NMS) has a comparable mAP (55.2%), suggesting NMS does not harm the detection mAP and may reduce false alarms.

Next, we separately investigate the roles of RPN's *cls* and *reg* outputs by turning off either of them at test-time. When the *cls* layer is removed at test-time (thus no NMS/ranking is used), we randomly sample N proposals from the unscored regions. The mAP is nearly unchanged with $N = 1000$ (55.8%), but degrades considerably to 44.6% when $N = 100$. This shows that the *cls* scores account for the accuracy of the highest ranked proposals.

On the other hand, when the *reg* layer is removed at test-time (so the proposals become anchor boxes), the mAP drops to 52.1%. This suggests that the high-quality proposals are mainly due to the regressed box bounds. The anchor boxes, though having multiple scales and aspect ratios, are not sufficient for accurate detection.

We also evaluate the effects of more powerful networks on the proposal quality of RPN alone. We use VGG-16 to train the RPN, and still use the above detector of SS+ZF. The mAP improves from 56.8% (using RPN+ZF) to 59.2% (using RPN+VGG). This is a promising result, because it suggests that the proposal quality of RPN+VGG is better than that of RPN+ZF. Because proposals of RPN+ZF are competitive with SS (both are 58.7% when consistently used for training and testing), we may expect RPN+VGG to be better than SS. The following experiments justify this hypothesis.

Performance of VGG-16. Table 3 shows the results of VGG-16 for both proposal and detection. Using RPN+VGG, the result is 68.5% for *unshared* features, slightly higher than the SS baseline. As shown above, this is because the proposals generated by RPN+VGG are more accurate than SS. Unlike SS that is pre-defined, the RPN is actively trained and benefits from better networks. For the feature-*shared* variant, the result is 69.9%—better than the strong SS baseline, yet with nearly cost-free proposals. We further train the RPN and detection network on the union set of PASCAL VOC 2007 trainval and 2012 trainval. The mAP is **73.2%**. Figure 5 shows some results on the PASCAL VOC 2007 test set. On the PASCAL VOC 2012 test set (Table 4), our method has an mAP of **70.4%** trained on the union set of VOC 2007 trainval+test and VOC 2012 trainval. Table 6 and Table 7 show the detailed numbers.

Table 6: Results on PASCAL VOC 2007 test set with Fast R-CNN detectors and VGG-16. For RPN, the train-time proposals for Fast R-CNN are 2000. RPN* denotes the unsharing feature version.

method	# box	data	mAP	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
SS	2000	07	66.9	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8
SS	2000	07+12	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	76.6	69.9	31.8	70.1	74.8	80.4	70.4
RPN*	300	07	68.5	74.1	77.2	67.7	53.9	51.0	75.1	79.2	78.9	50.7	78.0	61.1	79.1	81.9	72.2	75.9	37.2	71.4	62.5	77.4	66.4
RPN	300	07	69.9	70.0	80.6	70.1	57.3	49.9	78.2	80.4	82.0	52.2	75.3	67.2	80.3	79.8	75.0	76.3	39.1	68.3	67.3	81.1	67.6
RPN	300	07+12	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
RPN	300	COCO+07+12	**78.8**	**84.3**	**82.0**	**77.7**	**68.9**	**65.7**	**88.1**	**88.4**	**88.9**	**63.6**	**86.3**	**70.8**	**85.9**	**87.6**	**80.1**	**82.3**	**53.6**	**80.4**	**75.8**	**86.6**	**78.9**

Table 7: Results on PASCAL VOC 2012 test set with Fast R-CNN detectors and VGG-16. For RPN, the train-time proposals for Fast R-CNN are 2000.

method	# box	data	mAP	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
SS	2000	12	65.7	80.3	74.7	66.9	46.9	37.7	73.9	68.6	87.7	41.7	71.1	51.1	86.0	77.8	79.8	69.8	32.1	65.5	63.8	76.4	61.7
SS	2000	07++12	68.4	82.3	78.4	70.8	52.3	38.7	77.8	71.6	89.3	44.2	73.0	55.0	**87.5**	80.5	80.8	72.0	35.1	68.3	**65.7**	80.4	64.2
RPN	300	12	67.0	82.3	76.4	71.0	48.4	45.2	72.1	72.3	87.3	42.2	73.7	50.0	86.8	78.7	78.4	77.4	34.5	70.1	57.1	77.1	58.9
RPN	300	07++12	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
RPN	300	COCO+07++12	**75.9**	**87.4**	**83.6**	**76.8**	**62.9**	**59.6**	**81.9**	**82.0**	**91.3**	**54.9**	**82.6**	**59.0**	**89.0**	**85.5**	**84.7**	**84.1**	**52.2**	**78.9**	65.5	**85.4**	**70.2**

Table 8: Detection results of Faster R-CNN on PASCAL VOC 2007 test set using **different settings of anchors**. The network is VGG-16. The training data is VOC 2007 trainval. The default setting of using 3 scales and 3 aspect ratios (69.9%) is the same as that in Table 3.

settings	anchor scales	aspect ratios	mAP (%)
1 scale, 1 ratio	128^2	1:1	65.8
	256^2	1:1	66.7
1 scale, 3 ratios	128^2	{2:1, 1:1, 1:2}	68.8
	256^2	{2:1, 1:1, 1:2}	67.9
3 scales, 1 ratio	$\{128^2, 256^2, 512^2\}$	1:1	**69.8**
3 scales, 3 ratios	$\{128^2, 256^2, 512^2\}$	{2:1, 1:1, 1:2}	**69.9**

Table 9: Detection results of Faster R-CNN on PASCAL VOC 2007 test set using **different values of** λ in Equation (1). The network is VGG-16. The training data is VOC 2007 trainval. The default setting of using $\lambda = 10$ (69.9%) is the same as that in Table 3.

λ	0.1	1	10	100
mAP (%)	67.2	68.9	69.9	69.1

In Table 5 we summarize the running time of the entire object detection system. SS takes 1-2 seconds depending on content (on average about 1.5s), and Fast R-CNN with VGG-16 takes 320ms on 2000 SS proposals (or 223ms if using SVD on fully-connected layers [2]). Our system with VGG-16 takes in total **198ms** for both proposal and detection. With the convolutional features shared, the RPN alone only takes 10ms computing the additional layers. Our region-wise computation is also lower, thanks to fewer proposals (300 per image). Our system has a frame-rate of 17 fps with the ZF net.

Sensitivities to Hyper-parameters. In Table 8 we investigate the settings of anchors. By default we use 3 scales and 3 aspect ratios (69.9% mAP in Table 8). If using just one anchor at each position, the mAP drops by a considerable margin of 3-4%. The mAP is higher if using 3 scales (with 1 aspect ratio) or 3 aspect ratios (with 1 scale), demonstrating that using anchors of multiple sizes as the regression references is an effective solution. Using just 3 scales with 1 aspect ratio (69.8%) is as good as using 3 scales with 3 aspect ratios on this dataset, suggesting that scales and aspect ratios are not disentangled dimensions for the detection accuracy. But we still adopt these two dimensions in our designs to keep our system flexible.

In Table 9 we compare different values of λ in Equation (1). By default we use $\lambda = 10$ which makes the two terms in Equation (1) roughly equally weighted after normalization. Table 9 shows that our result is impacted just marginally (by $\sim 1\%$) when λ is within a scale of about two orders of magnitude (1 to 100). This demonstrates that the result is insensitive to λ in a wide range.

Analysis of Recall-to-IoU. Next we compute the recall of proposals at different IoU ratios with ground-truth boxes. It is noteworthy that the Recall-to-IoU metric is just *loosely* [19], [20], [21] related to the ultimate detection accuracy. It is more appropriate to use this metric to *diagnose* the proposal method than to evaluate it.

In Figure 4, we show the results of using 300, 1000, and 2000 proposals. We compare with SS and EB, and the N proposals are the top-N ranked ones based on the confidence generated by these methods. The plots show that the RPN method behaves gracefully when the number of proposals drops from 2000 to 300. This explains why the RPN has a good ultimate detection mAP when using as few as 300 proposals. As we analyzed before, this property is mainly attributed to the *cls* term of the RPN. The recall of SS and EB drops more quickly than RPN when the proposals are fewer.

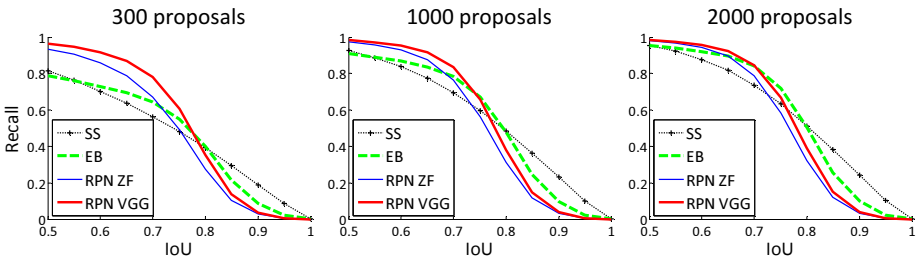


Figure 4: Recall *vs*. IoU overlap ratio on the PASCAL VOC 2007 test set.

Table 10: **One-Stage Detection *vs*. Two-Stage Proposal + Detection**. Detection results are on the PASCAL VOC 2007 test set using the ZF model and Fast R-CNN. RPN uses unshared features.

	proposals		detector	mAP (%)
Two-Stage	RPN + ZF, unshared	300	Fast R-CNN + ZF, 1 scale	58.7
One-Stage	dense, 3 scales, 3 aspect ratios	20000	Fast R-CNN + ZF, 1 scale	53.8
One-Stage	dense, 3 scales, 3 aspect ratios	20000	Fast R-CNN + ZF, 5 scales	53.9

One-Stage Detection *vs*. Two-Stage Proposal + Detection. The OverFeat paper [9] proposes a detection method that uses regressors and classifiers on sliding windows over convolutional feature maps. OverFeat is a *one-stage*, *class-specific* detection pipeline, and ours is a *two-stage cascade* consisting of class-agnostic proposals and class-specific detections. In OverFeat, the region-wise features come from a sliding window of one aspect ratio over a scale pyramid. These features are used to simultaneously determine the location and category of objects. In RPN, the features are from square (3×3) sliding windows and predict proposals relative to anchors with different scales and aspect ratios. Though both methods use sliding windows, the region proposal task is only the first stage of Faster R-CNN—the downstream Fast R-CNN detector *attends* to the proposals to refine them. In the second stage of our cascade, the region-wise features are adaptively pooled [1], [2] from proposal boxes that more faithfully cover the features of the regions. We believe these features lead to more accurate detections.

To compare the one-stage and two-stage systems, we *emulate* the OverFeat system (and thus also circumvent other differences of implementation details) by *one-stage* Fast R-CNN. In this system, the "proposals" are dense sliding windows of 3 scales (128, 256, 512) and 3 aspect ratios (1:1, 1:2, 2:1). Fast R-CNN is trained to predict class-specific scores and regress box locations from these sliding windows. Because the OverFeat system adopts an image pyramid, we also evaluate using convolutional features extracted from 5 scales. We use those 5 scales as in [1], [2].

Table 10 compares the two-stage system and two variants of the one-stage system. Using the ZF model, the one-stage system has an mAP of 53.9%. This is lower than the two-stage system (58.7%) by 4.8%. This experiment justifies the effectiveness of cascaded region proposals and object detection. Similar observations are reported in [2], [39], where replacing SS region proposals with sliding windows leads to ∼6% degradation in both papers. We also note that the one-stage system is slower as it has considerably more proposals to process.

4.2 Experiments on MS COCO

We present more results on the Microsoft COCO object detection dataset [12]. This dataset involves 80 object categories. We experiment with the 80k images on the training set, 40k images on the validation set, and 20k images on the test-dev set. We evaluate the mAP averaged for IoU $\in [0.5 : 0.05 : 0.95]$ (COCO's standard metric, simply denoted as mAP@[.5, .95]) and mAP@0.5 (PASCAL VOC's metric).

There are a few minor changes of our system made for this dataset. We train our models on an 8-GPU implementation, and the effective mini-batch size becomes 8 for RPN (1 per GPU) and 16 for Fast R-CNN (2 per GPU). The RPN step and Fast R-CNN step are both trained for 240k iterations with a learning rate of 0.003 and then for 80k iterations with 0.0003. We modify the learning rates (starting with 0.003 instead of 0.001) because the mini-batch size is changed. For the anchors, we use 3 aspect ratios and 4 scales (adding 64^2), mainly motivated by handling small objects on this dataset. In addition, in our Fast R-CNN step, the negative samples are defined as those with a maximum IoU with ground truth in the interval of $[0, 0.5)$, instead of $[0.1, 0.5)$ used in [1], [2]. We note that in the SPPnet system [1], the negative samples in $[0.1, 0.5)$ are used for network fine-tuning, but the negative samples in $[0, 0.5)$ are still visited in the SVM step with hard-negative mining. But the Fast R-CNN system [2] abandons the SVM step, so the negative samples in $[0, 0.1)$ are never visited. Including these $[0, 0.1)$ samples improves mAP@0.5 on the COCO dataset for both Fast R-CNN and Faster R-CNN systems (but the impact is negligible on PASCAL VOC).

Table 11: Object detection results (%) on the **MS COCO** dataset. The model is VGG-16.

method	proposals	training data	COCO val		COCO test-dev	
			mAP@.5	mAP@[.5, .95]	mAP@.5	mAP@[.5, .95]
Fast R-CNN [2]	SS, 2000	COCO train	-	-	35.9	19.7
Fast R-CNN [impl. in this paper]	SS, 2000	COCO train	38.6	18.9	39.3	19.3
Faster R-CNN	RPN, 300	COCO train	41.5	21.2	42.1	21.5
Faster R-CNN	RPN, 300	COCO trainval	-	-	**42.7**	**21.9**

The rest of the implementation details are the same as on PASCAL VOC. In particular, we keep using 300 proposals and single-scale ($s = 600$) testing. The testing time is still about 200ms per image on the COCO dataset.

In Table 11 we first report the results of the Fast R-CNN system [2] using the implementation in this paper. Our Fast R-CNN baseline has 39.3% mAP@0.5 on the test-dev set, higher than that reported in [2]. We conjecture that the reason for this gap is mainly due to the definition of the negative samples and also the changes of the mini-batch sizes. We also note that the mAP@[.5, .95] is just comparable.

Next we evaluate our Faster R-CNN system. Using the COCO training set to train, Faster R-CNN has 42.1% mAP@0.5 and 21.5% mAP@[.5, .95] on the COCO test-dev set. This is 2.8% higher for mAP@0.5 and **2.2% higher for mAP@[.5, .95]** than the Fast R-CNN counterpart under the same protocol (Table 11). This indicates that RPN performs excellent for improving the localization accuracy at higher IoU thresholds. Using the COCO trainval set to train, Faster R-CNN has 42.7% mAP@0.5 and 21.9% mAP@[.5, .95] on the COCO test-dev set. Figure 6 shows some results on the MS COCO test-dev set.

Faster R-CNN in ILSVRC & COCO 2015 competitions We have demonstrated that Faster R-CNN benefits more from better features, thanks to the fact that the RPN completely learns to propose regions by neural networks. This observation is still valid even when one increases the depth substantially to over 100 layers [18]. Only by replacing VGG-16 with a 101-layer residual net (ResNet-101) [18], the Faster R-CNN system increases the mAP from 41.5%/21.2% (VGG-16) to 48.4%/27.2% (ResNet-101) on the COCO val set. With other improvements orthogonal to Faster R-CNN, He *et al*. [18] obtained a single-model result of 55.7%/34.9% and an ensemble result of 59.0%/37.4% on the COCO test-dev set, which won the 1st place in the COCO 2015 object detection competition. The same system [18] also won the 1st place in the ILSVRC 2015 object detection competition, surpassing the second place by absolute 8.5%. RPN is also a building block of the 1st-place winning entries in ILSVRC 2015 localization and COCO 2015 segmentation competitions, for which the details are available in [18] and [15] respectively.

Table 12: Detection mAP (%) of Faster R-CNN on PASCAL VOC 2007 test set and 2012 test set using different training data. The model is VGG-16. "COCO" denotes that the COCO trainval set is used for training. See also Table 6 and Table 7.

training data	2007 test	2012 test
VOC07	69.9	67.0
VOC07+12	73.2	-
VOC07++12	-	70.4
COCO (no VOC)	76.1	73.0
COCO+VOC07+12	**78.8**	-
COCO+VOC07++12	-	**75.9**

4.3 From MS COCO to PASCAL VOC

Large-scale data is of crucial importance for improving deep neural networks. Next, we investigate how the MS COCO dataset can help with the detection performance on PASCAL VOC.

As a simple baseline, we directly evaluate the COCO detection model on the PASCAL VOC dataset, *without fine-tuning on any PASCAL VOC data*. This evaluation is possible because the categories on COCO are a superset of those on PASCAL VOC. The categories that are exclusive on COCO are ignored in this experiment, and the softmax layer is performed only on the 20 categories plus background. The mAP under this setting is 76.1% on the PASCAL VOC 2007 test set (Table 12). This result is better than that trained on VOC07+12 (73.2%) by a good margin, even though the PASCAL VOC data are not exploited.

Then we fine-tune the COCO detection model on the VOC dataset. In this experiment, the COCO model is in place of the ImageNet-pre-trained model (that is used to initialize the network weights), and the Faster R-CNN system is fine-tuned as described in Section 3.2. Doing so leads to 78.8% mAP on the PASCAL VOC 2007 test set. The extra data from the COCO set increases the mAP by 5.6%. Table 6 shows that the model trained on COCO+VOC has the best AP for every individual category on PASCAL VOC 2007. Similar improvements are observed on the PASCAL VOC 2012 test set (Table 12 and Table 7). We note that the test-time speed of obtaining these strong results is still about **200ms per image**.

5 Conclusion

We have presented RPNs for efficient and accurate region proposal generation. By sharing convolutional

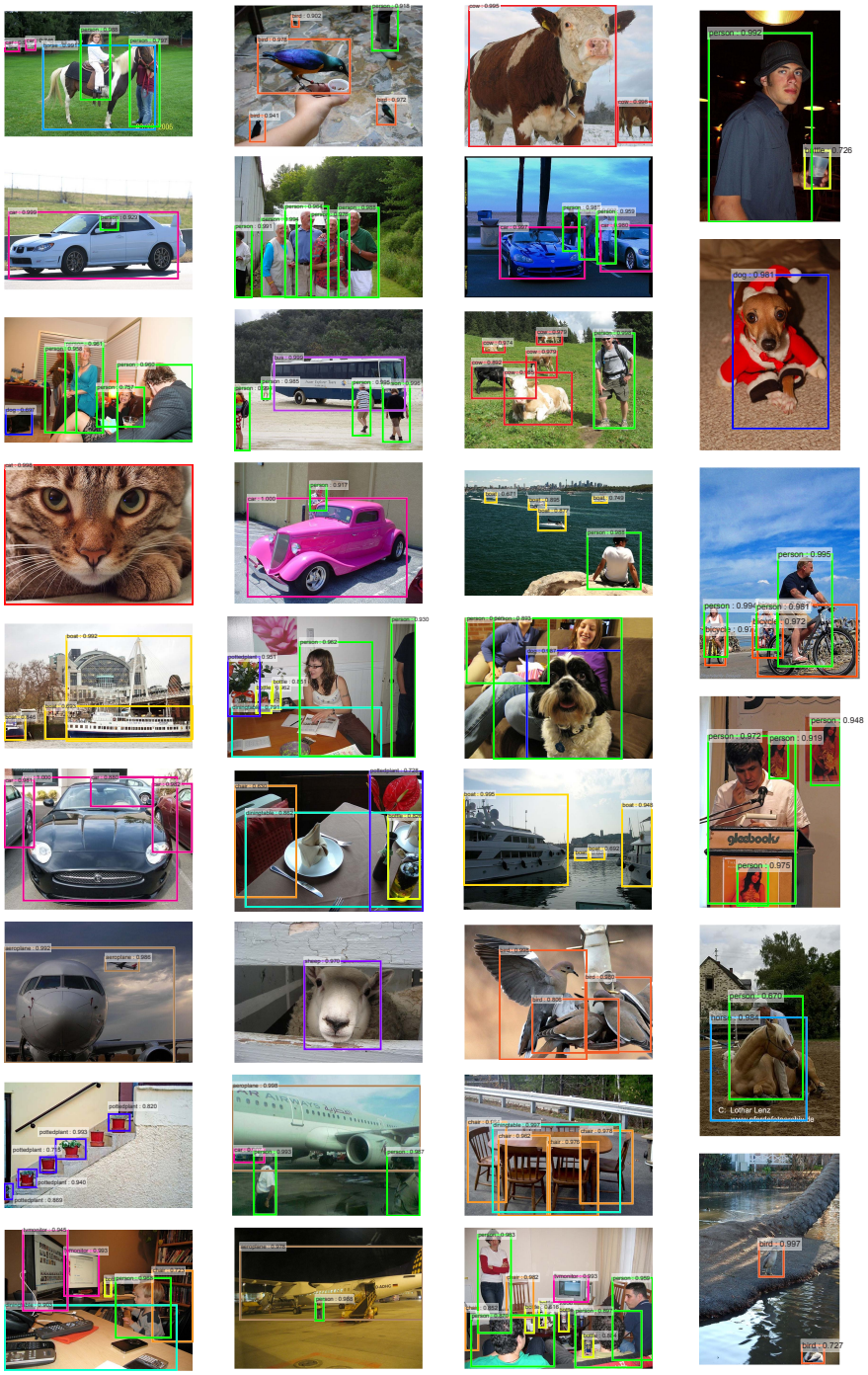


Figure 5: Selected examples of object detection results on the PASCAL VOC 2007 test set using the Faster R-CNN system. The model is VGG-16 and the training data is 07+12 trainval (73.2% mAP on the 2007 test set). Our method detects objects of a wide range of scales and aspect ratios. Each output box is associated with a category label and a softmax score in $[0, 1]$. A score threshold of 0.6 is used to display these images. The running time for obtaining these results is **198ms** per image, *including all steps*.

features with the down-stream detection network, the region proposal step is nearly cost-free. Our method enables a unified, deep-learning-based object detection system to run at near real-time frame rates. The learned RPN also improves region proposal quality and thus the overall object detection accuracy.

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *European Conference on Computer Vision (ECCV)*, 2014.

[2] R. Girshick, "Fast R-CNN," in *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional

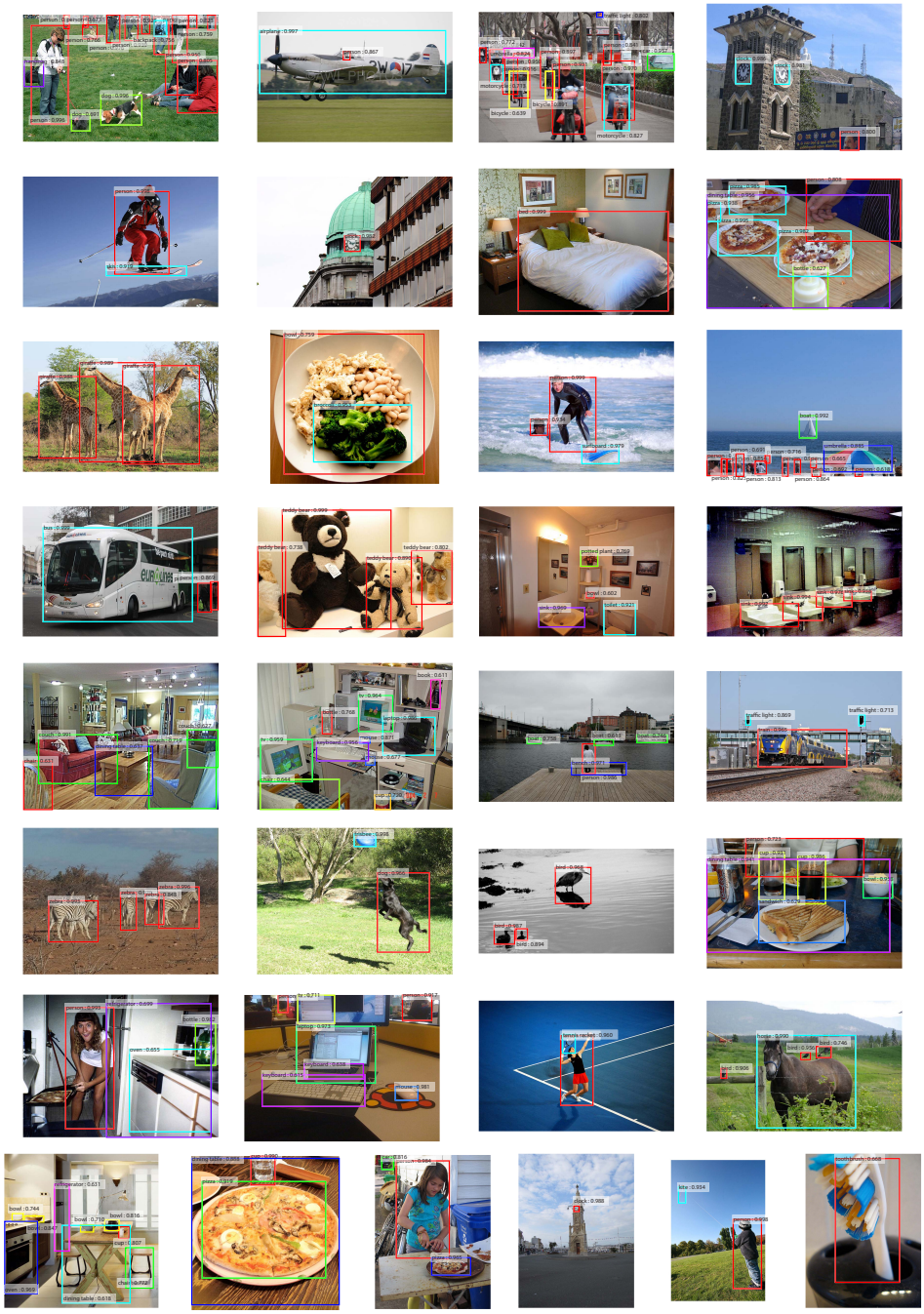


Figure 6: Selected examples of object detection results on the MS COCO test-dev set using the Faster R-CNN system. The model is VGG-16 and the training data is COCO trainval (42.7% mAP@0.5 on the test-dev set). Each output box is associated with a category label and a softmax score in $[0, 1]$. A score threshold of 0.6 is used to display these images. For each image, one color represents one object category in that image.

networks for large-scale image recognition," in *International Conference on Learning Representations (ICLR)*, 2015.

[4] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision (IJCV)*, 2013.

[5] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[6] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *European Conference on Computer Vision (ECCV)*, 2014.

[7] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[8] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2010.

[9] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2014.

[10] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards

real-time object detection with region proposal networks," in *Neural Information Processing Systems (NIPS)*, 2015.

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

[12] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *European Conference on Computer Vision (ECCV)*, 2014.

[13] S. Song and J. Xiao, "Deep sliding shapes for amodal 3d object detection in rgb-d images," *arXiv:1511.02300*, 2015.

[14] J. Zhu, X. Chen, and A. L. Yuille, "DeePM: A deep part-based model for object detection and semantic part localization," *arXiv:1511.07131*, 2015.

[15] J. Dai, K. He, and J. Sun, "Instance-aware semantic segmentation via multi-task network cascades," *arXiv:1512.04412*, 2015.

[16] J. Johnson, A. Karpathy, and L. Fei-Fei, "Densecap: Fully convolutional localization networks for dense captioning," *arXiv:1511.07571*, 2015.

[17] D. Kislyuk, Y. Liu, D. Liu, E. Tzeng, and Y. Jing, "Human curation and convnets: Powering item-to-item recommendations on pinterest," *arXiv:1511.04003*, 2015.

[18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv:1512.03385*, 2015.

[19] J. Hosang, R. Benenson, and B. Schiele, "How good are detection proposals, really?" in *British Machine Vision Conference (BMVC)*, 2014.

[20] J. Hosang, R. Benenson, P. Dollár, and B. Schiele, "What makes for effective detection proposals?" *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2015.

[21] N. Chavali, H. Agrawal, A. Mahendru, and D. Batra, "Object-Proposal Evaluation Protocol is 'Gameable'," *arXiv: 1505.05836*, 2015.

[22] J. Carreira and C. Sminchisescu, "CPMC: Automatic object segmentation using constrained parametric min-cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2012.

[23] P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik, "Multiscale combinatorial grouping," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[24] B. Alexe, T. Deselaers, and V. Ferrari, "Measuring the objectness of image windows," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2012.

[25] C. Szegedy, A. Toshev, and D. Erhan, "Deep neural networks for object detection," in *Neural Information Processing Systems (NIPS)*, 2013.

[26] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[27] C. Szegedy, S. Reed, D. Erhan, and D. Anguelov, "Scalable, high-quality object detection," *arXiv:1412.1441 (v1)*, 2015.

[28] P. O. Pinheiro, R. Collobert, and P. Dollar, "Learning to segment object candidates," in *Neural Information Processing Systems (NIPS)*, 2015.

[29] J. Dai, K. He, and J. Sun, "Convolutional feature masking for joint object and stuff segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[30] S. Ren, K. He, R. Girshick, X. Zhang, and J. Sun, "Object detection networks on convolutional feature maps," *arXiv:1504.06066*, 2015.

[31] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Neural Information Processing Systems (NIPS)*, 2015.

[32] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional neural networks," in *European Conference on Computer Vision (ECCV)*, 2014.

[33] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *International Conference on Machine Learning (ICML)*, 2010.

[34] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, and A. Rabinovich, "Going deeper with convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[35] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, 1989.

[36] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," in *International Journal of Computer Vision (IJCV)*, 2015.

[37] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Neural Information Processing Systems (NIPS)*, 2012.

[38] 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.

[39] K. Lenc and A. Vedaldi, "R-CNN minus R," in *British Machine Vision Conference (BMVC)*, 2015.