Rich feature hierarchies for accurate object detection and semantic segmentation Supplementary material

Ross Girshick Jeff Donahue Trevor Darrell Jitendra Malik
UC Berkeley
{rbg,jdonahue,trevor,malik}@eecs.berkeley.edu

1. Object proposal transformations

The convolutional neural network used in this work requires a fixed-size input of 227×227 pixels. For detection, we consider object proposals that are arbitrary image rectangles. We evaluated two approaches for transforming object proposals into valid CNN inputs.

The first method ("tightest square with context") encloses each object proposal inside the tightest square and then scales (isotropically) the image contained in that square to the CNN input size. Figure 1 column (B) shows this transformation. A variant on this method ("tightest square without context") excludes the image content that surrounds the original object proposal. Figure 1 column (C) shows this transformation. The second method ("warp") anisotropically scales each object proposal to the CNN input size. Figure 1 column (D) shows the warp transformation.

For each of these transformations, we also consider including additional image context around the original object proposal. The amount of context padding (p) is defined as a border size around the original object proposal in the transformed input coordinate frame. Figure 1 shows $p = 0$ pixels in the top row of each example and $p = 16$ pixels in the bottom row. In all methods, if the source rectangle extends beyond the image, the missing data is replaced with the image mean (which is then subtracted before inputing the image into the CNN). A pilot set of experiments showed that warping with context padding ($p = 16$ pixels) outperformed the alternatives by a large margin (3-5 mAP points). Obviously more alternatives are possible, including using replication instead of mean padding. Exhaustive evaluation of these alternatives is left as future work.

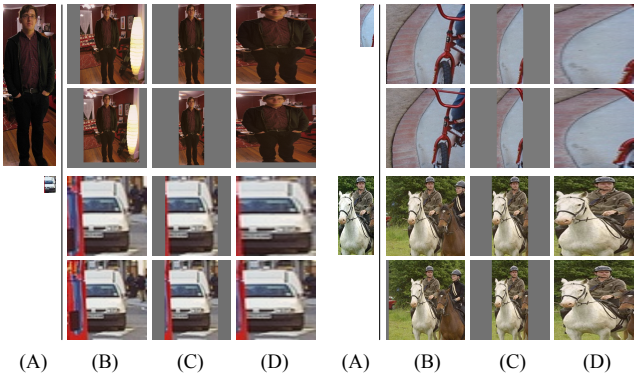


Figure 1: Different object proposal transformations. (A) the original object proposal at its actual scale relative to the transformed CNN inputs; (B) tightest square with context; (C) tightest square without context; (D) warp. Within each column and example proposal, the top row corresponds to $p = 0$ pixels of context padding while the bottom row has $p = 16$ pixels of context padding.

2. Positive *vs*. negative examples and softmax

Two design choices warrant further discussion. The first is: Why are positive and negative examples defined differently for fine-tuning the CNN versus training the object detection SVMs? To review the definitions briefly, for fine-tuning we map each object proposal to the ground-truth instance with which it has maximum IoU overlap (if any) and label it as a positive for the matched ground-truth class if the IoU is at least 0.5. All other proposals are labeled "background" (*i.e.*, negative examples for all classes). For training SVMs, in contrast, we take only the ground-truth boxes as positive examples for their respective classes and label proposals with less than 0.3 IoU overlap with all instances of a class as a negative for that class. Proposals that fall into the grey zone (more than 0.3 IoU overlap, but are not ground truth) are ignored.

Historically speaking, we arrived at these definitions because we started by training SVMs on features computed by the ImageNet pre-trained CNN, and so fine-tuning was not a consideration at that point in time. In that setup, we found that our particular label definition for training SVMs was optimal within the set of options we evaluated (which included the setting we now use for fine-tuning). When we started using fine-tuning, we initially used the same positive and negative example definition as we were using for SVM

training. However, we found that results were much worse than those obtained using our current definition of positives and negatives.

Our hypothesis is that this difference in how positives and negatives are defined is not fundamentally important and arises from the fact that fine-tuning data is limited. Our current scheme introduces many "jittered" examples (those proposals with overlap between 0.5 and 1, but not ground truth), which expands the number of positive examples by approximately 30x. We conjecture that this large set is needed when fine-tuning the *entire* network to avoid overfitting. However, we also note that using these jittered examples is likely suboptimal because the network is not being fine-tuned for precise localization.

This leads to the second issue: Why, after fine-tuning, train SVMs at all? It would be cleaner to simply apply the last layer of the fine-tuned network, which is a 21-way softmax regression classifier, as the object detector. We tried this and found that performance on VOC 2007 dropped from 54.2% to 50.9% mAP. This performance drop likely arises from a combination of several factors including that the definition of positive examples used in fine-tuning does not emphasize precise localization and the softmax classifier was trained on randomly sampled negative examples rather than on the subset of "hard negatives" used for SVM training.

This result shows that it's possible to obtain close to the same level of performance without training SVMs after fine-tuning. We conjecture that with some additional tweaks to fine-tuning the remaining performance gap may be closed. If true, this would simplify and speed up R-CNN training with no loss in detection performance.

3. Bounding-box regression

We use a simple bounding-box regression stage to improve localization performance. After scoring each selective search proposal with a class-specific detection SVM, we predict a new bounding box for the detection using a class-specific bounding-box regressor. This is similar in spirit to the bounding-box regression used in deformable part models [3]. The primary difference between the two approaches is that here we regress from features computed by the CNN, rather than from geometric features computed on the inferred DPM part locations.

The input to our training algorithm is a set of N training pairs $\{(P^i, G^i)\}_{i=1,\ldots,N}$, where $P^i = (P^i_x, P^i_y, P^i_w, P^i_h)$ specifies the pixel coordinates of the center of proposal P^i's bounding box together with P^i's width and height in pixels. Hence forth, we drop the superscript i unless it is needed. Each ground-truth bounding box G is specified in the same way: $G = (G_x, G_y, G_w, G_h)$. Our goal is to learn a transformation that maps a proposed box P to a ground-truth box G.

We parameterize the transformation in terms of four functions $d_x(P)$, $d_y(P)$, $d_w(P)$, and $d_h(P)$. The first two specify a scale-invariant translation of the center of P's bounding box, while the second two specify log-space translations of the width and height of P's bounding box. After learning these functions, we can transform an input proposal P into a predicted ground-truth box $\hat{G}$ by applying the transformation

$$\hat{G}_x = P_w d_x(P) + P_x \quad (1)$$
$$\hat{G}_y = P_h d_y(P) + P_y \quad (2)$$
$$\hat{G}_w = P_w \exp(d_w(P)) \quad (3)$$
$$\hat{G}_h = P_h \exp(d_h(P)). \quad (4)$$

Each function $d_\star(P)$ (where $\star$ is one of x, y, h, w) is modeled as a linear function of the pool_5 features of proposal P, denoted by $\boldsymbol{\phi}_5(P)$. (The dependence of $\boldsymbol{\phi}_5(P)$ on the image data is implicitly assumed.) Thus we have $d_\star(P) = \mathbf{w}_\star^{\mathrm{T}} \boldsymbol{\phi}_5(P)$, where $\mathbf{w}_\star$ is a vector of learnable model parameters. We learn $\mathbf{w}_\star$ by optimizing the regularized least squares objective (ridge regression):

$$\mathbf{w}_\star = \underset{\hat{\mathbf{w}}_\star}{\operatorname{argmin}} \sum_i^N (t^i_\star - \hat{\mathbf{w}}_\star^{\mathrm{T}} \boldsymbol{\phi}_5(P^i))^2 + \lambda \left\| \hat{\mathbf{w}}_\star \right\|^2 . \quad (5)$$

The regression targets $t_\star$ for the training pair (P, G) are defined as

$$t_x = (G_x - P_x)/P_w \quad (6)$$
$$t_y = (G_y - P_y)/P_h \quad (7)$$
$$t_w = \log(G_w/P_w) \quad (8)$$
$$t_h = \log(G_h/P_h). \quad (9)$$

As a standard regularized least squares problem, this can be solved efficiently in closed form.

We found two subtle issues while implementing bounding-box regression. The first is that regularization is important: we set $\lambda = 1000$ based on a validation set. The second issue is that care must be taken when selecting which training pairs (P, G) to use. Intuitively, if P is far from all ground-truth boxes, then the task of transforming P to a ground-truth box G does not make sense. Using examples like P would lead to a hopeless learning problem. Therefore, we only learn from a proposal P if it is *nearby* at least one ground-truth box. We implement "nearness" by assigning P to the ground-truth box G with which it has maximum IoU overlap (in case it overlaps more than one) if and only if the overlap is greater than a threshold (which we set to 0.6 using a validation set). All unassigned proposals are discarded. We do this once for each object class in order to learn a set of class-specific bounding-box regressors.

At test time, we score each proposal and predict its new detection window only once. In principle, we could iterate this procedure (*i.e.*, re-score the newly predicted bounding box, and then predict a new bounding box from it, and so on). However, we found that iterating does not improve results.

4. Additional feature visualizations

Figure 2 shows additional visualizations for 60 pool_5 units. For each unit, we show the 24 region proposals that maximally activate that unit out of the full set of approximately 10 million regions in all of VOC 2007 test.

We label each unit by its (y, x, channel) position in the $6 \times 6 \times 256$ dimensional pool_5 feature map. Within each channel, the CNN computes exactly the same function of the input region, with the (y, x) position changing only the receptive field.

5. Per-category segmentation results

In Table 1 we show the per-category segmentation accuracy on VOC 2011 val for each of our six segmentation methods in addition to the O_2P method [1]. These results show which methods are strongest across each of the 20 PASCAL classes, plus the background class.

6. Analysis of cross-dataset redundancy

One concern when training on an auxiliary dataset is that there might be redundancy between it and the test set. Even though the tasks of object detection and whole-image classification are substantially different, making such cross-set redundancy much less worrisome, we still conducted a thorough investigation that quantifies the extent to which PASCAL test images are contained within the ILSVRC 2012 training and validation sets. Our findings may be useful to researchers who are interested in using ILSVRC 2012 as training data for the PASCAL image classification task.

We performed two checks for duplicate (and near-duplicate) images. The first test is based on exact matches of flickr image IDs, which are included in the VOC 2007 test annotations (these IDs are intentionally kept secret for subsequent PASCAL test sets). All PASCAL images, and about half of ILSVRC, were collected from flickr.com. This check turned up 31 matches out of 4952 (0.63%).

The second check uses GIST [4] descriptor matching, which was shown in [2] to have excellent performance at near-duplicate image detection in large (> 1 million) image collections. Following [2], we computed GIST descriptors on warped 32×32 pixel versions of all ILSVRC 2012 trainval and PASCAL 2007 test images.

Euclidean distance nearest-neighbor matching of GIST descriptors revealed 38 near-duplicate images (including all 31 found by flickr ID matching). The matches tend to vary slightly in JPEG compression level and resolution, and to a lesser extent cropping. These findings show that the overlap is small, less than 1%. For VOC 2012, because flickr IDs are not available, we used the GIST matching method only. Based on GIST matches, 1.5% of VOC 2012 test images are in ILSVRC 2012 trainval. The slightly higher rate for VOC 2012 is likely due to the fact that the two datasets were collected closer together in time than VOC 2007 and ILSVRC 2012 were.

References

[1] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu. Semantic segmentation with second-order pooling. In *ECCV*, 2012.

[2] M. Douze, H. Jégou, H. Sandhawalia, L. Amsaleg, and C. Schmid. Evaluation of gist descriptors for web-scale image search. In *Proc. of the ACM International Conference on Image and Video Retrieval*, 2009.

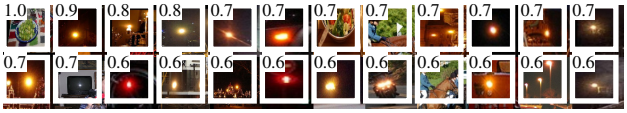
[3] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *TPAMI*, 2010.

[4] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 2001.

VOC 2011 val	bg	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean
O_2P [1]	**84.0**	**69.0**	21.7	47.7	42.2	42.4	**64.7**	**65.8**	57.4	**12.9**	37.4	20.5	43.7	35.7	52.7	51.0	**35.8**	**51.0**	28.4	59.8	49.7	46.4
full R-CNN fc_6	81.3	56.2	23.9	42.9	40.7	38.8	59.2	56.5	53.2	11.4	34.6	16.7	48.1	37.0	51.4	46.0	31.5	44.0	24.3	53.7	51.1	43.0
full R-CNN fc_7	81.0	52.8	**25.1**	43.8	40.5	42.7	55.4	57.7	51.3	8.7	32.5	11.5	48.1	37.0	50.5	46.4	30.2	42.1	21.2	57.7	**56.0**	42.5
fg R-CNN fc_6	81.4	54.1	21.1	40.6	38.7	**53.6**	59.9	57.2	52.5	9.1	36.5	**23.6**	46.4	38.1	53.2	51.3	32.2	38.7	**29.0**	53.0	47.5	43.7
fg R-CNN fc_7	80.9	50.1	20.0	40.2	34.1	40.9	59.7	59.8	52.7	7.3	32.1	14.3	48.8	42.9	54.0	48.6	28.9	42.6	24.9	52.2	48.8	42.1
full+fg R-CNN fc_6	83.1	60.4	23.2	48.4	**47.3**	52.6	61.6	60.6	**59.1**	10.8	**45.8**	20.9	**57.7**	43.3	**57.4**	**52.9**	34.7	48.7	28.1	60.0	48.6	**47.9**
full+fg R-CNN fc_7	82.3	56.7	20.6	**49.9**	44.2	43.6	59.3	61.3	57.8	7.7	38.4	15.1	53.4	**43.7**	50.8	52.0	34.1	47.8	24.7	**60.1**	55.2	45.7

Table 1: Per-category segmentation accuracy (%) on the VOC 2011 validation set.

pool5 feature: (3,3,1) (top 1 – 24)



pool5 feature: (3,3,2) (top 1 – 24)

pool5 feature: (3,3,3) (top 1 – 24)

pool5 feature: (3,3,4) (top 1 – 24)

pool5 feature: (3,3,5) (top 1 – 24)

pool5 feature: (3,3,6) (top 1 – 24)

pool5 feature: (3,3,7) (top 1 – 24)

pool5 feature: (3,3,8) (top 1 – 24)

pool5 feature: (3,3,9) (top 1 – 24)

pool5 feature: (3,3,10) (top 1 – 24)

pool5 feature: (3,3,11) (top 1 – 24)

pool5 feature: (3,3,12) (top 1 – 24)

pool5 feature: (3,3,13) (top 1 – 24)

pool5 feature: (3,3,14) (top 1 – 24)

pool5 feature: (3,3,15) (top 1 – 24)

pool5 feature: (3,3,16) (top 1 – 24)

pool5 feature: (3,3,17) (top 1 – 24)

pool5 feature: (3,3,18) (top 1 – 24)

pool5 feature: (3,3,19) (top 1 – 24)

pool5 feature: (3,3,20) (top 1 – 24)

pool5 feature: (3,3,21) (top 1 – 24)

pool5 feature: (3,3,22) (top 1 – 24)

pool5 feature: (3,3,23) (top 1 – 24)

pool5 feature: (3,3,24) (top 1 – 24)

pool5 feature: (3,3,25) (top 1 – 24)

pool5 feature: (3,3,26) (top 1 – 24)

pool5 feature: (3,3,27) (top 1 – 24)

pool5 feature: (3,3,28) (top 1 – 24)

pool5 feature: (3,3,29) (top 1 – 24)

pool5 feature: (3,3,30) (top 1 – 24)

pool5 feature: (3,3,31) (top 1 – 24)

pool5 feature: (3,3,32) (top 1 – 24)

pool5 feature: (3,3,33) (top 1 – 24)

pool5 feature: (3,3,34) (top 1 – 24)

pool5 feature: (3,3,35) (top 1 – 24)

pool5 feature: (3,3,36) (top 1 – 24)

pool5 feature: (3,3,37) (top 1 – 24)

pool5 feature: (3,3,38) (top 1 – 24)

pool5 feature: (3,3,39) (top 1 – 24)

pool5 feature: (3,3,40) (top 1 – 24)

pool5 feature: (3,3,41) (top 1 – 24)

pool5 feature: (3,3,42) (top 1 – 24)

pool5 feature: (3,3,43) (top 1 – 24)

pool5 feature: (3,3,44) (top 1 – 24)

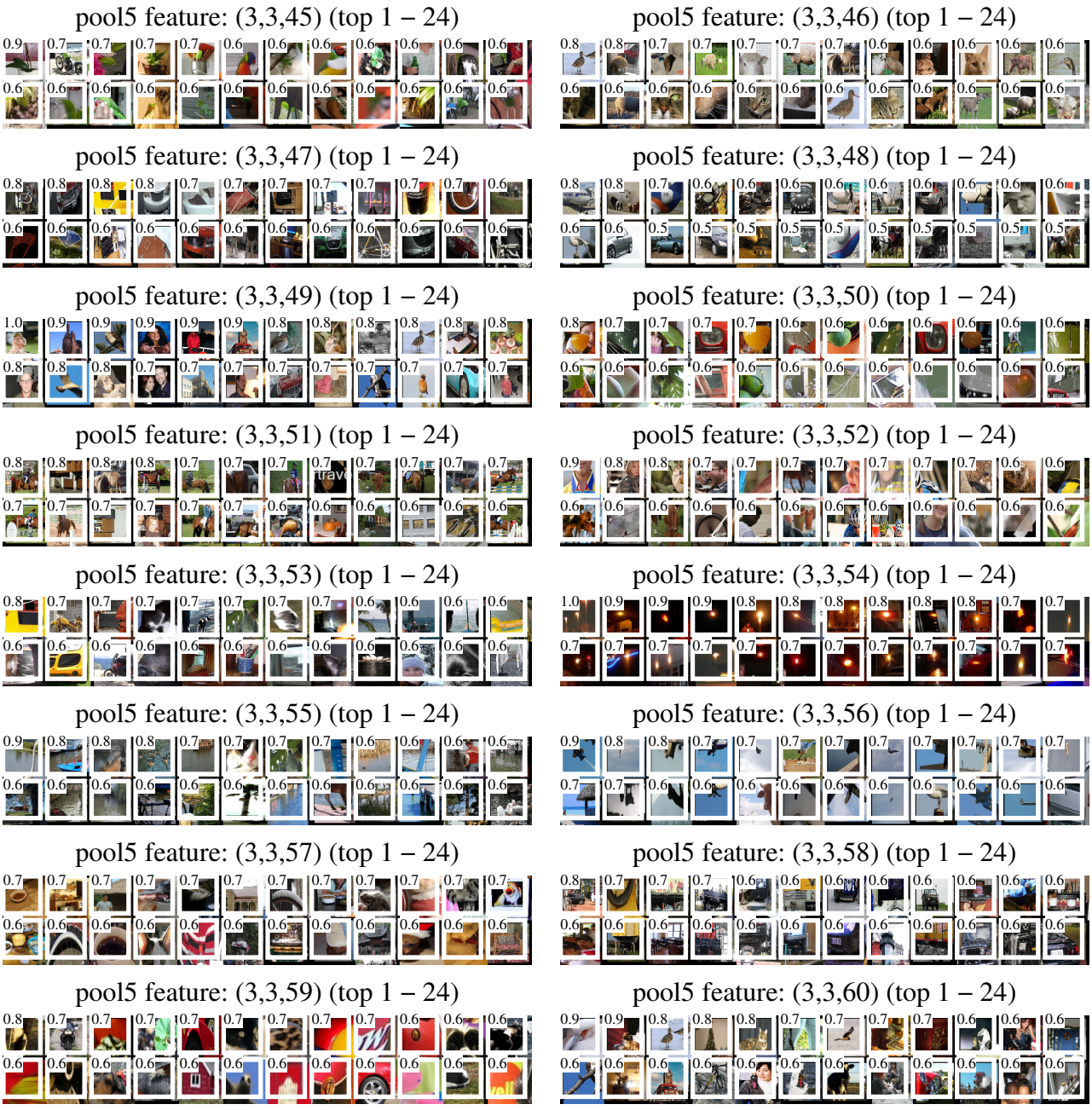


Figure 2: We show the 24 region proposals, out of the approximately 10 million regions in VOC 2007 test, that most strongly activate each of 60 units. Each montage is labeled by the unit's (y, x, channel) position in the $6 \times 6 \times 256$ dimensional pool_5 feature map. Each image region is drawn with an overlay of the unit's receptive field in white. The activation value (which we normalize by dividing by the max activation value over all units in a channel) is shown in the receptive field's upper-left corner. (Best viewed digitally with zoom.)