This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the version available on IEEE Xplore.

ProNet: Learning to Propose Object-specific Boxes for Cascaded Neural Networks

Chen Sun[1,2] Manohar Paluri[2] Ronan Collobert[2] Ram Nevatia[1] Lubomir Bourdev[3]
[1] USC [2] Facebook AI Research [3] UC Berkeley
{chensun, nevatia}@usc.edu {mano, locronan}@fb.com lubomir.bourdev@gmail.com

Abstract

This paper aims to classify and locate objects accurately and efficiently, without using bounding box annotations. It is challenging as objects in the wild could appear at arbitrary locations and in different scales. In this paper, we propose a novel classification architecture ProNet based on convolutional neural networks. It uses computationally efficient neural networks to propose image regions that are likely to contain objects, and applies more powerful but slower networks on the proposed regions. The basic building block is a multi-scale fully-convolutional network which assigns object confidence scores to boxes at different locations and scales. We show that such networks can be trained effectively using image-level annotations, and can be connected into cascades or trees for efficient object classification. ProNet outperforms previous state-of-the-art significantly on PASCAL VOC 2012 and MS COCO datasets for object classification and point-based localization.

1. Introduction

We address the problem of object classification and localization in natural images. As objects could be small and appear at arbitrary locations, several frameworks [18, 37] rely on bounding boxes to train object-centric classifiers, and apply the classifiers by searching over different locations of the images. However, the annotation process for object bounding boxes is usually resource intensive and difficult to scale up. In light of this, we aim to simultaneously classify and locate objects given only *image-level annotations* for training.

To cope with the lack of object-level annotations, several methods [2, 10, 28] extract feature activations from convolutional neural networks (CNN) by scanning over different image regions. They then aggregate the extracted features into image-level representations for classification purpose. Under this scheme, regions that belong to the background are considered as important as regions that contain objects.

Figure 1. We approach object classification problem by zooming onto promising image boxes. No object-level annotations are needed during training.

Such global approaches tend to be sensitive to background, and cannot be used directly for localization.

We choose to use the fully-convolutional network (FCN) architecture [16, 19, 23, 27, 21] for simultaneous object classification and localization. It replaces the fully-connected layers of a standard CNN (e.g. AlexNet [12]) with convolutional layers. This enables an FCN to take images of arbitrary sizes, and generate classification score maps efficiently. Each element in a score map corresponds to a rectangular box (*receptive field*) in the original image. The score maps can then be used for classification and localization.

The sampling strides and box sizes are determined by the FCN's network architecture. As box sizes are fixed, FCN might face difficulty dealing with objects of different scales. We address this problem by using a multi-stream multi-scale architecture. All streams share the same parameters, but take input images of different scales. To train the multi-scale FCN without object-level annotations, we generate image-level scores by pooling the score maps over multiple-scales, and compute the losses with image-level la-

bels for back-propagation.

Once a multi-scale FCN is trained, it can be used for classification and localization directly. From another perspective, it also proposes a set of *promising boxes* that are likely to contain objects. We can then build a *cascade* architecture by zooming onto those promising boxes, and train new classifiers to verify them. The cascade allows the system to balance accuracy and speed: each stage filters out parts of image regions that are unlikely to contain objects. We name this *propose* and *zoom* pipeline as **ProNet**. Figure 1 provides the high-level intuition behind ProNet: three boxes are proposed for *bird*, *potted plant* and *cat* categories. The boxes are cropped out and verified further, until a certain decision is made.

To train the later classifiers in ProNet, we sample *hard negatives* based on image-level labels. For positives, as no object-level annotations are available, it is impossible to tell objects from background. To avoid over-fitting, we randomly sample positive boxes above a relative low threshold. Different positive boxes from the same image can be sampled at different iterations of the stochastic gradient descent training process. At test time, only a small subset of boxes (10 to 20 per image) with highest object confidence scores are fed to the later classifiers. This allows us to utilize CNNs that have stronger representation power with little computational overhead.

ProNet is highly configurable: for example, one could set a list of important object categories, and only verify the proposed boxes for those categories. Moreover, apart from a traditional chain-structured cascade, we show that it is also possible to build tree-structured cascades, where each branch handles categories from a particular domain (*e.g.* set of *vehicles* or *animals*).

In summary, our paper makes the following contributions:

- We propose ProNet, a cascaded neural network framework that zooms onto promising object-specific boxes for efficient object classification and localization.
- We introduce strategies to train ProNet with image-level annotations effectively; and demonstrate the implementations of chain- and tree-structured cascades.
- We show that ProNet outperforms previous state-of-the-art significantly on the object classification and point-based localization tasks of the PASCAL VOC 2012 dataset and the recently released MS COCO dataset.

2. Related Work

Object classification is a fundamental problem in Computer Vision. Earlier work [11, 13, 26] focused on classification from object-centric images. They usually extract hand-crafted low-level features [17] and aggregate the features into image-level feature vectors [22, 34]. More challenging datasets [7, 15, 24] have since been collected. They are of larger scale, and contain smaller objects which could be partially occluded.

Recently, deep convolutional neural networks (CNN) have achieved state-of-the-art performance on a wide range of visual recognition tasks, including object classification [12, 28, 27] and detection [9, 8]. Although CNNs require large amount of data for training, it has been shown that they are able to learn representations that generalize to other tasks. Such representations can be adapted to image classification by fine-tuning [18], or extracted as holistic features for classification with linear SVMs [2]. When used as generic feature extractors, feature aggregation techniques designed for hand-crafted features can also work with CNN embeddings and achieve competitive performance [10].

An alternative approach for object classification is via detection. Among those utilizing bounding box annotations, RCNN [9] achieves competitive performance by directly representing image boxes with CNN features and learning classifiers on top of the features. Object proposal techniques [32, 39] are used to sample the image patches for classification. A recent framework, fast RCNN [8], uses fully-convolutional networks (FCN) to generate box-level features in batch, and is thus more computational efficient.

Object localization with image-level annotations is a weakly-supervised problem. It can be formulated as a multiple instance learning problem, and has been addressed to learn concept detectors from Internet data [3, 6, 36]. It has also been studied for object detection [1, 4, 29, 25] and segmentation [5, 20, 23]. For object classification, Wei et al. [35] treat images as bags of patches, where the patches are selected using objectness criteria. They then use max pooling to fine-tune CNNs based on image-level annotations. Oquab et al. [19] follow a similar approach, but make the training process end-to-end by converting CNNs into FCNs. The proposal generation network in ProNet is also based on FCN, but uses a multi-stream architecture and cross-scale LSE pooling to achieve scale-awareness.

Cascaded classifiers [33] are a well-studied technique in Computer Vision. Cascades with CNNs have been explored for facial point detection [30], face detection [14] and pose estimation [31]. However, such methods require fully annotated training examples. ProNet adopts the cascade philosophy to balance speed and accuracy, but does not require object bounding boxes for training. Since ProNet is a general object classifier, it can also be extended to have tree structure, where each leaf is a domain expert.

3. ProNet Framework

ProNet has two basic components: an object-specific box proposal unit, and a verification unit. For each image, for each object category, the box proposal unit generates a list of confidence scores of the presence of the object in-

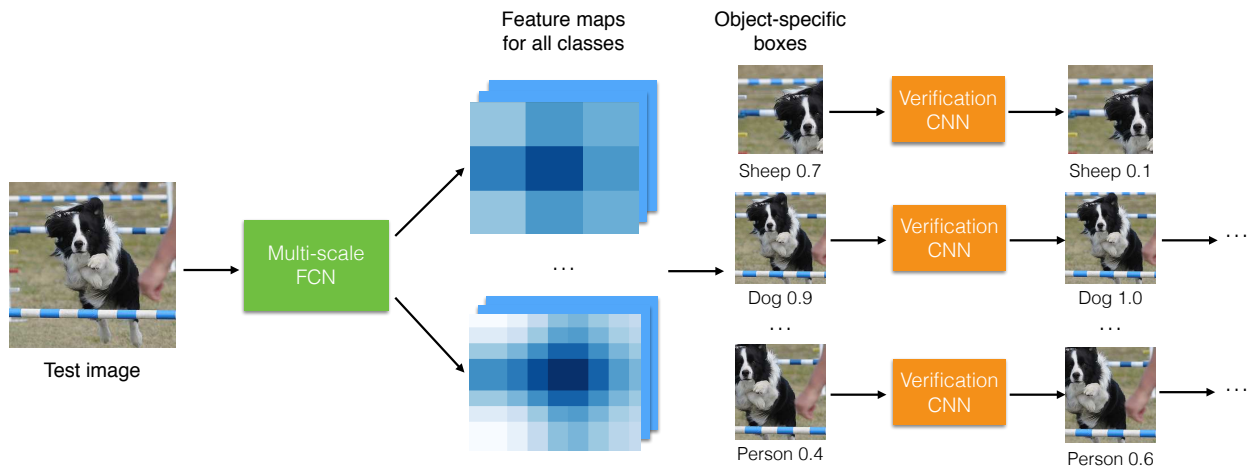

Figure 2. Illustration of the proposed ProNet framework. Given a test image, it first applies a multi-scale fully-convolutional network to select boxes that are likely to contain objects. It then feeds the selected boxes to CNNs trained on harder instances for verification. CNNs at different levels are connected as chains or trees, and trained in a cascade fashion.

stances, and the (x, y) coordinates indicating the locations of the objects. ProNet then zooms onto image boxes with higher scores to further verify if they are positive or hard negatives. The verification units can either take all boxes, which forms a chain structure; or a subset of boxes corresponding to certain domains (*e.g. animal*), which forms a tree structure. We implement these two units with convolutional neural networks. Figure 2 illustrates the overall ProNet framework.

3.1. Proposal Generation

The first stage in our framework is to generate object-specific box proposals with CNNs. For an input image $\mathcal{I}$ and object category c, we want to learn a proposal scoring function

$$P(\mathcal{I}, c, \mathbf{l}) \in \mathbb{R}$$

where $\mathbf{l} = \{x_1, y_1, x_2, y_2\}$ corresponds to the location of a rectangular image region denoted by its top left and bottom right corners.

A typical CNN architecture for image classification task (e.g. AlexNet [12]) involves a hierarchy of convolutional layers and fully connected layers. The convolutional layers operate on local image patches to extract feature representations. For a $W \times H$ color image with 3 channels, the convolutional layers generate a feature map of $D \times W' \times H'$ elements, where D is the output feature dimension. W' and H' correspond to the width and height of the feature map, they are controlled by input image size, as well as the kernel size, sampling step and padding size of the convolutional layers. The fully connected layers serve as classifiers which take fixed-size inputs, thus require the width and height of input images to be fixed. Therefore, one possible way to compute $P(\mathcal{I}, c, \mathbf{l})$ is to enumerate locations and scales in a sliding window fashion or with bounding box proposals, and feed such image regions to CNNs.

We take an alternative approach based on fully convolutional networks (e.g. OverFeat [27]). Fully convolutional networks (FCN) do not contain fully-connected layers. Rather, they use only the convolutional layers, which allows them to process images of arbitrary sizes. The outputs of FCNs are in the form of $C \times W' \times H'$ feature maps, where C is the number of categories. Each element in a feature map corresponds to the activation response for a particular category over a certain region. Such regions are called *receptive fields* for the activations. Compared with region sampling with sliding windows or bounding box proposals, FCNs offer a seamless solution for end-to-end training under the CNN framework, and also naturally allow the sharing of intermediate features over overlapping image regions.

Scale adaptation with multi-stream FCNs. One issue in use of FCNs is that the sizes of receptive fields are typically fixed, while the object scales may vary a lot. We address this problem by using a multi-stream architecture.

Assume an FCN has been trained with inputs where objects have been resized to the same scale. We expand the network into K streams, where every stream shares the same parameters as the pre-trained one. Given an image $\mathcal{I}$, we scale it to different sizes $\{\mathcal{I}_1, \mathcal{I}_2, ..., \mathcal{I}_K\}$ and feed to the K-stream FCN. The output feature map of each stream corresponds to a different scale in the original image.

Training with image-level annotations. When object bounding boxes are available, training FCNs is straightforward: one could either crop images with the bounding boxes, or use a loss function which operates directly on feature maps and takes the object locations into account. As

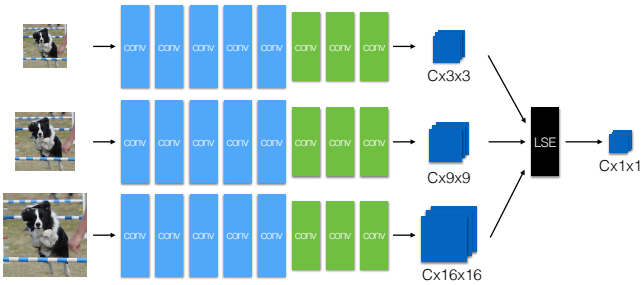

Figure 3. Illustration of 3-stream FCN with LSE pooling.

such supervision is absent, we need to aggregate local responses into global ones so that image-level labels can be used for training. We use the *log-sum-exp* (LSE) pooling function applied by [23] for semantic segmentation:

$$s_c = r^{-1} \log \left[\frac{1}{M} \sum_{x,y,k} \exp(r \cdot s_{c,x,y,k}) \right] \tag{1}$$

where c is the category, k corresponds to the k-th stream of FCN, x, y correspond to location in the feature map, M is the total number of such elements and r is a hyper parameter. The function's output is close to average when r is small and maximum when r is large. Setting r larger makes the aggregation focus on a smaller subset of image boxes, and has the potential to handle smaller objects better.

LSE pooling function can be implemented as a layer in a neural network. As illustrated in Figure 3, it is connected to the final layers of all K-stream FCNs and produces a C dimensional vector for each image. We then compute the loss for each category and back-propagate the error gradients to the earlier layers.

Computing proposal scores. Once the FCNs have been trained, we compute proposal scores $P(\mathcal{I}, c, \mathbf{l})$ from the feature maps. Specifically, for every neuron in the final layer of single-stream FCN, we compute its receptive field and use it as the location $\mathbf{l}$; the corresponding activation of the neuron is used as proposal score.

Although the exact receptive field may vary due to different padding strategies, we use a simple estimation which has been reported to work well in practice [27]. Denote the sampling stride of a spatial convolutional layer C_i as d_{C_i} and the kernel size of a max pooling layer M_j as k_{M_j}, the overall sampling stride D is given by

$$D = \prod_{\mathcal{C}} d_{C_i} \cdot \prod_{\mathcal{M}} k_{M_j} \tag{2}$$

where $\mathcal{C}$ is the collection of all convolutional layers and $\mathcal{M}$ is the collection of all max pooling layers.

Implementation. Our K-stream FCNs are implemented with Torch. For each stream, we use the CNN-M 2048 architecture proposed in [2]. It has 5 convolutional layers and 3 fully-connected layers. It achieves higher accuracy on ImageNet than AlexNet, while being faster and less memory consuming than very deep CNNs [28]. We use the model parameters released by the authors, which were pre-trained from ImageNet dataset with 1,000 categories. We convert the model into an FCN by replacing the three fully-connected layers with convolutional layers. The first convolutional layer has 512 input planes, 4096 output planes and kernel size of 6. The second has 4096 input planes, 2048 output planes and kernel size of 1. Since the final layer is task-specific, it is initialized from scratch with 2048 input planes, $|C|$ output planes and kernel size of 1. To adapt the model parameters for object classification on different datasets, we only fine-tune the final two layers and freeze the model parameters from previous layers. The sampling stride of feature maps is 32 pixels, and the window size is 223 pixels.

We set the number of streams to be 3. During training, all three streams share the same set of parameters. To facilitate training with mini-batches, every image is rescaled to 300×300, 500×500 and 700×700 pixels. As the aspect ratios of images could be different, we rescale the longer edge to 300, 500 and 700 respectively, and fill the empty pixels by mirroring the images.

Traditional cross entropy loss for multi-class classification introduces competition between different classes, thus it is not suitable for images with multiple labels. We compute the loss with binary cross entropy criteria for each class separately, and sum up the error gradients from losses of all classes for back-propagation.

3.2. Cascade-style Proposal Verification

By setting thresholds on proposal scores, a small subset of image boxes which might contain objects are selected. Similar to object detection frameworks, we run CNN classifiers on the selected boxes. The proposal step also serves as a filter whose goal is to preserve the object boxes with high recall rate, while removing the easy negatives. The verification classifiers then address a more focused problem on a smaller set of instances. Connecting the two steps is essentially the same as training a cascade of classifiers.

Verification network architecture. As a later classifier in the cascade, accuracy is more important than speed. We choose the VGG-16 network architecture [28]. Compared with AlexNet variants, it offers better accuracy for most visual recognition tasks, but is also slower and more memory demanding. We use the VGG-16 model parameters released by the authors, which was trained on 1,000 ImageNet categories. We use the same binary cross entropy criterion to compute losses. To make the training process faster, we only fine-tune the final two fully-connected layers and freeze all previous layers.

Algorithm 1: Mini-batch sampling algorithm for training cascade classifier with stochastic gradient descent.

Input : Training images with proposal scores $\mathcal{I}_p$, batch size b, threshold $t \in [0, 1]$

while *stopping criteria not met* **do**
  Randomly select b images $I_1, ..., I_b$ from $\mathcal{I}_p$;
  Initialize mini-batch $\mathbf{T}$;
  for $j = 1, b$ **do**
    if I_j *has proposal with score* $\geq t$ **then**
      Randomly sample a proposal $\mathbf{l}$ where $P(I_j, c, \mathbf{l}) \geq t$;
      Set the sample's active class to c;
      Add proposed region to $\mathbf{T}$;
    end
    else
      Resize and add full image to $\mathbf{T}$;
      Set all classes as active;
    end
  end
  Forward pass with $\mathbf{T}$;
  Compute loss for the active class of each sample;
  Update model parameters.
end

Training strategy for the cascade. Ideally, we want the verification network to handle *hard* examples from both positive and negative data. When a proposed region from an image not containing a given label has a high score of that class, we know it is a hard negative. However, it is impossible to tell a hard positive from background without using bounding box annotations. We attempt to avoid using background by selecting only the top scoring image region for each positive class. This results in significant over-fitting and poor generalizability for the trained verification net.

The main problem with the above sampling strategy is that for positive instances, only *easy* examples which have been learned well are preserved. To fix this, we use a random sampling strategy as described in Algorithm 1. For each image, we randomly select an image box whose proposal score is higher than threshold t for class c. In practice, the threshold is set to a relative low value (*e.g.* 0.1). If c is labeled as positive for the image, we treat the box as a positive instance (though it might belong to background), and otherwise negative. Note that the sampled box could be easy negatives for classes beyond c. To avoid oversampling the easy negatives, we set c as the *active class* during back-propagation and only compute the loss for the active class.

Inference with cascade. During inference, an image is passed to the proposal generation FCN to compute proposal scores. A small subset of proposed boxes with high scores

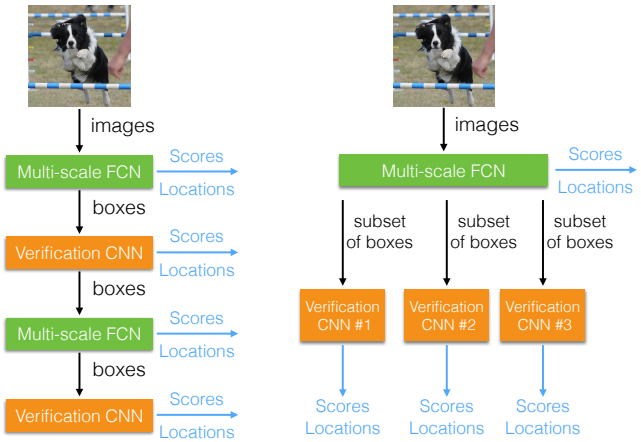

Figure 4. The chain-structure and tree-structure cascades we used in implementing ProNet.

are then passed to the verification network. For each class, we select the top k scoring proposals if the scores are higher than threshold t. We then use the following equation to combine the outputs from both networks:

$$s_c = \begin{cases} \max_{l \in \mathcal{L}_c} s_c^l & \text{if } \mathcal{L}_c \neq \emptyset \\ s_c^p & \text{otherwise} \end{cases} \tag{3}$$

where $\mathcal{L}_c$ is the set of selected proposals for class c, s_c^p is the score of class c from the proposal network after LSE pooling, and s_c^l is the verification network's output for class c on region l. When no proposal is selected, we preserve scores from the proposal network without calibration as they are typically low.

Discussion. Decomposing classification into cascade of proposal and verification networks allows the system to achieve high accuracy while maintaining a reasonable computational cost. It is also a flexible framework for different design choices. For example, one could decide to verify a subset of object classes which require higher accuracy. With the cascade training algorithm, we can build tree-structured cascaded neural networks, where each branch focuses on a subset of categories. We can also extend the cascade to have more stages, and train the new stages with newly annotated training data. Figure 4 illustrates these structures.

4. Experiments

Experimental setup. We work with the PASCAL VOC 2012 dataset [7] and the MS COCO dataset [15]. VOC 2012 has 5,000 images for training, 5,000 for validation and 10,000 for testing. There are 20 object classes in total. COCO has 80,000 images for training and 40,000 images for validation. It has 80 object classes in 12 super-categories.

Average pooling Max pooling LSE pooling

Figure 5. Heat map for class *train* generated by proposal network trained with average pooling, max pooling and LSE pooling respectively.

We evaluated ProNet on object classification and point-based object localization tasks. For object classification, we use the average precision metric. We used VOC's result server to compute *average precisions* on the VOC 2012 dataset. For point-based object localization, we use the criteria introduced in [19]. For every image and every class, we output a location with maximum response for that class. The location is deemed correct if it falls into any bounding box associated with that class, with a tolerance of 18 pixels as used in [19]. This information is then used to compute average precision. Although object extent is not evaluated, the metric remains challenging as shown by [19]. To generate localization coordinates for evaluation, we kept track of the image boxes which give highest responses at each stage, and used the center point of the selected boxes.

We tried different values of hyper-parameter r for LSE pooling, and found that $r \in [5, 12]$ generally gave good performance. We fixed $r = 10$ in all the following experiments. We used the stochastic gradient descent algorithm for training. To train proposal network, the learning rate was set to 0.01; to train verification network, the learning rate was set to 0.001. We set the filtering threshold for cascade to 0.1.

Which pooling method is better? We compare *maximum pooling*, *average pooling* and *LSE pooling* methods to train proposal network with image-level supervision. Table 1 lists the classification and localization performance of the three different pooling methods. We can see that LSE achieves the best classification mAP. Average pooling is 3.7% worse than LSE, which we believe is because it assigns equal importance to foreground and background. Max pooling is 1.4% worse; compared with LSE pooling, it only uses a single patch to generate image-level score, thus is more sensitive to noise and model initialization during training.

We also generated visualizations to study the impact of pooling method on trained models. Figure 5 shows heat maps of the class *train* when different models are applied to the same image. We can see that the model trained by average pooling has high activations not only on the train but also on part of the background. For max pooling, only the wheel of the train has high response, presumably because it is the most discriminative for the train. Model trained by LSE pooling has high response on the train, but not on the background.

Method	Classification	Localization
Oquab et al. [19]	81.8	74.5
RCNN [9]	79.2	74.8
Fast RCNN [8]	87.0	**81.9**
Proposal (Max)	83.4	72.5
Proposal (Mean)	81.1	62.8
Proposal (LSE)	**84.8**	**74.8**
Cascade	88.1	77.7
Second Cascade	**89.0**	**78.5**

Table 1. Classification and localization mAPs on VOC 2012 validation set. Higher mAP is better. Second cascade uses additional training data from MS COCO.

Method	Classification	Localization
Oquab et al. [19]	62.8	41.2
Proposal	67.8	43.5
Chain cascade	69.2	45.4
Tree cascade	**70.9**	**46.4**

Table 2. Classification and localization mAPs on COCO validation set. Higher mAP is better.

Does cascade help? We study the impact of adding cascaded classifiers on classification and localization performance. We first use a single level of cascade with one multi-scale FCN and one verification network. For each image and each class, we selected the top 3 regions per scale if their scores are higher than 0.1. The average number of regions to be verified is 24 per image. In Table 1, we can see that on PASCAL VOC 2012, using a cascade helps improve classification mAP by 3.3% and localization mAP by 2.9%.

Is a longer cascade better? We are interested in observing how the performance changes with more levels of cascade. For this purpose, we first trained another set of proposal and verification networks using PASCAL VOC data alone, but found that the network overfitted easily. Since the training set of VOC 2012 has only 5,000 images, we

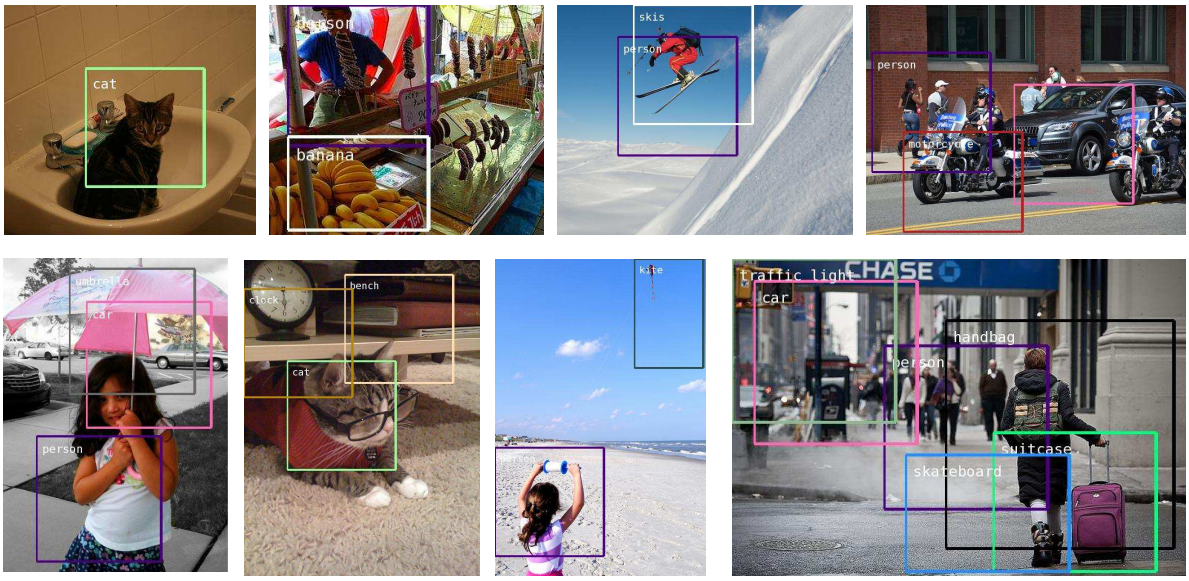

Figure 6. Object localization examples on COCO images. For each object class, we show the box with max score if greater than 0.8. Center point of each box is used for point-based localization evaluation.

k	Ave. #Proposals	Cls.	Loc.
1	9.0	87.7	76.3
2	16.6	87.9	76.8
3	23.9	88.1	77.1
Fast RCNN	#Proposals	Cls.	Loc.
	10	43.2	34.7
	50	70.1	63.2
	500	85.8	80.8
	1000	87.0	81.9

Table 3. Impact of number of boxes passed to verification network on VOC 2012 validation set. We also compare the impact of selective search proposal number for fast RCNN.

found that the first set of proposal and verification networks "perfectly solved" this training set, leaving little room to improve its generalizability.

In light of this, we used the 80,000 images from COCO training set as complementary data source. It covers the 20 categories used in VOC but also has 60 other categories. Rather than re-training all the networks by combining VOC and COCO data, we take that the previous CNNs in the cascade have already been trained and fixed, and only train new CNNs with the extra data. Note that our cascade architecture offers a natural way to select the challenging instances from such incoming images.

The final row in Table 1 shows the mAPs after adding a new set of cascades trained from COCO images. We can see that it offers another 1% improvement over the previous cascade, which indicates that it is desirable to train a longer cascade when more training data becomes available.

Expanding cascades into trees. We also investigated the effect of building tree-structured cascades. COCO dataset is used for evaluation as it has 3 times more categories than VOC.

We trained 12 verification networks corresponding to the 12 super-categories of COCO. Each network focuses on a single super-category, and processes the sampled boxes whose active classes belong to that super-category. At test time, each proposed box only goes through a single root to leaf path in the tree. The final row of Table 2 shows its classification and localization performance. We can see that compared with the chain structured cascade, tree-structured cascade achieves better performance, probably because it trains the neural networks to be focused on a small subset of similar categories.

Comparison with detection based approaches. We compare our proposed framework with two recent state-of-the-art object detection methods: RCNN [9] and Fast RCNN [8]. Unlike our framework, they require bounding box annotations for training. Both methods use *selective search* to generate object proposals and CNNs for classification. RCNN uses AlexNet [12] pre-trained from ImageNet, while fast RCNN uses VGG-16 [28] pre-trained from ImageNet. To generate classification and localization results, for each class we select the detection output with maximum confidence score, and use the center of the detected bounding box for localization evaluation.

We first fix the number of window proposals to 1000 for RCNN and fast RCNN. Table 1 shows the performance comparison. We can see that for classification, our proposed framework outperforms both RCNN and fast RCNN. For localization, our proposed framework outperforms RCNN,

Method	BBox	plane	bike	bird	boat	btl	bus	car	cat	chair	cow	tabl	dog	hors	moto	pers	plant	sheep	sofa	train	tv	mAP
NUS-PSL [37]	Yes	97.3	84.2	80.8	85.3	60.8	89.9	86.8	89.3	75.4	77.8	75.1	83.0	87.5	90.1	95.0	57.8	79.2	73.4	94.5	80.7	82.2
Oquab et al. [18]	Yes	94.6	82.9	88.2	84.1	60.3	89.0	84.4	90.7	72.1	86.8	69.0	92.1	93.4	88.6	**96.1**	64.3	86.6	62.3	91.1	79.8	82.8
NUS [35]+[37]⋆	Yes	**98.9**	**91.8**	**94.8**	**92.4**	**72.6**	**95.0**	**91.8**	**97.4**	**85.2**	**92.9**	**83.1**	**96.0**	**96.6**	**96.1**	94.9	**68.4**	**92.0**	**79.6**	**97.3**	**88.5**	**90.3**
Zeiler et al. [38]	No	96.0	77.1	88.4	85.5	55.8	85.8	78.6	91.2	65.0	74.4	67.7	87.8	86.0	85.1	90.9	52.2	83.6	61.1	91.8	76.1	79.0
Chatfield et al. [2]	No	96.8	82.5	91.5	88.1	62.1	88.3	81.9	94.8	70.3	80.2	76.2	92.9	90.3	89.3	95.2	57.4	83.6	66.4	93.5	81.9	83.2
NUS-HCP [35]	No	97.5	84.3	93.0	89.4	62.5	90.2	84.6	94.8	69.7	90.2	74.1	93.4	93.7	88.8	93.2	59.7	90.3	61.8	94.4	78.0	84.2
Oquab et al. [19]	No	96.7	88.8	92.0	87.4	64.7	91.1	87.4	94.4	74.9	89.2	76.3	93.7	95.2	91.1	97.6	66.2	91.2	70.0	94.5	83.7	86.3
Simonyan et al. [28]⋆	No	**99.0**	88.8	**95.9**	**93.8**	73.1	92.1	85.1	**97.8**	**79.5**	**91.1**	**83.3**	**97.2**	**96.3**	94.5	96.9	63.1	**93.4**	75.0	**97.1**	87.1	89.0
Our Proposal	No	97.0	88.3	92.4	89.8	67.9	90.7	86.2	95.5	73.0	85.5	76.7	94.8	91.1	91.9	97.0	66.1	87.8	68.1	94.1	87.0	86.0
Our Cascade⋆	No	97.6	**91.3**	94.3	93.2	**74.3**	**93.0**	**88.5**	96.8	78.4	90.7	80.1	96.3	95.2	**94.8**	**98.0**	**70.9**	90.3	**75.8**	96.3	**89.4**	**89.3**

Table 4. Classification performance measured by average precision on PASCAL VOC 2012 test set. BBox column indicates whether the training algorithm uses bounding box annotation or not. ⋆: uses VGG-16 models.

Method	BBox	plane	bike	bird	boat	btl	bus	car	cat	chair	cow	tabl	dog	hors	moto	pers	plant	sheep	sofa	train	tv	mAP
RCNN [9]	Yes	92.0	80.8	80.8	73.0	**49.9**	86.8	77.7	87.6	50.4	72.1	57.6	82.9	79.1	89.8	88.1	56.1	83.5	50.1	81.5	76.6	74.8
Fast RCNN [8]	Yes	**95.2**	**88.2**	**88.4**	**77.9**	49.0	**93.4**	**83.6**	**95.1**	**59.4**	**86.6**	**71.0**	**92.6**	**93.1**	**93.0**	**92.2**	**58.2**	**88.0**	**63.6**	**91.9**	**77.3**	**81.9**
Oquab et al. [19]	No	90.3	77.4	81.4	79.2	41.4	87.8	66.4	91.0	47.3	**83.7**	55.1	88.8	**93.6**	85.2	**87.4**	43.5	86.2	50.8	86.8	66.5	74.5
Our Proposal	No	91.6	82.0	85.1	78.6	45.9	87.9	67.1	92.2	51.0	72.9	60.8	89.3	85.1	85.3	86.4	45.6	83.5	55.1	85.6	65.9	74.8
Our Cascade	No	**92.6**	**85.6**	**87.4**	**79.6**	**48.3**	**88.7**	**68.9**	**94.2**	**54.6**	83.2	**62.8**	**92.0**	89.9	**88.2**	87.1	**49.2**	**86.9**	**57.2**	**86.8**	**70.0**	**77.7**

Table 5. Localization performance measured by average precision on PASCAL VOC 2012 validation set.

but is 4% worse than fast RCNN.

We also study the impact of number of proposed boxes on our system's performance. For this purpose, we let the proposal network select top $k = 1, 2, 3$ regions per scale for each class, and compute the average number of proposed boxes per image. For comparison, we ask fast RCNN to use up to 10, 50, 500 and 1000 selective search proposals per image. Table 3 shows the classification and localization performances respectively. We can see that ProNet is quite robust to the number of proposed boxes, and achieves reasonably good performance with only 9 boxes on average. This confirms that ProNet offers better accuracy with relatively small computational overhead. Meanwhile, fast RCNN requires many more proposals to reach peak performance, presumably because the selective search proposals are for general *objectness* and not optimized for object classification in cascade fashion.

Comparison with other weakly-supervised methods. We compare ProNet with several state-of-the-art object classification frameworks. Classification and localization performance on PASCAL VOC 2012 are shown in Table 4 and Table 5 respectively. Table 2 and Figure 6 show results and localization examples on COCO dataset. Among the compared systems, Oquab et al. and NUS-HCP use CNNs pre-trained on the expanded ImageNet data with more than 1500 categories, which has been shown to be useful for classification. Since ProNet uses cascades or trees of CNNs, it can apply a more powerful CNN model VGG-16 with small computational overhead. This helps our system outperform most of the previous state-of-the-art systems significantly on both datasets. ProNet is also slightly better than Simonyan et al. which extracts VGG-16 features at three different scales over full images. Their system is 3x to 6x slower than our cascade at test time.

Limitation. We evaluate ProNet using the standard IOU metric, which considers object extent as well as location. Since the boxes generated by our proposal CNN have fixed aspect ratios, we follow [19] to aggregate the heat maps over 1000 bounding box proposals generated by selective search per image. No bounding box regression is conducted. Cascade CNN is then used to verify the high-scoring proposals. On PASCAL VOC 2012 validation set, our proposal CNN has an mAP of 13.0% when overlap threshold is 0.5. The cascade CNN improves the mAP to 15.5%. Although both results are higher than 11.7% as reported by [19], there is still a huge gap between the state-of-the-art object detection pipelines. Our proposal network tends to select the most discriminative / confusing parts of objects, which is good for cascade classification but bad for getting full object extents. Separating and counting multiple objects are also challenging issues.

5. Conclusion

We proposed ProNet, a cascaded neural network for object classification and localization. ProNet learns to propose object-specific boxes by multi-scale FCNs trained from image-level annotations. It then sends a small subset of promising boxes to latter CNNs for verification. Detailed experimental evaluations have shown the effectiveness of ProNet on the challenging PASCAL VOC 2012 dataset and MS COCO dataset.

Acknowledgement: We would like to thank Sergey Zagoruyko for help with fast RCNN experiments; Pedro O. Pinheiro, Bolei Zhou, Maxime Oquab, Joël Legrand, Yuandong Tian, Léon Bottou and Florent Perronnin for valuable discussions.

References

[1] H. Bilen, M. Pedersoli, and T. Tuytelaars. Weakly supervised object detection with convex clustering. In *CVPR*, 2015. 2

[2] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*, 2014. 1, 2, 4, 8

[3] X. Chen, A. Shrivastava, and A. Gupta. NEIL: Extracting visual knowledge from web data. In *ICCV*, 2013. 2

[4] R. G. Cinbis, J. J. Verbeek, and C. Schmid. Multi-fold MIL training for weakly supervised object localization. In *CVPR*, 2014. 2

[5] J. Dai, K. He, and J. Sun. Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In *ICCV*, 2015. 2

[6] S. K. Divvala, A. Farhadi, and C. Guestrin. Learning everything about anything: Webly-supervised visual concept learning. In *CVPR*, 2014. 2

[7] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010. 2, 5

[8] R. B. Girshick. Fast R-CNN. In *ICCV*, 2015. 2, 6, 7, 8

[9] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 2, 6, 7, 8

[10] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *ECCV*, 2014. 1, 2

[11] K. Grauman and T. Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *ICCV*, 2005. 2

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*. 2012. 1, 2, 3, 7

[13] F. Li, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *CVIU*, 2007. 2

[14] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua. A convolutional neural network cascade for face detection. In *CVPR*, 2015. 2

[15] T. Lin, M. Maire, S. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. 2, 5

[16] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 1

[17] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 2

[18] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*, 2014. 1, 2, 8

[19] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Is object localization for free? - weakly-supervised learning with convolutional neural networks. In *CVPR*, 2015. 1, 2, 6, 8

[20] G. Papandreou, L. Chen, K. Murphy, and A. L. Yuille. Weakly- and semi-supervised learning of a DCNN for semantic image segmentation. In *ICCV*, 2015. 2

[21] D. Pathak, E. Shelhamer, J. Long, and T. Darrell. Fully convolutional multi-class multiple instance learning. In *ICLR*, 2015. 1

[22] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *CVPR*, 2007. 2

[23] P. H. O. Pinheiro and R. Collobert. From image-level to pixel-level labeling with convolutional networks. In *CVPR*, 2015. 1, 2, 4

[24] 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, 2014. 2

[25] O. Russakovsky, Y. Lin, K. Yu, and F. Li. Object-centric spatial pooling for image classification. In *ECCV*, 2012. 2

[26] J. Sánchez, F. Perronnin, T. Mensink, and J. J. Verbeek. Image classification with the fisher vector: Theory and practice. *IJCV*, 2013. 2

[27] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229, 2013. 1, 2, 3, 4

[28] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1, 2, 4, 7, 8

[29] H. O. Song, R. B. Girshick, S. Jegelka, J. Mairal, Z. Harchaoui, and T. Darrell. On learning to localize objects with minimal supervision. In *ICML*, 2014. 2

[30] Y. Sun, X. Wang, and X. Tang. Deep convolutional network cascade for facial point detection. In *CVPR*, 2013. 2

[31] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. In *CVPR*, 2014. 2

[32] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *IJCV*, 2013. 2

[33] P. A. Viola and M. J. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001. 2

[34] J. Wang, J. Yang, K. Yu, F. Lv, T. S. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, 2010. 2

[35] Y. Wei, W. Xia, J. Huang, B. Ni, J. Dong, Y. Zhao, and S. Yan. CNN: single-label to multi-label. *CoRR*, abs/1406.5726, 2014. 2, 8

[36] J. Wu, Y. Yu, C. Huang, and K. Yu. Deep multiple instance learning for image classification and auto-annotation. *CVPR*, 2015. 2

[37] S. Yan, J. Dong, Q. Chen, Z. Song, Y. Pan, W. Xia, H. Zhongyang, Y. Hua, and S. Shen. Generalized hierarchical matching for subcategory aware object classification. In *ECCV Workshop*, 2012. 1, 8

[38] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 8

[39] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014. 2