

# Object Detection Networks on Convolutional Feature Maps

Shaoqing Ren   Kaiming He   Ross Girshick   Xiangyu Zhang   Jian Sun

Microsoft Research

{v-shren, kahe, rbg, v-xiangz, jiansun}@microsoft.com

## Abstract

*Most object detectors contain two important components: a feature extractor and an object classifier. The feature extractor has rapidly evolved with significant research efforts leading to better deep ConvNet architectures. The object classifier, however, has not received much attention and most state-of-the-art systems (like R-CNN) use simple multi-layer perceptrons. This paper demonstrates that carefully designing deep networks for object classification is just as important. We take inspiration from traditional object classifiers, such as DPM, and experiment with deep networks that have part-like filters and reason over latent variables. We discover that on pre-trained convolutional feature maps, even randomly initialized deep classifiers produce excellent results, while the improvement due to fine-tuning is secondary; on HOG features, deep classifiers outperform DPMs and produce the best HOG-only results without external data. We believe these findings provide new insight for developing object detection systems. Our framework, called Networks on Convolutional feature maps (NoC), achieves outstanding results on the PASCAL VOC 2007 (73.3% mAP) and 2012 (68.8% mAP) benchmarks.*

## 1. Introduction

Most object detectors contain two important components: a feature extractor and a classifier. The feature extractor in traditional object detection methods is a hand-engineered module, such as HOG [4]. The classifier is often a linear SVM (possibly with a latent structure over the features) [8], a non-linear boosted classifier [30], or an additive kernel SVM [27].

Recently, large performance improvements have been realized by training deep ConvNets [18] for object detection. R-CNN [9], one particularly successful approach, starts with a pre-trained ImageNet [5] classification network and then fine-tunes the ConvNet, end-to-end, for detection. Although these methods blur the distinction between the feature extractor and the classifier, a logical division can still be imposed. For example, an R-CNN can be thought of

as a convolutional feature extractor, ending at the last pooling layer, followed by a multi-layer perceptron (MLP). To date, even when extra training data are used by traditional methods [34], they still trail far behind deep ConvNets on detection benchmarks.

One research stream [24, 11, 29, 36] attempting to bridge the performance gap between traditional detectors and deep ConvNets creates a hybrid of the two: the feature extractor is “upgraded” to a pre-trained deep ConvNet, but the classifier is left as a traditional model, such as a DPM [24, 11, 29] or a boosted classifier [36]. These hybrid approaches outperform their HOG/SIFT/LBP-based counterparts [8, 30], but still lag far behind R-CNN, even when the DPM is trained end-to-end with deep ConvNet features [29]. Interestingly, the detection accuracy of these hybrid methods is close to that of R-CNN when using a linear SVM on the last convolutional features, without the fully-connected layers.

The SPPnet approach [13] for object detection occupies a middle ground between the hybrid models and R-CNN. SPPnet, like the hybrid models but *unlike* R-CNN, uses a pre-trained deep ConvNet as a feature extractor. For classification, SPPnet uses a fine-tuned multi-layer perceptron, just like R-CNN but *unlike* the hybrid methods. For the range of models considered in [13], SPPnet performs similarly to R-CNN and thus significantly outperforms the hybrid methods.

From these systems and results, the de facto strategy for object detection is now: use a pre-trained deep ConvNet for feature extraction (with or without fine-tuning) followed by a fine-tuned MLP for classification. This strategy is, however, likely only driven by the pre-trained architecture, which motivates us to ask two questions: (1) Can we design a better region classifier for detection than an MLP? and (2) How important is fine-tuning for region classification?

We investigate these questions within an experimental framework we call Networks on Convolutional feature maps, or *NoCs* for short. We propose to use a fixed, pre-trained deep ConvNet as a feature extractor and explore different NoC architectures, each of which implements an object classifier. To answer the first question (*Is there a better classifier architecture?*), we design and test three NoC fam-

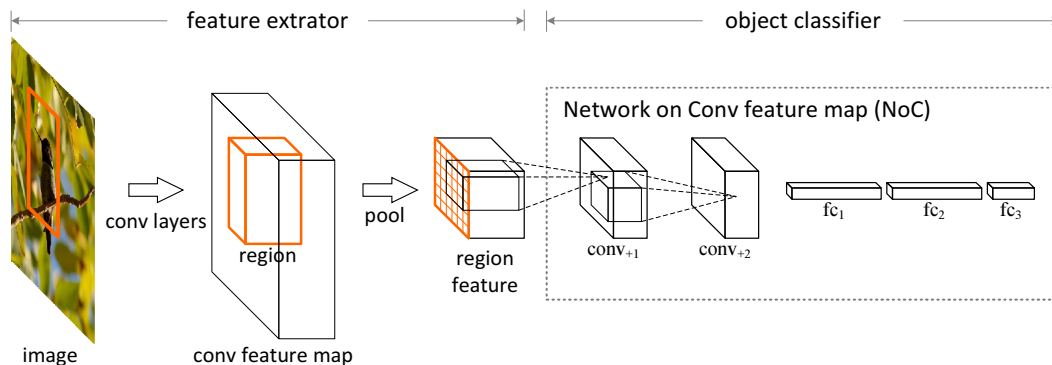


Figure 1. Overview of NoC. The convolutional feature maps are generated by the convolutional layers of a pre-trained model. A feature map region is extracted and pooled into a fixed-resolution “tiny image.” A new network, called a NoC, is then designed and trained on these tiny images. In this illustration, the NoC architecture consists of two convolutional layers and three fully-connected layers.

ilies: MLPs of various depths, ConvNets of various depths, and ConvNets with maxout [12] for latent scale selection. To answer the second question (*How important is classifier fine-tuning?*), we train NoCs from random initialization and from pre-trained ImageNet weights (fine-tuning). We assess the relative improvement in mAP coming from NoC design versus fine-tuning.

Unsurprisingly, we find that pre-training plus fine-tuning works the best. However, we also find a surprising result: a well-designed NoC (ConvNet with maxout) performs extremely well when trained from a random initialization and the improvement from NoC design is larger than from fine-tuning. These results show that the de facto strategy is not the end of the story; there are still significant gains to be had from designing new classification networks with novel detection-tailored properties, like additional convolutional layers motivated by part models [11] and maxout for latent scale selection.

We believe these findings provide new insights into the design of object detection systems. In particular, they suggest that designing deep classifier networks on top of feature extractors is a good strategy. One hypothesis inspired by these results is that a well-designed NoC applied to HOG features will outperform traditional HOG classifiers, such as DPMs and boosted classifiers. To test this hypothesis we design and experiment with NoCs on top of HOG feature maps. Doing so leads to 39% mAP when training on PASCAL VOC 2007 [7] data only, which is the best HOG-only detection performance published to date. This compares favorably to DPM (34% mAP) [8], showing the generality of our findings. These results in turn suggest that the accuracy gap between the hybrid methods [24, 11, 29] and R-CNN is mainly due to the power of the classifier.

As a final contribution, we improve upon the state-of-the-art detection results by exploiting pre-trained very deep ConvNets [26] with NoCs. With this approach we achieve excellent results on PASCAL VOC 2007 (**73.3%** mAP) and

2012 (**68.8%** mAP). Our method is also computationally efficient, as is the case with methods that share computation through convolutional feature maps [13, 21].

## 2. Related Work

**Traditional Object Detection.** The pioneering work of Viola and Jones [28] uses simple Haar-like features and boosted classifiers. The pedestrian detection method in [4] proposes HOG features used with linear SVMs. The DPM method [8] develops deformable graphical models and latent SVM as a sliding-window classifier. The Selective Search paper [27] relies on spatial pyramid features [19] on SIFT vectors [22] and an additive kernel SVM. The Regionlet method [30] learns boosted classifiers on a combination of HOG, LBP [2], and covariance features. In these methods, the regions to be classified are either densely enumerated sliding windows [28, 4, 8] or computed object proposals [27, 30].

**Convolutional Feature Maps.** Convolutional layers can be applied to images of arbitrary size yielding proportionally-sized feature maps. In the Overfeat method [25], the fully-connected layers are used on each sliding window of the convolutional feature maps for efficient classification. The Overfeat method also proposes a regressor on the feature maps, which outputs bounding box coordinates for localization or detection. In the SPP-based object detection method [13], features are pooled from proposal regions [27] on convolutional feature maps, and fed into the original fully-connected layers for classifying regions. In [3], networks with a masking layer on feature maps are developed for semantic segmentation. The Fully Convolutional Network (FCN) [21] reshapes all fully-connected layers as convolutions for efficient semantic segmentation. For these methods [13, 3, 21], the fully-connected layers are initialized from a pre-trained model and fine-tuned.

### 3. Experimental Setting

**Dataset.** We investigate NoCs on the PASCAL VOC 2007 object detection benchmark [7]. This dataset covers 20 object categories, and performance is measured by mAP on the *test* set of 5k images. We investigate two sets of training images: (i) the original *trainval* set of 5k images in VOC 2007, and (ii) an augmented set of 16k images that consists of VOC 2007 trainval images and VOC 2012 trainval images, following [1]. We test certain NoC configurations on the 2012 *test* set using the evaluation server.

**Pre-trained Models.** As a common practice [9, 13, 21], we adopt deep CNNs pre-trained on the 1000-class ImageNet dataset [5] as feature extractors. We investigate Zeiler and Fergus’s (ZF) model [32] and VGG models [26]. The ZF model has five convolutional (conv) layers and three fully-connected (fc) layers. We use a ZF model with an SPP layer, which is released by [13].<sup>1</sup> The VGG-16/19 models [26]<sup>2</sup> have 13/16 conv layers and three fc layers.

### 4. Method

On the convolutional feature maps, we design and train new networks (referred to as NoCs) for object detection. We overview the algorithm pipeline in this section, and detail the NoC designs in the next section.

**Region Extraction.** Given an image, we apply the conv layers of a pre-trained model to compute the convolutional feature map of the entire image. As in [25, 13, 11], we extract feature maps from multiple image scales. These feature maps are fixed and the pre-trained conv layers will not be further tuned. We also extract about 2,000 candidate regions proposed by Selective Search [27]. The rectangular boundary of each proposal is projected onto the convolutional feature map [13].

The feature map regions are of arbitrary spatial sizes. We generate a *fixed-resolution* feature map region via a region pooling operation that has a fixed output resolution. Formally, we define a desired fixed output spatial resolution  $m \times m$ , which is the output spatial size of the last pooling layer in the pre-trained model (e.g.,  $6 \times 6$  for ZF net and  $7 \times 7$  for VGG-16/19). For an arbitrary feature map region of size  $w \times h$ , we produce the  $m \times m$  output by max pooling in spatial bins of a size  $\frac{w}{m} \times \frac{h}{m}$ . This operation is a special case of SPP in [13]. Here, the pyramid has a single level of  $m \times m$  spatial resolution. The pooled feature map regions can be thought of as tiny multi-channel images.

**Training and Inference.** We consider the tiny images of feature map regions as a new data source and design var-

ious NoC architectures (Sec. 5) to classify this data. The NoC structures have multiple layers, and the last layer is an  $(n+1)$ -way classifier for  $n$  object categories plus background. We implement this classifier by an  $(n+1)$ -d fc layer followed by softmax. Each NoC is trained by backpropagation and stochastic gradient descent (SGD) using the “tiny image” data generated on the detection training set. After network training, we use the second-to-last fc layer in the NoC to extract features from regions, and train a linear SVM classifier for each category using these features, as in R-CNN [9].

For inference, the tiny images of feature map regions are extracted and fed into the NoC till the second-to-last fc layer. The SVM classifier is then used to score each region, followed by non-maximum suppression [9].

### 5. Networks on Convolutional Feature Maps

Given the  $m \times m$  region features, we design and investigate various network architectures as classifiers.

#### 5.1. NoC as an MLP

A simple design of NoC is to use only fc layers, known as a multi-layer perceptron (MLP) [17]. We investigate using 2 to 4 fc layers. The last fc layer is always  $(n+1)$ -d with softmax, and the other fc layers are 4,096-d with Rectified Linear Units (ReLUs) [23]. As an example, we denote the NoC structure with 3 fc layers as “f4096-f4096-f21” where “f” denotes an fc layer and 21 is for the VOC categories.

When the ZF net is used, we apply multi-level pooling before the first fc layer. On the  $m \times m = 6 \times 6$  feature map region, we pool three low-resolution feature maps of spatial sizes  $\{3 \times 3, 2 \times 2, 1 \times 1\}$ . The  $6 \times 6$  and lower-resolution feature maps are concatenated and fed into the first fc layer. As such, in the special case when 3 fc layers are used, the NoC has the same structure as the SPP detection method [13], as we will discuss in Sec. 5.4.

Table 1 shows the results of using MLP as NoC. Here we randomly initialize the weights by Gaussian distributions, so various NoC architectures can be compared fairly. The result of “SVM on pool<sub>5</sub>” is based on [13], for which the SVM is trained on the pool<sub>5</sub> features. The results of NoC with 2 to 4 fc layers are in Table 1. Compared with the SVM classifier trained on pool<sub>5</sub>, the 4-fc NoC (53.6%) as a classifier on the same features has 7.8% higher mAP. Note that in this comparison the NoC has no pre-training. The gain is solely because the multi-layer network models the data better than the single-layer SVM.

#### 5.2. NoC as a ConvNet

The first fc layer in an MLP plays the role of a large number (4096) of mixture components, conceptually similar to the root filters in a DPM [8]. Motivated by the part filters

<sup>1</sup>research.microsoft.com/en-us/um/people/kahe/eccv14sppnet/

<sup>2</sup>www.robots.ox.ac.uk/~vgg/research/very\_deep/

method	architecture	VOC 07
SVM on pool <sub>5</sub> [13]	f21	45.8
2-fc NoC	f4096-f21	49.0
3-fc NoC	f4096-f4096-f21	53.1
4-fc NoC	f4096-f4096-f4096-f21	<b>53.6</b>

Table 1. Detection results of **NoC as MLP** for PASCAL VOC 07 using a ZF net. The training set is PASCAL VOC 07 trainval. The NoCs are randomly initialized. No bbox regression is used.

method	architecture	VOC 07	07+12
3-fc NoC	f4096-f4096-f21	53.1	56.5
1-conv NoC	c256-f4096-f4096-f21	<b>53.3</b>	58.5
2-conv NoC	c256-c256-f4096-f4096-f21	51.4	<b>58.9</b>
3-conv NoC	c256-c256-c256-f4096-f4096-f21	51.3	58.8

Table 2. Detection results of **NoC as ConvNet** for PASCAL VOC 07 using a ZF net. The training sets are PASCAL VOC 07 trainval and PASCAL VOC 07+12 trainval respectively. The NoCs are randomly initialized. No bbox regression is used.

method	architecture	VOC 07+12
2-conv NoC	c256-c256-f4096-f4096-f21	58.9
mo input	mo-c256-c256-f4096-f4096-f21	60.1
mo conv <sub>+1</sub>	c256-mo-c256-f4096-f4096-f21	<b>60.7</b>
mo fc <sub>1</sub>	c256-c256-f4096-mo-f4096-f21	60.3
mo output	c256-c256-f4096-f4096-f21-mo	60.1

Table 3. Detection results of **maxout NoC** for PASCAL VOC 07 using a ZF net. The training set is PASCAL VOC 07+12 trainval. The NoCs are randomly initialized. No bbox regression is used.

of a DPM, we investigate adding conv layers to NoCs. A conv filter has a smaller receptive field on the feature map, as is the case with a part filter for a DPM. In fact, a DPM part filter can be recast as a conv layer followed by distance transform pooling [11].

We investigate using 1 to 3 additional conv layers. All conv layers are followed by ReLUs. We use 256 filters for the ZF net and 512 for the VGG net, which is equal to the filter number of the last conv layer in the pre-trained models. The conv filters have a spatial size of  $3 \times 3$  and a padding of 1, so the  $(m \times m)$  output spatial resolution is unchanged. After the last additional conv layer, we apply three fc layers as in the above MLP case. As an example, we denote an NoC with 2 conv layers as “c256-c256-f4096-f4096-f21”. We apply SPP after the last additional conv layer in the case of the ZF net.

In Table 2 we compare the cases of no conv layer (*i.e.*, 3-layer MLP) and using 1 to 3 additional conv layers. Here we randomly initialize all NoC layers. When using VOC 07 trainval for training, the mAP improves slightly when using

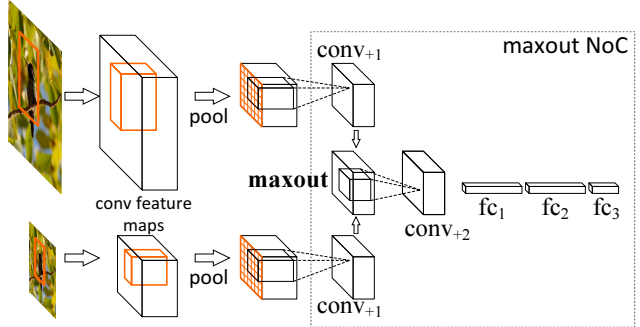


Figure 2. A maxout NoC of “c256-mo-c256-f4096-f4096-f21”. The features are pooled from two feature maps computed at two scales. After pooling, the two features have the same size (*e.g.*,  $6 \times 6$ ), and are used as two inputs to the network. In this figure, maxout is used after conv<sub>+1</sub>.

1 additional conv layer, but drops when using more conv layers. The degradation is a result of overfitting, because we observe that the additional conv layers effectively reduce the training error. The VOC 07 trainval set is too small to train deeper models.

However, NoCs with conv layers show improvements when trained on the VOC 07+12 trainval set (Table 2). For this training set, the 3-fc NoC baseline is lifted to 56.5% mAP. The deeper 2-conv NoC improves over this baseline to 58.9%. This justifies the effects of the additional conv layers. Table 2 also shows that the mAP remains the same when using 3 additional conv layers. But it is reasonable to hope that more training data will further improve deeper convolutional NoC models.

### 5.3. Maxout for Scale Selection

Our convolutional feature maps are extracted from multiple discrete scales, known as a feature pyramid [8]. In the following, we incorporate a local competition operation (maxout) [12] into NoCs to improve scale selection from the feature pyramid.

In the above, a region feature is pooled from a single selected scale (detailed in Sec. 7). To improve scale invariance, for each proposal region we select two adjacent scales in the feature pyramid. Two fixed-resolution  $(m \times m)$  features are pooled, and the NoC model has two data sources.

Maxout [12] (element-wise max) is a widely considered operation for merging two or multiple competing sources. In DPM [8], the prediction scores of multiple components compete with each other, and the component with the highest score is taken. This can be formulated as a maxout operation on the outputs of multiple networks [11]. Alternatively, the maxout operation can be applied on the network inputs from multiple scales to select the most responsive features.

In the case of a deep structure, the maxout operation

method	init.	VOC 07	07+12
SVM on pool <sub>5</sub> [13]	-	45.8	47.7
3-fc NoC	random	53.1	56.5
	fine-tune	<b>55.8</b>	<b>58.0</b>
maxout 2-conv NoC	random	54.7	60.7
	fine-tune	<b>57.7</b>	<b>62.9</b>

Table 4. Detection results of NoC with/without **fine-tuning** for PASCAL VOC 07 using a ZF net. The training sets are PASCAL VOC 07 trainval and PASCAL VOC 07+12 trainval respectively. The “maxout NoC” is the 2-conv NoC with maxout after conv<sub>+1</sub>. No bounding box regression is used.

can also be applied on any intermediate layers. We investigate NoCs with maxout after several different layers. As an example, the NoC model of “c256-mo-c256-f4096-f4096-f21” is illustrated in Fig. 2. When the maxout operation is used, the two feature maps (for the two scales) are merged into a single feature of the same dimensionality using element-wise max. There are two pathways before the maxout, and we let the corresponding layers in both pathways share their weights. Thus the total number of weights is unchanged when using maxout.

Table 3 shows the mAP of the four variants of maxout NoCs. Their mAP is higher than that of the non-maxout counterpart. The case of maxout after conv<sub>+1</sub> performs the best, and it improves over the no-maxout baseline by 1.8% to 60.7% mAP. We note that the results thus far are all without bounding-box regression.

#### 5.4. Fine-tuning

In the above, all NoC architectures are initialized randomly. Whenever possible, we can still transfer weights from a pre-trained architecture and fine-tune the NoCs. The comparison of random initialization vs. fine-tuning provides new insights into the impacts of the well established fine-tuning strategy [9].

For fine-tuning, we initialize the two 4096-d layers by the two corresponding fc layers in the pre-trained model. As such, the fine-tuned 3-fc NoC becomes equivalent to the SPPnet object detection method [13]. For the cases of additional conv layers, each conv layer is initialized to the identity map, and thus the initial network state is equivalent to the pre-trained 3-fc structure. We compare the results of an SVM on pool<sub>5</sub>, randomly initialized NoC, and fine-tuned NoC. Table 4 shows the cases of the 3-fc NoC and the best performing maxout 2-conv NoC.

**How important is fine-tuning for classifiers?** Unsurprisingly, the fine-tuned models boost the results, and the maxout 2-conv NoC has an mAP of 62.9%. However, it is less expected to see that the randomly initialized NoCs produce excellent results. Compared with the SVM counterpart using the same features (47.7%, Table 4), the randomly ini-

pre-trained	method	VOC 07+12
VGG-16	3-fc NoC	64.6
VGG-16	2-conv NoC	66.1
VGG-16	maxout 2-conv NoC	<b>68.8</b>

Table 5. Detection results of NoC for PASCAL VOC 07 using VGG nets. The training set is PASCAL VOC 07+12 trainval. The NoC is the fine-tuned version (Sec. 5.4). The “maxout NoC” is the 2-conv NoC with maxout after conv<sub>+1</sub>. No bounding box regression is used.

tialized NoC (60.7%) showcases an improvement of 13.0%, whereas the fine-tuned counterpart (62.9%) has an extra 2.2% gain. This indicates that the fine-tuning procedure, for the classifier, obtains a majority of accuracy via training a network on the detection data, rather than inheriting pre-trained information.

For the above result (60.7%), the pre-trained features are fixed and not fine-tuned, the training is not end-to-end, and the randomly initialized NoCs as classifiers are not transferred. Under these ablation settings, we are able to delve into the “black boxes” and separately probe the importance of each building block. Our investigations suggest that *deep features* aside, a carefully designed *deep classifier* is an essential factor for object detection. While the depth of our classifier is limited by the amount of training data available (e.g., 16k VOC 07+12 trainval images for training a 5-layer NoC), it is still a deeper and more powerful classifier than an SVM.

It is worth noticing that we do not undervalue the importance of feature fine-tuning and end-to-end training. Whenever it is feasible, we conjecture that an end-to-end training (fine-tuning) of deep features plus deep classifiers may produce even better results. We have not done so in this paper, because the implementation of such end-to-end training is difficult. But our entire pipeline (Fig. 1) has been readily formulated as an end-to-end network, and forward/backward propagations are thus possible. We plan to investigate this in the future.

#### 5.5. Using Very Deep Pre-trained Models

We find that our NoCs also enjoy benefits from deeper pre-trained models [26], and a deep classifier is complementary to pre-trained very deep features.

Table 5 shows the fine-tuned NoC results using the VGG-16 model [26]. The mAP of the baseline 3-fc NoC is 64.6%. The 2-conv NoC improves to 66.1%, and the maxout 2-conv NoC further increases the mAP to 68.8%. On the other hand, the baseline 3-fc NoC using the VGG-19 model is 65.1%, which is inferior to the results of sophisticated NoCs. With the maxout 2-conv NoC, VGG-19 and VGG-16 perform just comparably.

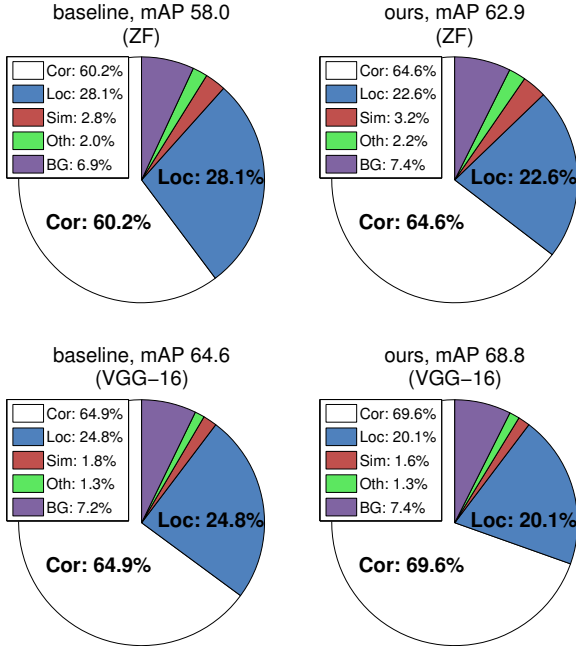


Figure 3. Distribution of top-ranked True Positives (TP) and False Positives (FP), generated by the published diagnosis code of [16]. The types of positive predictions are categorized [16] as Cor (correct), Loc (false due to poor localization), Sim (confusion with a similar category), Oth (confusion with a dissimilar category), BG (fired on background). The total number of samples in each disk is the same and equal to the total number of ground-truth labels [16]. The baseline is the fine-tuned 3-fc NoC [13]. “Ours” refers to the fine-tuned maxout 2-conv NoC.

**Error Analysis.** To better understand the effect of NoC, we use the diagnosis tool of [16] to analyze the top-ranked false-positive predictions (Fig. 3). The false positives due to *poor localization* are denoted as “Loc”. The false positives due to *category recognition error* consist of “Sim” (confusion with a similar category), “Oth” (confusion with a dissimilar category), “BG” (fired on background). Here the baseline is the fine-tuned 3-fc NoC. “Ours” refers to the fine-tuned maxout 2-conv NoC.

Fig. 3 shows that the deeper VGG-16 model has an overall lower recognition error than the shallower ZF model, respectively for the baseline cases and our NoC cases. The effect of NoC is mainly the considerable reduction of localization error, *e.g.*, reducing from 24.8% to 20.1% in the VGG-16 case. This analysis indicates that a deeper NoC helps to substantially reduce *localization* errors, while the *recognition* error is mainly reduced by a deeper pre-trained model.

method	mAP
DPM + HOG, v5 [10]	33.7
Regionlet + HOG [30]	35.1
NoC + HOG	37.0
NoC + HOG, bb	<b>39.4</b>

Table 6. Detection results **using HOG features only** for PASCAL VOC 07. The training set for all methods is PASCAL VOC 07 trainval. No external data is used.

## 6. Networks on HOG Feature Maps

The above experiments suggest that designing deep networks on top of feature maps is important. To provide a further proof of concept, we experiment with NoCs on top of HOG feature maps. There is no pre-training or fine-tuning available in this setting.

We extract HOG feature maps using the public code of DPM v5 [10]. These feature maps consist of 31-dimensional HOG vectors and have a stride of 8 pixels. We compute HOG feature maps from rescaled images of five scales {480, 576, 688, 864, 1200}. The proposal regions are generated using Selective Search [27]. During training, a scale is randomly selected for each proposal region; and for inference, a proposal region is mapped to the “closest” scale where the rescaled region has an area closest to  $60 \times 60$  on the feature map. For each region, we max pool the features to a spatial resolution of  $m \times m$ , where  $m = 24$ . The resulting  $24 \times 24 \times 31$ -d features are the input to NoCs. Training and inference are analogous to those in the previous sections.

We investigate a maxout NoC structure of c96s2-mo-c256s2-f4096-f21 that consists of two conv and two fc layers. The first conv layer has  $5 \times 5$  spatial filters, and the second has  $3 \times 3$ . The notation “s2” means a stride of 2 is used. Before the first fc layer, the features are pooled using SPP into a pyramid of spatial resolutions { $6 \times 6$ ,  $3 \times 3$ ,  $2 \times 2$ ,  $1 \times 1$ }, concatenated, and fed into the fc layers. This NoC is randomly initialized and trained on VOC 07 trainval, without external data.

The mAP of this network on the VOC 07 test data is 37.0% and 39.4%, respectively, without and with bounding box regression (Table 6). These results compare favorably to DPM [8] and Regionlet [30] trained on HOG features. To our knowledge, this is the best HOG-only result to date for VOC 07 without external data.

This investigation in turn explains the major accuracy gap between the hybrid DPM+CNN methods [24, 11, 29] and R-CNN/SPPnet. The DPM model is a less powerful classifier than a multi-layer network, whether applied to deep convolutional features or shallow HOG features.

The work in [34] shows that a DPM classifier has diminishing improvement when given more training data. To



method	trained on	mAP	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
R-CNN [9]	VOC 07	62.2	71.6	73.5	58.1	42.2	39.4	70.7	76.0	74.5	38.7	71.0	56.9	74.5	67.9	69.6	59.3	35.7	62.1	64.0	66.5	71.2
R-CNN,bb [9]	VOC 07	66.0	73.4	77.0	63.4	45.4	44.6	75.1	78.1	79.8	40.5	73.7	62.2	79.4	78.1	73.1	64.2	35.6	66.8	67.2	70.4	71.1
SPP [13] (3fc)	VOC 07	60.4	69.4	70.4	58.8	47.3	39.2	72.2	70.4	71.5	38.1	70.3	52.8	69.3	69.8	71.1	51.4	33.5	58.5	52.6	67.1	73.8
SPP [13] (3fc)	07+12	64.6	70.8	78.1	65.6	51.0	43.4	74.4	71.2	76.6	43.6	73.8	55.0	76.9	73.8	73.1	55.2	33.7	65.3	65.0	69.4	75.6
NoC	07+12	68.8	74.6	77.7	68.5	53.3	45.5	78.0	75.5	82.1	47.9	77.2	63.0	81.1	75.9	75.1	61.6	41.7	72.9	73.3	73.8	77.7
NoC, bb	07+12	71.6	75.4	79.4	71.9	57.4	50.9	83.0	77.4	<b>85.9</b>	51.3	77.5	65.9	82.8	82.7	77.7	65.2	<b>45.6</b>	70.2	75.7	76.8	<b>78.6</b>
NoC, +EB	07+12	71.8	74.2	79.8	73.0	60.4	59.0	82.0	76.8	82.8	51.3	<b>81.5</b>	63.2	82.3	81.9	<b>78.7</b>	<b>70.0</b>	40.3	<b>73.0</b>	73.2	77.9	75.4
NoC, +EB, bb	07+12	<b>73.3</b>	<b>76.3</b>	<b>81.4</b>	<b>74.4</b>	<b>61.7</b>	<b>60.8</b>	<b>84.7</b>	<b>78.2</b>	82.9	<b>53.0</b>	79.2	<b>69.2</b>	<b>83.2</b>	<b>83.2</b>	78.5	68.0	45.0	71.6	<b>76.7</b>	<b>82.2</b>	75.7

Table 7. Detection results for **PASCAL VOC 2007** test set using the VGG-16 model [26]. Here “bb” denotes bounding box regression [9]. EB denotes additional EdgeBoxes [35] for region proposal. The NoC used is c512-mo-c512-f4096-f4096-f21.

understand the impact of the dataset scale for an NoC classifier, we train the above HOG-only model using VOC 07+12 trainval data. The mAP increases significantly from 39.4% to 46.1%. The NoC as a region classifier benefits from the amount of training data, similar to image classifier networks [18, 32]. This result suggests that the role of larger-scale data for object detection is not merely on learning generic features, but also on improving object classifiers.

## 7. Implementation

The implementation details follow those in [9, 13]. The proposal regions are extracted by the “fast” mode of Selective Search [27]. The convolutional feature maps are computed from multiple scales [13]. We use seven scales where the shorter side of the rescaled image is in  $\{224, 360, 480, 576, 688, 864, 1200\}$ . For NoC training, a scale is randomly selected for each region at each iteration, and the feature is computed using this scale; if maxout is used, two adjacent scales are randomly selected. For inference, a scale (or two scales for maxout) is selected when the rescaled proposal region has an area closest to  $224 \times 224$  pixels.

To compute feature maps, we pad  $\lfloor p/2 \rfloor$  pixels for a layer with a filter size of  $p$ . As such, for a response centered at  $(x', y')$ , its effective receptive field in the image domain is centered at  $(x, y) = (16x', 16y')$  where 16 is the effective stride in ZF or VGG models. Given a window in the image, we project the left (top) boundary to the conv feature map by:  $x' = \lfloor x/16 \rfloor + 1$  and the right (bottom) boundary  $x' = \lceil x/16 \rceil - 1$ .

For NoC training, we use a mini-batch size of 128. We start with a learning rate of 0.001, and divide by 10 twice after each 50k iterations. When the NoC weights are randomly initialized, we use the scaled Gaussian distribution [14]. All fc layers (except the last 21-d fc) are with a dropout [15] ratio of 50%.

**Running Time.** A prominent behavior of the NoC is that it benefits considerably from larger-scale training data and more region proposals. This is made feasible by the

shared computation of convolutional feature maps as in other feature-map-based methods [13, 24, 11, 29], because the region-wise computation is kept low. Using a VGG-16 model, it takes our method 3.9 seconds evaluating an image. This is reasonably slower than SPPnet (using VGG-16) that takes 2.3 seconds per image, but is over one order of magnitude faster than R-CNN that takes 48 seconds. The running time is averaged on 100 random PASCAL images, and is evaluated by a single Nvidia K40 GPU and using 2,000 proposal windows (proposal time not counted). For fine-tuning, our method takes about one day caching feature maps using the single K40 GPU and less than one day training the NoC for 150k iterations.

## 8. Results

By exploiting NoCs and the pre-trained very deep VGG models [26], we achieve state-of-the-art results in PASCAL VOC 2007 and 2012. In this section, we compare our method with previous leading methods.

Table 7 shows the comparisons on PASCAL VOC 2007, all using the VGG-16 model [26]. R-CNN fine-tuned on VOC 07 trainval has 62.2% mAP without bounding box regression. The SPPnet method has 60.4% mAP under the same setting, which is 1.8% lower than R-CNN. This suggests that fine-tuning all layers of a very deep model (as in R-CNN) improves accuracy compared with using fixed pre-trained convolutional layers (as in SPPnet).

The SPPnet method has 64.6% mAP when fine-tuned on VOC 07+12. Our NoC method improves the mAP to 68.8% under the same settings. This is a 4.2% increase due to a carefully designed NoC as a region classifier.

With bounding box (bbox) regression [9] for post-processing, the mAP is 71.6%. We use the features of the last conv layer (conv<sub>+2</sub>) for bbox regression. The bbox regression boosts the mAP by 2.8%. This gain, however, is smaller than the gain of bbox regression for R-CNN (3.8%). This is perhaps because the bbox regression technique is for reducing localization error, which is also the main improvement of NoCs (Fig. 3).

method	trained on	mAP	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
R-CNN [9]	VOC 12	59.2	76.8	70.9	56.6	37.5	36.9	62.9	63.6	81.1	35.7	64.3	43.9	80.4	71.6	74.0	60.0	30.8	63.4	52.0	63.5	58.7
R-CNN, bb [9]	VOC 12	62.4	79.6	72.7	61.9	41.2	41.9	65.9	66.4	84.6	38.5	67.2	46.7	82.0	74.8	76.0	65.2	35.6	65.4	54.2	67.4	60.3
BabyLearning, bb [20]	VOC+ext.	63.2	78.0	74.2	61.3	45.7	42.7	68.2	66.8	80.2	40.6	70.0	49.8	79.0	74.5	77.9	64.0	35.3	67.9	55.7	68.7	62.6
NUS_NIN_c2000, bb [6]	unknown	63.8	80.2	73.8	61.9	43.7	43.0	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	38.6	68.3	58.0	68.7	63.3
R-CNN+Str+FGS, bb [33]	VOC 12	66.4	<b>82.9</b>	76.1	64.1	44.6	49.4	70.3	<b>71.2</b>	84.6	42.7	68.6	<b>55.8</b>	82.7	77.1	<b>79.9</b>	68.7	<b>41.4</b>	69.0	<b>60.0</b>	72.0	66.2
NoC, +EB	VOC 07+12	67.6	82.5	77.8	71.3	50.0	50.2	<b>74.4</b>	67.8	84.6	44.8	74.2	50.5	84.9	80.1	79.8	71.0	37.8	72.0	57.2	73.8	67.1
NoC, +EB, bb	VOC 07+12	<b>68.8</b>	82.8	<b>79.0</b>	<b>71.6</b>	<b>52.3</b>	<b>53.7</b>	74.1	69.0	<b>84.9</b>	<b>46.9</b>	<b>74.3</b>	53.1	<b>85.0</b>	<b>81.3</b>	79.5	<b>72.2</b>	38.9	<b>72.4</b>	59.5	<b>76.7</b>	<b>68.1</b>

Table 8. Detection results for **PASCAL VOC 2012** test set reported by the evaluation server. The entries of R-CNN use pre-trained VGG models. The entries of BabyLearning, NUS\_NIN\_c2000, and [33] are based on the R-CNN framework.

Data augmentation is a standard method for improving ConvNet accuracy for image classification [18]. NoCs also shows improved accuracy if the region data source is augmented. A simple approach is to use two complementary object proposal algorithms. We augment the Selective Search proposal regions by EdgeBoxes [35]. We use the top-ranked 2,000 proposals generated by the default setting of EdgeBoxes (for a threshold of 0.7). These regions, combined with 2k Selective Search proposals, are used for training and inference.

The result using EdgeBoxes is increased by 1.7% from 71.6% to **73.3%**. Our result is 4.8% higher than the previous best result published recently (68.5% by [33]). The method in [33] trains an R-CNN model with a structured loss, which is complementary to our method and can in principle be combined with NoCs.

We also evaluate our model on the PASCAL VOC 2012 test set. In Table 8 we compare our results with the state-of-the-art on this set. Our result of NoC is **68.8%**, which is 2.4% higher than the previous best result (66.4%, [33]).

Lastly, it is interesting to evaluate our object detection model on the PASCAL VOC 2012 *classification* benchmark. Without training any new model, we use the above model to generate predictions of object bounding boxes with their scores. For each category, we use the highest score in the predictions as the category score of an image. This simple strategy leads to 90.6% mAP in the PASCAL VOC 2012 classification test set, on par with 90.3% reported by [31] which also exploits ground-truth bounding box annotations for training. On the other hand, the VGG-16 model leads to 89.0% mAP [26] by fine-tuning without using bounding boxes. Our result suggests that a superior object detector can perform favorably for multi-label image classification.

## 9. Conclusion and Future Work

In this work, we delve into the detection pipeline and provide insights about the importance of each component. We discover that deep classifiers are just as important as deep feature extractors. Randomly initialize deep classi-

fiers showcase excellent results, on both deep pre-trained features and shallow HOG features, suggesting that fine-tuning is not the sole strategy of designing object detectors. We also find that deep classifiers (on conv/HOG features) favorably benefit from the increasing amount of detection training data, indicating that large-scale data are not merely useful for learning generic features, but also for learning object detectors.

As discussed in Sec. 5.4, we conjecture that an end-to-end training of deep features plus deep object classifiers may produce better results. As such, the fine-tuning procedure will behave like that of R-CNN, but the classifiers on feature maps will still enjoy the efficiency of shared features. We plan to investigate this direction in the future.

## References

- [1] P. Agrawal, R. Girshick, and J. Malik. Analyzing the performance of multilayer neural networks for object recognition. In *ECCV*, 2014.
- [2] T. Ahonen, A. Hadid, and M. Pietikäinen. Face recognition with local binary patterns. In *ECCV*, 2004.
- [3] J. Dai, K. He, and J. Sun. Convolutional feature masking for joint object and stuff segmentation. In *CVPR*, 2015.
- [4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.
- [5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.
- [6] J. Dong, Q. Chen, S. Yan, and A. Yuille. Towards unified object detection and semantic segmentation. In *ECCV*, 2014.
- [7] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) Challenge. *IJCV*, 2010.
- [8] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *TPAMI*, 2010.
- [9] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.
- [10] R. Girshick, P. Felzenszwalb, and D. McAllester. Discriminatively trained deformable part models, release 5, 2012.



- [11] R. Girshick, F. Iandola, T. Darrell, and J. Malik. Deformable part models are convolutional neural networks. In *CVPR*, 2015.
- [12] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. *arXiv:1302.4389*, 2013.
- [13] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014.
- [14] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *arXiv:1502.01852*, 2015.
- [15] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*, 2012.
- [16] D. Hoiem, Y. Chodpathumwan, and Q. Dai. Diagnosing error in object detectors. In *ECCV*, 2012.
- [17] K. Hornik, M. Stinchcombe, and H. White. Multilayer feed-forward networks are universal approximators. *Neural networks*, 1989.
- [18] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.
- [19] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.
- [20] X. Liang, S. Liu, Y. Wei, L. Liu, L. Lin, and S. Yan. Computational baby learning. *arXiv:1411.2861*, 2014.
- [21] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.
- [22] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004.
- [23] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.
- [24] P.-A. Savalle, S. Tsogkas, G. Papandreou, and I. Kokkinos. Deformable part models with CNN features. In *Parts and Attributes Workshop, ECCV*, 2014.
- [25] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014.
- [26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.
- [27] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *IJCV*, 2013.
- [28] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001.
- [29] L. Wan, D. Eigen, and R. Fergus. End-to-end integration of a convolutional network, deformable parts model and non-maximum suppression. *arXiv:1411.5309*, 2014.
- [30] X. Wang, M. Yang, S. Zhu, and Y. Lin. Regionlets for generic object detection. In *ICCV*, 2013.
- [31] Y. Wei, W. Xia, J. Huang, B. Ni, J. Dong, Y. Zhao, and S. Yan. CNN: single-label to multi-label. *arXiv:1406.5726*, 2014.
- [32] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional neural networks. In *ECCV*, 2014.
- [33] Y. Zhang, K. Sohn, R. Villegas, G. Pan, and H. Lee. Improving object detection with deep convolutional networks via bayesian optimization and structured prediction. In *CVPR*, 2015.
- [34] X. Zhu, C. Vondrick, D. Ramanan, and C. Fowlkes. Do we need more training data or better models for object detection? In *BMVC*, 2012.
- [35] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014.
- [36] W. Y. Zou, X. Wang, M. Sun, and Y. Lin. Generic object detection with dense neural patterns and regionlets. In *BMVC*, 2014.