Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition

Kaiming He[1], Xiangyu Zhang[2⋆], Shaoqing Ren[3⋆], and Jian Sun[1]

[1]Microsoft Research [2]Xi'an Jiaotong University
[3]University of Science and Technology of China

Abstract. Existing deep convolutional neural networks (CNNs) require a fixed-size (*e.g.* 224×224) input image. This requirement is "artificial" and may hurt the recognition accuracy for the images or sub-images of an arbitrary size/scale. In this work, we equip the networks with a more principled pooling strategy, "spatial pyramid pooling", to eliminate the above requirement. The new network structure, called SPP-net, can generate a fixed-length representation regardless of image size/scale. By removing the fixed-size limitation, we can improve all CNN-based image classification methods in general. Our SPP-net achieves state-of-the-art accuracy on the datasets of ImageNet 2012, Pascal VOC 2007, and Caltech101.

The power of SPP-net is more significant in object detection. Using SPP-net, we compute the feature maps from the entire image only once, and then pool features in arbitrary regions (sub-images) to generate fixed-length representations for training the detectors. This method avoids repeatedly computing the convolutional features. In processing test images, our method computes convolutional features 30-170× faster than the recent leading method R-CNN (and 24-64× faster overall), while achieving better or comparable accuracy on Pascal VOC 2007.[1]

1 Introduction

We are witnessing a rapid, revolutionary change in our vision community, mainly caused by deep convolutional neural networks (CNNs) [18] and the availability of large scale training data [6]. Deep-networks-based approaches have recently been substantially improving upon the state of the art in image classification [16, 31, 24], object detection [12, 33, 24], many other recognition tasks [22, 27, 32, 13], and even non-recognition tasks.

However, there is a technical issue in the training and testing of the CNNs: the prevalent CNNs require a *fixed* input image size (*e.g.*, 224×224), which limits both the aspect ratio and the scale of the input image. When applied to images of arbitrary sizes, current methods mostly fit the input image to the fixed size, either via cropping [16, 31] or via warping [7, 12], as shown in Fig. 1 (top). But the cropped region may not contain the entire object, while the warped

⋆ This work was done when X. Zhang and S. Ren were interns at Microsoft Research.
[1] A longer technical report of our paper is in http://arxiv.org/abs/1406.4729v1.pdf

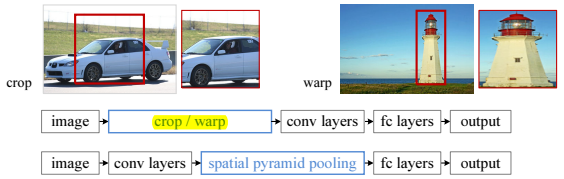

Fig. 1. Top: cropping or warping to fit a fixed size. Middle: a conventional deep convolutional network structure. Bottom: our spatial pyramid pooling network structure.

content may result in unwanted geometric distortion. Recognition accuracy can be compromised due to the content loss or distortion. Besides, a pre-defined scale (*e.g.*, 224) may not be suitable when object scales vary. Fixing the input size overlooks the issues involving scales.

So why do CNNs require a fixed input size? A CNN mainly consists of two parts: convolutional layers, and fully-connected layers that follow. The convolutional layers operate in a sliding-window manner and output feature maps which represent the spatial arrangement of the activations (Fig. 2). In fact, convolutional layers do not require a fixed image size and can generate feature maps of any sizes. On the other hand, the fully-connected layers need to have fixed-size/length input by their definition. Hence, the fixed-size constraint comes only from the fully-connected layers, which exist at a deeper stage of the network.

In this paper, we introduce a *spatial pyramid pooling* (SPP) [14, 17] layer to remove the fixed-size constraint of the network. Specifically, we add an SPP layer on top of the last convolutional layer. The SPP layer pools the features and generates fixed-length outputs, which are then fed into the fully-connected layers (or other classifiers). In other words, we perform some information "aggregation" at a deeper stage of the network hierarchy (between convolutional layers and fully-connected layers) to avoid the need for cropping or warping at the beginning. Fig. 1 (bottom) shows the change of the network architecture by introducing the SPP layer. We call the new network structure *SPP-net*.

We believe that aggregation at a deeper stage is more physiologically sound and more compatible with the hierarchical information processing in our brains. When an object comes into our field of view, it is more reasonable that our brains consider it as a whole instead of cropping it into several "views" at the beginning. Similarly, it is unlikely that our brains distort all object candidates into fixed-size regions for detecting/locating them. It is more likely that our brains handle arbitrarily-shaped objects at some deeper layers, by aggregating the already deeply processed information from the previous layers.

Spatial pyramid pooling [14, 17] (popularly known as spatial pyramid matching or SPM [17]), as an extension of the Bag-of-Words (BoW) model [25], is one of the most successful methods in computer vision. It partitions the image into divisions from finer to coarser levels, and aggregates local features in them. SPP has long been a key component in the leading and competition-winning systems for classification (*e.g.*, [30, 28, 21]) and detection (*e.g.*, [23]) before the recent

Fig. 2. Visualization of the feature maps. (a) Two images in Pascal VOC 2007. (b) The feature maps of some conv_5 (the fifth convolutional layer) filters. The arrows indicate the strongest responses and their corresponding positions in the images. (c) The ImageNet images that have the strongest responses of the corresponding filters. The green rectangles mark the receptive fields of the strongest responses.

prevalence of CNNs. Nevertheless, SPP has not been considered in the context of CNNs. We note that SPP has several remarkable properties for deep CNNs: 1) SPP is able to generate a fixed-length output regardless of the input size, while the sliding window pooling used in the previous deep networks [16] cannot; 2) SPP uses multi-level spatial bins, while the sliding window pooling uses only a single window size. Multi-level pooling has been shown to be robust to object deformations [17]; 3) SPP can pool features extracted at variable scales thanks to the flexibility of input scales. Through experiments we show that all these factors elevate the recognition accuracy of deep networks.

The flexibility of SPP-net makes it possible to generate a full-image representation for testing. Moreover, it also allows us to feed images with varying sizes or scales during training, which increases scale-invariance and reduces the risk of over-fitting. We develop a simple multi-size training method to exploit the properties of SPP-net. Through a series of controlled experiments, we demonstrate the gains of using multi-level pooling, full-image representations, and variable scales. On the ImageNet 2012 dataset, our network reduces the top-1 error by 1.8% compared to its counterpart without SPP. The fixed-length representations given by this pre-trained network are also used to train SVM classifiers on other datasets. Our method achieves 91.4% accuracy on Caltech101 [9] and 80.1% mean Average Precision (mAP) on Pascal VOC 2007 [8] using only a *single* full-image representation (single-view testing).

SPP-net shows even greater strength in object detection. In the leading object detection method R-CNN [12], the features from candidate windows are extracted via deep convolutional networks. This method shows remarkable detection accuracy on both the VOC and ImageNet datasets. But the feature computation in R-CNN is time-consuming, because it repeatedly applies the deep convolutional networks to the raw pixels of thousands of warped regions per image. In this paper, we show that we can run the convolutional layers only *once* on the entire image (regardless of the number of windows), and then extract features by SPP-net on the feature maps. This method yields a speedup of over one hundred times over R-CNN. Note that training/running a detector on the

feature maps (rather than image regions) is actually a more popular idea [10, 5, 23, 24]. But SPP-net inherits the power of the deep CNN feature maps and also the flexibility of SPP on arbitrary window sizes, which leads to outstanding accuracy and efficiency. In our experiment, the SPP-net-based system (built upon the R-CNN pipeline) computes convolutional features 30-170× faster than R-CNN, and is overall 24-64× faster, while has better or comparable accuracy. We further propose a simple model combination method to achieve a new state-of-the-art result (mAP 60.9%) on the Pascal VOC 2007 detection task.

2 Deep Networks with Spatial Pyramid Pooling

2.1 Convolutional Layers and Feature Maps

Consider the popular seven-layer architectures [16, 31]. The first five layers are convolutional, some of which are followed by pooling layers. These pooling layers can also be considered as "convolutional", in the sense that they are using sliding windows. The last two layers are fully connected, with an N-way softmax as the output, where N is the number of categories.

The deep network described above needs a fixed image size. However, we notice the requirement of fixed sizes is only due to the fully-connected layers that demand fixed-length vectors as inputs. On the other hand, the convolutional layers accept inputs of arbitrary sizes. The convolutional layers use sliding filters, and their outputs have roughly the same aspect ratio as the inputs. These outputs are known as *feature maps* [18] - they involve not only the strength of the responses, but also their spatial positions. In Fig. 2, we visualize some feature maps. They are generated by some filters of the conv_5 layer.

It is worth noticing that we generate the feature maps in Fig. 2 without fixing the input size. These feature maps generated by deep convolutional layers are analogous to the feature maps in traditional methods [2, 4]. In those methods, SIFT vectors [2] or image patches [4] are densely extracted and then encoded, *e.g.*, by vector quantization [25, 17, 11], sparse coding [30, 28], or Fisher kernels [21]. These encoded features consist of the feature maps, and are then pooled by Bag-of-Words (BoW) [25] or spatial pyramids [14, 17]. Analogously, the deep convolutional features can be pooled in a similar way.

2.2 The Spatial Pyramid Pooling Layer

The convolutional layers accept arbitrary input sizes, but they produce outputs of variable sizes. The classifiers (SVM/softmax) or fully-connected layers require fixed-length vectors. Such vectors can be generated by the Bag-of-Words (BoW) approach [25] that pools the features together. Spatial pyramid pooling [14, 17] improves BoW in that it can maintain spatial information by pooling in local spatial bins. These spatial bins have sizes proportional to the image size, so the number of bins is fixed regardless of the image size. This is in contrast to the sliding window pooling of the previous deep networks [16], where the number of sliding windows depends on the input size.

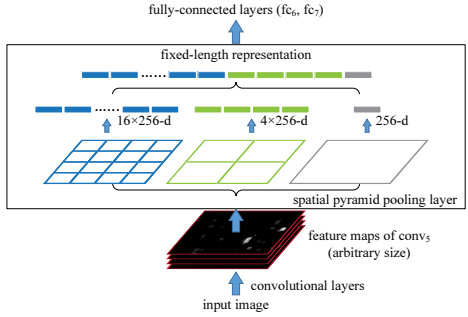

Fig. 3. The network structure with a **spatial pyramid pooling layer**.

To adopt the deep network for images of arbitrary sizes, we replace the pool_5 layer (the pooling layer after conv_5) with a *spatial pyramid pooling layer*. Fig. 3 illustrates our method. In each spatial bin, we pool the responses of each filter (throughout this paper we use max pooling). The outputs of SPP are $256M$-d vectors with the number of bins denoted as M (256 is the number of conv_5 filters). The fixed-dimensional vectors are the input to the fc layer (fc_6).

With SPP, the input image can be of any sizes; this not only allows arbitrary aspect ratios, but also allows arbitrary scales. We can resize the input image to any scale (*e.g.*, $\min(w, h)$=180, 224, ...) and apply the same deep network. When the input image is at different scales, the network (with the same filter sizes) will extract features at different scales. The scales play important roles in traditional methods, *e.g.*, the SIFT vectors are often extracted at multiple scales [19, 2] (determined by the sizes of the patches and Gaussian filters). We will show that the scales are also important for the accuracy of deep networks.

2.3 Training the Network with the Spatial Pyramid Pooling Layer

Theoretically, the above network structure can be trained with standard back-propagation [18], regardless of the input image size. But in practice the GPU implementations (such as *convnet* [16] and *Caffe* [7]) are preferably run on fixed input images. Next we describe our training solution that takes advantage of these GPU implementations while still preserving the SPP behaviors.

Single-size training As in previous works, we first consider a network taking a fixed-size input (224×224) cropped from images. The cropping is for the purpose of data augmentation. For an image with a given size, we can pre-compute the bin sizes needed for spatial pyramid pooling. Consider the feature maps after conv_5 that have a size of $a \times a$ (*e.g.*, 13×13). With a pyramid level of $n \times n$ bins, we implement this pooling level as a sliding window pooling, where the window size $win = \lceil a/n \rceil$ and stride $str = \lfloor a/n \rfloor$ with $\lceil \cdot \rceil$ and $\lfloor \cdot \rfloor$ denoting ceiling and floor operations. With an l-level pyramid, we implement l such layers. The next fc

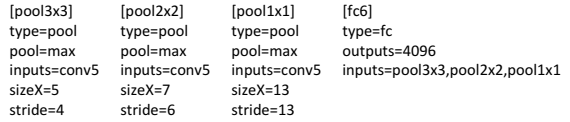

Fig. 4. An example 3-level pyramid pooling in the convnet style [16]. Here sizeX is the size of the pooling window. This is for a network whose feature map size of conv_5 is 13×13, so $\text{pool}_{3\times3}$, $\text{pool}_{2\times2}$, and $\text{pool}_{1\times1}$ will have 3×3, 2×2, and 1×1 bins respectively.

layer (fc_6) will concatenate the l outputs. Fig. 4 shows an example configuration of 3-level pyramid pooling (3×3, 2×2, 1×1) in the *convnet* style [16].

The main purpose of our single-size training is to enable the multi-level pooling behavior. Experiments show that this is one reason for the gain of accuracy.

Multi-size training Our network with SPP is expected to be applied on images of any sizes. To address the issue of varying image sizes in training, we consider a set of pre-defined sizes. We use two sizes (180×180 in addition to 224×224) in this paper. Rather than crop a smaller 180×180 region, we resize the aforementioned 224×224 region to 180×180. So the regions at both scales differ only in resolution but not in content/layout. For the network to accept 180×180 inputs, we implement another fixed-size-input (180×180) network. The feature map size after conv_5 is $a\times a = 10\times10$ in this case. Then we still use $win = \lceil a/n \rceil$ and $str = \lfloor a/n \rfloor$ to implement each pyramid level. The output of the SPP layer of this 180-network has the same fixed length as the 224-network. As such, this 180-network has exactly the same parameters as the 224-network in each layer. In other words, during training we implement the varying-size-input SPP-net by two fixed-size-input networks that share parameters.

To reduce the overhead to switch from one network (*e.g.*, 224) to the other (*e.g.*, 180), we train each full epoch on one network, and then switch to the other one (copying all weights) for the next full epoch. This is iterated. In experiments, we find the convergence rate of this multi-size training to be similar to the above single-size training. We train 70 epochs in total as is a common practice.

The main purpose of multi-size training is to simulate the varying input sizes while still leveraging the existing well-optimized fixed-size implementations. In theory, we could use more scales/aspect ratios, with one network for each scale/aspect ratio and all networks sharing weights, or we could develop a varying-size implementation to avoid switching. We will study this in the future.

Note that the above single/multi-size solutions are for training only. At the testing stage, it is straightforward to apply SPP-net on images of any sizes.

3 SPP-net for Image Classification

3.1 Experiments on ImageNet 2012 Classification

We trained our network on the 1000-category training set of ImageNet 2012. Our training details follow the practices of previous work [16, 31, 15]. The images are

Table 1. Error rates in the validation set of ImageNet 2012. All the results are based on **a single network**. The number of views in Overfeat depends on the scales and strides, for which there are several hundreds at the finest scale

	method	test scale	test views	top-1 val	top-5 val
(a)	Krizhevsky *et al.*[16]	1	10	40.7	18.2
(b1)	Overfeat (fast) [24]	1	-	39.01	16.97
(b2)	Overfeat (fast) [24]	6	-	38.12	16.27
(b3)	Overfeat (big) [24]	4	-	35.74	14.18
(c1)	Howard (base) [15]	3	162	37.0	15.8
(c2)	Howard (high-res) [15]	3	162	36.8	16.2
(d1)	Zeiler & Fergus (ZF) (fast) [31]	1	10	38.4	16.5
(d2)	Zeiler & Fergus (ZF) (big) [31]	1	10	37.5	16.0
(e1)	our impl of ZF (fast)	1	10	35.99	14.76
(e2)	SPP-net$_4$, single-size trained	1	10	35.06	14.04
(e3)	SPP-net$_6$, single-size trained	1	10	34.98	14.14
(e4)	SPP-net$_6$, multi-size trained	1	10	34.60	13.64
(e5)	SPP-net$_6$, multi-size trained	1	8+2full	**34.16**	**13.57**

Table 2. Error rates in the validation set of ImageNet 2012 using a single view. The images are resized so $\min(w, h) = 256$. The crop view is the central 224×224

method	test view	top-1 val
SPP-net$_6$, single-size trained	1 crop	38.01
SPP-net$_6$, single-size trained	1 full	**37.55**
SPP-net$_6$, multi-size trained	1 crop	37.57
SPP-net$_6$, multi-size trained	1 full	**37.07**

resized so that the smaller dimension is 256, and a 224×224 crop is picked from the center or the four corners from the entire image[2]. The data are augmented by horizontal flipping and color altering [16]. Dropout [16] is used on the two fully-connected layers. The learning rate starts from 0.01, and is divided by 10 (twice) when the error plateaus. Our implementation is based on the publicly available code of *convnet* [16]. Our experiments are run on a GTX Titan GPU.

As our baseline model, we implement the 7-layer network of Zeiler and Fergus's (ZF) "fast" (smaller) model [31], which produces competitive results with a moderate training time (two weeks). The filter numbers (sizes) of the five convolutional layers are: $96(7 \times 7)$, $256(5 \times 5)$, $384(3 \times 3)$, $384(3 \times 3)$, and $256(3 \times 3)$. The first two layers have a stride of 2, and the rest have a stride of 1. The first two layers are followed by (sliding window) max pooling with a stride of 2, window size of 3, and contrast normalization operations. The outputs of the two fully-connected layers are 4096-d. At the testing stage, we follow the standard 10-view prediction in [16]: each view is a 224×224 crop and their scores are averaged. Our replication of this network gives 35.99% top-1 error (Tab. 1 (e1)), better than 38.4% (Tab. 1 (d1)) as reported in [31]. We believe this margin is because the corner crops are from the entire image (rather than from the corners of the central 256×256 square), as is reported in [15].

[2] In [16], the four corners are picked from the corners of the central 256×256 crop.

Tab. 1 (e2)(e3) show our results using single-size training. The training and testing sizes are both 224×224. In these networks, the convolutional layers have the same structures as the ZF fast model, whereas the pooling layer after $conv_5$ is replaced with the SPP layer. We use a 4-level pyramid. The pyramid is {4×4, 3×3, 2×2, 1×1}, totally 30 bins and denoted as SPP-net_4 (e2), or {6×6, 3×3, 2×2, 1×1}, totally 50 bins and denoted as SPP-net_6 (e3). In these results, we use 10-view prediction with each view a 224×224 crop. The top-1 error of SPP-net_4 is 35.06%, and of SPP-net_6 is 34.98%. These results show considerable improvement over the no-SPP counterpart (e1). Since we are still using the same 10 cropped views as in (e1), this gain is solely because of multi-level pooling. Note that SPP-net_4 has even fewer parameters than the no-SPP model (fc_6 has 30×256-d inputs instead of 36×256-d). So the gain of the multi-level pooling is **not** simply due to more parameters. Rather, it is because the multi-level pooling is robust to the variance in object deformations and spatial layout [17].

Tab. 1 (e4) shows our result using multi-size training. The training sizes are 224 and 180, while the testing size is still 224. In (e4) we still use the 10 cropped views for prediction. The top-1 error drops to 34.60%. Note the networks in (e3) and (e4) have exactly the same structure and the same method for testing. So the gain is solely because of the multi-size training.

Next we investigate the accuracy of the full-image views. We resize the image so that $\min(w, h)$=256 while maintaining its aspect ratio. The SPP-net is applied on this full image to compute the scores of the full view. For fair comparison, we also evaluate the accuracy of the single view in the center 224×224 crop (which is used in the above evaluations). The comparisons of single-view testing accuracy are in Tab. 2. The top-1 error rates are reduced by about 0.5%. This shows the importance of maintaining the complete content. Even though our network is trained using square images only, it generalizes well to other aspect ratios.

In Tab. 1 (e5), we replace the two center cropped views with two full-views (with flipping) for testing. The top-1 error is further reduced to 34.16%. This again indicates that the full-image views are more representative than the cropped views[3]. The SPP-net in (e5) is better than the no-SPP counterpart (e1) by 1.8%.

There are previous CNN solutions [24, 15] that deal with various scales/sizes, but they are based on model averaging. In Overfeat [24] and Howard's method [15], the single network is applied at multiple scales in the testing stage, and the scores are averaged. Howard further trains two different networks on low/high-resolution image regions and averages the scores. These methods generate much more views (*e.g.*, over hundreds), but the sizes of the views are still pre-defined beforehand. On the contrary, our method builds the SPP structure into the network, and uses multi-size images to train a single network. Our method also enables the use of full-view as a single image representation.

Our results can be potentially improved further. The usage of the SPP layer does not depend on the design of the convolutional layers. So our method may benefit from, *e.g.*, increased filter numbers or smaller strides [31, 24]. Multiple-model averaging also may be applied. We will study these in the future.

[3] However, the combination of the 8 cropped views is still useful.

Table 3. Classification mAP in Pascal VOC 2007

model size	(a) plain net crop 224×224	(b) SPP-net crop 224×224	(c) SPP-net full 224×-	(d) SPP-net full 392×-
$conv_4$	59.96	57.28	-	-
$conv_5$	66.34	65.43	-	-
$pool_5$ (6×6)	69.14	68.76	70.82	71.67
fc_6	74.86	75.55	77.32	78.78
fc_7	75.90	76.45	78.39	**80.10**

Table 4. Classification accuracy in Caltech101

model size	(a) plain net 224×224 crop	(b) SPP-net 224×224 crop	(c) SPP-net 224×- full
$conv_4$	80.12	81.03	-
$conv_5$	84.40	83.76	-
$pool_5$ (6×6)	87.98	87.60	89.46
SPP $pool_5$	-	89.47	**91.44**
fc_6	87.86	88.54	89.50
fc_7	85.30	86.10	87.08

3.2 Experiments on Pascal VOC 2007 Classification

With the networks pre-trained on ImageNet, we extract representations from the images in other datasets and re-train SVM classifiers [1] for the new datasets. In the SVM training, we intentionally do not use any data augmentation (flip/multi-view). We l_2-normalize the features and fix the SVM's soft margin parameter C to 1. We use our multi-size trained model in Tab. 1 (e5).

The classification task in Pascal VOC 2007 [8] involves 9,963 images in 20 categories. 5,011 images are for training, and the rest are for testing. The performance is evaluated by mAP. Tab. 3 summarizes our results for different settings.

We start from a baseline in Tab. 3 (a). The model is the one in Tab. 1 (e1) without SPP ("plain net"). To apply this model, we resize the image so that $\min(w, h) = 224$, and crop the center 224×224 region. The SVM is trained via the features of a layer. On this dataset, the deeper the layer is, the better the result is. In col.(b), we replace the plain net with our SPP-net. As a first-step comparison, we still apply the SPP-net on the center 224×224 crop. The results of the fc layers improve. This gain is mainly due to multi-level pooling.

Tab. 3 (c) shows our results on the full images which are resized so that $\min(w, h) = 224$. The results are considerably improved (78.39% *vs.* 76.45%). This is due to the full-image representation that maintains the complete content.

Because the usage of our network does not depend on scale, we resize the images so that $\min(w, h) = s$ and use the same network to extract features. We find that $s = 392$ gives the best results (Tab. 3 (d)) based on the validation set. This is mainly because the objects occupy smaller regions in VOC 2007 but larger regions in ImageNet, so the relative object scales are different between the two sets. These results indicate scale matters in the classification tasks, and SPP-net can partially address this "scale mismatch" issue.

Tab. 5 summarizes our results and the comparisons with previous state-of-the-art methods. Among these methods, VQ [17], LCC [28], and FK [21] are all

Table 5. Classification results for Pascal VOC 2007 (mAP) and Caltech101 (accuracy). †numbers reported by [2]. ‡our implementation as in Tab. 3 (a)

method	VOC 2007	Caltech101
VQ [17]†	56.07	74.41±1.0
LLC [28]†	57.66	76.95±0.4
FK [21]†	61.69	77.78±0.6
DeCAF [7]	-	86.91±0.7
Zeiler & Fergus [31]	75.90‡	86.5±0.5
Oquab *et al.*[20]	77.7	-
ours	**80.10**	**91.44±0.7**

based on spatial pyramids matching, and [7, 31, 20] are based on deep networks. Our method outperforms these methods. We note that Oquab *et al.*[20] achieves 77.7% with 500 views per image, whereas we achieve **80.10%** with a single full-image view. Our result may be further improved if data argumentation, multi-view testing, or network fine-tuning is used.

3.3 Experiments on Caltech101

Caltech101 [9] contains 9,144 images in 102 categories (one background). We randomly sample 30 images/category for training and up to 50 images/category for testing. We repeat 10 random splits and average the accuracy.

Tab. 4 summarizes our results. There are some common observations in the Pascal VOC 2007 and Caltech101 results: SPP-net is better than the plain net (Tab. 4 (b) *vs.* (a)), and the full-view representation is better than the crop ((c) *vs.* (b)). But the results in Caltech101 have some differences with Pascal VOC. The fully-connected layers are less accurate, and the pool_5 and SPP layers are better. This is possibly because the object categories in Caltech101 are less related to those in ImageNet, and the deeper layers are more category-specialized. Further, we find that the scale 224 has the best performance among the scales we tested on this dataset. This is mainly because the objects in Caltech101 also occupy large regions of the images, as is the case of ImageNet.

Tab. 5 summarizes our results compared with several previous state-of-the-art methods on Caltech101. Our result (**91.44%**) exceeds the previous state-of-the-art results (86.91%) by a substantial margin (4.5%).

4 SPP-net for Object Detection

Deep networks have been used for object detection. We briefly review the recent state-of-the-art R-CNN method [12]. R-CNN first extracts about 2,000 candidate windows from each image via selective search [23]. Then the image region in each window is warped to a fixed size (227×227). A pre-trained deep network is used to extract the feature of each window. A binary SVM classifier is then trained on these features for detection. R-CNN generates results of compelling quality and substantially outperforms previous methods (30% relative improvement!).

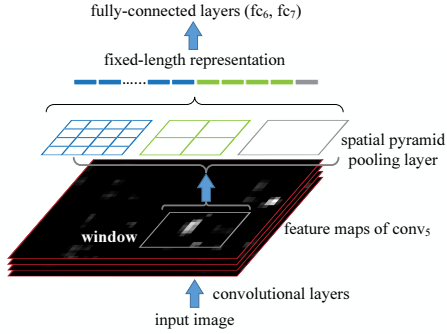

Fig. 5. SPP-net for object detection. The feature maps are computed from the entire image. The pooling is performed in candidate windows.

However, because R-CNN repeatedly applies the deep convolutional network to about 2,000 windows per image, it is time-consuming.

Our SPP-net can also be used for object detection. We extract the feature maps from the entire image only once. Then we apply the spatial pyramid pooling on each candidate window of the feature maps to pool a fixed-length representation of this window (see Fig. 5). Because the time-consuming convolutional network is only applied once, our method can run *orders of magnitude* faster.

Our method extracts window-wise features from regions of the feature maps, while R-CNN extracts directly from image regions. In previous works, the Deformable Part Model (DPM) [10] extracts from windows in HOG [5] feature maps, and Selective Search [23] extracts from windows in encoded SIFT feature maps. The Overfeat detection method [24] also extracts from windows in CNN feature maps, but needs to pre-define the window size. On the contrary, our method enables feature extraction in any windows from CNN feature maps.

4.1 Detection Algorithm

We use the "fast" mode of selective search [23] to generate about 2,000 candidate windows per image. Then we resize the image such that $\min(w, h) = s$, and extract the feature maps of conv_5 from the entire image. We use our pre-trained model of Tab. 1 (e3) for the time being. In each candidate window, we use a 4-level spatial pyramid (1×1, 2×2, 3×3, 6×6, totally 50 bins) to pool the features. This generates a 12,800-d (256×50) representation for each window. These representations are provided to the fully-connected layers of the network. Then we train a binary linear SVM classifier for each category on these features.

Our implementation of the SVM training follows [23, 12]. We use the ground-truth windows to generate the positive samples. The negative samples are those overlapping a positive window by at most 30% . Any negative sample is removed if it overlaps another negative sample by more than 70%. We apply the standard hard negative mining [10] to train the SVM. This step is iterated once. It takes less than 1 hour to train SVMs for all 20 categories. In testing, the classifier is

Table 6. Detection results (mAP) on Pascal VOC 2007. "ft" and "bb" denote fine-tuning and bounding box regression. More details are in our technical report

	SPP (1-sc)	SPP (5-sc)	R-CNN
$pool_5$	43.0	44.9	44.2
fc_6	42.5	44.8	46.2
$ftfc_6$	52.3	53.7	53.1
$ftfc_7$	54.5	55.2	54.2
$ftfc_7$ bb	58.0	**59.2**	58.5
conv time (GPU)	0.053s	0.293s	8.96s
fc time (GPU)	0.089s	0.089s	0.07s
total time (GPU)	0.142s	0.382s	9.03s
speedup (*vs.* RCNN)	**64**×	**24**×	-

used to score the candidate windows. Then we use non-maximum suppression [10] (threshold of 30%) on the scored windows.

Our method can be improved by multi-scale feature extraction. We resize the image such that $\min(w, h) = s \in \mathcal{S} = \{480, 576, 688, 864, 1200\}$, and compute the feature maps of $conv_5$ for each scale. One strategy of combining the features from these scales is to pool them channel-by-channel. But we empirically find that another strategy provides better results. For each candidate window, we choose a single scale $s \in \mathcal{S}$ such that the scaled candidate window has a number of pixels closest to 224×224. Then we only use the feature maps extracted from this scale to compute the feature of this window. If the pre-defined scales are dense enough and the window is approximately square, our method is roughly equivalent to resizing the window to 224×224 and then extracting features from it. Nevertheless, our method only requires computing the feature maps once (at each scale) from the entire image, regardless of the number of candidate windows.

We also fine-tune our pre-trained network, following [12]. Since our features are pooled from the $conv_5$ feature maps from windows of any sizes, for simplicity we only fine-tune the fully-connected layers. In this case, the data layer accepts the fixed-length pooled features after $conv_5$, and the $fc_{6,7}$ layers and a new 21-way (one extra negative category) fc_8 layer follow. The fc_8 weights are initialized with a Gaussian distribution of σ=0.01. We fix all the learning rates to 1e-4 and then adjust to 1e-5 for all three layers. During fine-tuning, the positive samples are those overlapping with a ground-truth window by $[0.5, 1]$, and the negative samples by $[0.1, 0.5)$. In each mini-batch, 25% of the samples are positive. We train 250k mini-batches using the learning rate 1e-4, and then 50k mini-batches using 1e-5. Because we only fine-tune the fc layers, the training is very fast and takes about 2 hours on the GPU. Also following [12], we use bounding box regression to post-process the prediction windows. The features used for regression are the pooled features from $conv_5$ (as a counterpart of the $pool_5$ features used in [12]). The windows used for the regression training are those overlapping with a ground-truth window by at least 50%.

We will release the code to facilitate reproduction of the results[4].

[4] research.microsoft.com/en-us/um/people/kahe/

Table 7. Comparisons of detection results on Pascal VOC 2007

method	mAP	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
DPM [10]	33.7	33.2	60.3	10.2	16.1	27.3	54.3	58.2	23.0	20.0	24.1	26.7	12.7	58.1	48.2	43.2	12.0	21.1	36.1	46.0	43.5
SS [23]	33.8	43.5	46.5	10.4	12.0	9.3	49.4	53.7	39.4	12.5	36.9	42.2	26.4	47.0	52.4	23.5	12.1	29.9	36.3	42.2	48.8
Regionlet [29]	41.7	54.2	52.0	20.3	24.0	20.1	55.5	68.7	42.6	19.2	44.2	49.1	26.6	57.0	54.5	43.4	16.4	36.6	37.7	59.4	52.3
DetNet [26]	30.5	29.2	35.2	19.4	16.7	3.7	53.2	50.2	27.2	10.2	34.8	30.2	28.2	46.6	41.7	26.2	10.3	32.8	26.8	39.8	47.0
RCNN ftfc_7	54.2	64.2	69.7	50.0	41.9	32.0	62.6	71.0	60.7	32.7	58.5	46.5	56.1	60.6	66.8	54.2	31.5	52.8	48.9	57.9	64.7
SPP ftfc_7	55.2	65.5	65.9	51.7	38.4	32.7	62.6	68.6	69.7	33.1	66.6	53.1	58.2	63.6	68.8	50.4	27.4	53.7	48.2	61.7	64.7
RCNN ftfc_7 bb	58.5	68.1	**72.8**	56.8	**43.0**	36.8	**66.3**	**74.2**	67.6	**34.4**	63.5	54.5	61.2	69.1	68.6	**58.7**	**33.4**	**62.9**	**51.1**	62.5	64.8
SPP ftfc_7 bb	**59.2**	**68.6**	69.7	**57.1**	41.2	**40.5**	**66.3**	71.3	**72.5**	**34.4**	**67.3**	**61.7**	**63.1**	**71.0**	**69.8**	57.6	29.7	59.0	50.2	**65.2**	**68.0**

Table 8. Detection results on Pascal VOC 2007 using model combination

method	mAP	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
SPP-net (1)	59.2	**68.6**	69.7	57.1	41.2	40.5	66.3	71.3	72.5	34.4	**67.3**	61.7	63.1	71.0	69.8	57.6	29.7	59.0	50.2	65.2	68.0
SPP-net (2)	59.1	65.7	71.4	57.4	**42.4**	39.9	67.0	71.4	70.6	32.4	66.7	61.7	64.8	71.7	70.4	56.5	30.8	59.9	53.2	63.9	64.6
combination	**60.9**	68.5	**71.7**	**58.7**	41.9	**42.5**	**67.7**	**72.1**	**73.8**	**34.7**	67.0	**63.4**	**66.0**	**72.5**	**71.3**	**58.9**	**32.8**	**60.9**	**56.1**	**67.9**	**68.8**

4.2 Detection Results

We evaluate our method on the detection task of the Pascal VOC 2007 dataset. Tab. 6 shows our results on various layers, by using 1-scale (s=688) or 5-scale. Using the pool_5 layers (in our case the pooled features), our result (44.9%) is comparable with R-CNN's result (44.2%). But using the non-fine-tuned fc_6 layers, our results are inferior. An explanation is that our fc layers are pre-trained using image regions, while in the detection case they are used on the feature map regions. The feature map regions can have strong activations near the window boundaries, while the image regions may not. This difference of usages can be addressed by fine-tuning. Using the fine-tuned fc layers ($\text{ftfc}_{6,7}$), our results are comparable with or slightly better than the fine-tuned results of R-CNN. After bounding box regression, our 5-scale result (**59.2**%) is 0.7% better than R-CNN (58.5%), and our 1-scale result (58.0%) is 0.5% worse. In Tab. 7, we show the results for each category. Our method outperforms R-CNN in 11 categories, and has comparable numbers in two more categories.

In Tab. 7, Selective Search (SS) [23] applies spatial pyramid matching on SIFT feature maps. DPM [10] and Regionlet [29] are based on HOG features [5]. Regionlet improves to 46.1% [33] by combining various features including conv_5. DetectorNet [26] trains a deep network that outputs pixel-wise object masks. This method only needs to apply the deep network once to the entire image, as is the case for our method. But this method has lower mAP (30.5%).

4.3 Complexity and Running Time

Despite having comparable accuracy, our method is much faster than R-CNN. The complexity of the convolutional feature computation in R-CNN is $O(n \cdot 227^2)$ with the window number n ($\sim$2000). This complexity of our method is $O(r \cdot s^2)$ at a scale s, where r is the aspect ratio. Assume r is about 4/3. In

the single-scale version when $s = 688$, this complexity is about 1/160 of R-CNN's; in the 5-scale version, this complexity is about 1/24 of R-CNN's. In Tab. 6, we compare the experimental running time of the convolutional feature computation. The implementation of R-CNN is from the code published by the authors implemented in *Caffe* [7]. For fair comparison, we also implement our feature computation in *Caffe*. In Tab. 6 we evaluate the average time of 100 random VOC images using GPU. R-CNN takes 8.96s per image, while our 1-scale version takes only 0.053s per image. So ours is 170× faster than R-CNN. Our 5-scale version takes 0.293s per image, so is 30× faster than R-CNN.

Our convolutional feature computation is so fast that the computational time of fc layers takes a considerable portion. Tab. 6 shows that the GPU time of computing the 4,096-d fc_7 features (from the conv_5 feature maps) is 0.089s per image. Considering both convolutional and fully-connected features, our 1-scale version is 64× faster than R-CNN and is just 0.5% inferior in mAP; our 5-scale version is 24× faster and has better results. The overhead of the fc computation can be significantly reduced if smaller fc layers are used, *e.g.*, 1,024-d.

We do not consider the window proposal time in the above comparison. The selective search window proposal [23] takes about 1-2 seconds per image on the CPU. There are recent works (*e.g.*, [3]) on reducing window proposal time to milliseconds. We will evaluate this and expect a fast entire system.

4.4 Model Combination for Detection

Model combination is an important strategy for boosting CNN-based classification accuracy [16]. Next we propose a simple model combination method for detection. We pre-train another network in ImageNet, using the same structure but different random initializations. Then we repeat the above detection algorithm. Tab. 8 (SPP-net (2)) shows the results of this network. Its mAP is comparable with the first network (59.1% *vs.* 59.2%), and outperforms the first network in 11 categories. Given the two models, we first use either model to score all candidate windows on the test image. Then we perform non-maximum suppression on the union of the two sets of candidate windows (with their scores). A more confident window given by one method can suppress those less confident given by the other method. After combination, the mAP is boosted to **60.9%** (Tab. 8). In 17 out of all 20 categories the combination performs better than either individual model. This indicates that the two models are complementary.

5 Conclusion

Image scales and sizes are important in visual recognition, but received little consideration in the context of deep networks. We have suggested a solution to train a deep network with an SPP layer. The resulting SPP-net shows outstanding accuracy in classification/detection tasks and greatly accelerates DNN-based detection. Our studies also show that many time-proven techniques/insights in computer vision can still play important roles in deep-networks-based recognition.

References

1. Chang, C.C., Lin, C.J.: Libsvm: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology (TIST) (2011)
2. Chatfield, K., Lempitsky, V., Vedaldi, A., Zisserman, A.: The devil is in the details: an evaluation of recent feature encoding methods. In: BMVC (2011)
3. Cheng, M.M., Zhang, Z., Lin, W.Y., Torr, P.: BING: Binarized normed gradients for objectness estimation at 300fps. In: CVPR (2014)
4. Coates, A., Ng, A.: The importance of encoding versus training with sparse coding and vector quantization. In: ICML (2011)
5. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR (2005)
6. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR (2009)
7. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: Decaf: A deep convolutional activation feature for generic visual recognition. arXiv:1310.1531 (2013)
8. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results (2007)
9. Fei-Fei, L., Fergus, R., Perona, P.: Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. CVIU (2007)
10. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. PAMI (2010)
11. van Gemert, J.C., Geusebroek, J.M., Veenman, C.J., Smeulders, A.W.: Kernel codebooks for scene categorization. In: ECCV (2008)
12. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR (2014)
13. Gong, Y., Wang, L., Guo, R., Lazebnik, S.: Multi-scale orderless pooling of deep convolutional activation features. In: ArXiv:1403.1840 (2014)
14. Grauman, K., Darrell, T.: The pyramid match kernel: Discriminative classification with sets of image features. In: ICCV (2005)
15. Howard, A.G.: Some improvements on deep convolutional neural network based image classification. ArXiv:1312.5402 (2013)
16. Krizhevsky, A., Sutskever, I., Hinton, G.: Imagenet classification with deep convolutional neural networks. In: NIPS (2012)
17. Lazebnik, S., Schmid, C., Ponce, J.: Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: CVPR (2006)
18. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. Neural computation (1989)
19. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. IJCV (2004)
20. Oquab, M., Bottou, L., Laptev, I., Sivic, J., et al.: Learning and transferring mid-level image representations using convolutional neural networks. In: CVPR (2014)
21. Perronnin, F., Sánchez, J., Mensink, T.: Improving the fisher kernel for large-scale image classification. In: ECCV (2010)
22. Razavian, A.S., Azizpour, H., Sullivan, J., Carlsson, S.: Cnn features off-the-shelf: An astounding baseline for recogniton. In: CVPR 2014, DeepVision Workshop (2014)

23. van de Sande, K.E., Uijlings, J.R., Gevers, T., Smeulders, A.W.: Segmentation as selective search for object recognition. In: ICCV (2011)
24. Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y.: Overfeat: Integrated recognition, localization and detection using convolutional networks. arXiv:1312.6229 (2013)
25. Sivic, J., Zisserman, A.: Video google: a text retrieval approach to object matching in videos. In: ICCV (2003)
26. Szegedy, C., Toshev, A., Erhan, D.: Deep neural networks for object detection. In: NIPS (2013)
27. Taigman, Y., Yang, M., Ranzato, M., Wolf, L.: Deepface: Closing the gap to human-level performance in face verification. In: CVPR (2014)
28. Wang, J., Yang, J., Yu, K., Lv, F., Huang, T., Gong, Y.: Locality-constrained linear coding for image classification. In: CVPR (2010)
29. Wang, X., Yang, M., Zhu, S., Lin, Y.: Regionlets for generic object detection. In: ICCV (2013)
30. Yang, J., Yu, K., Gong, Y., Huang, T.: Linear spatial pyramid matching using sparse coding for image classification. In: CVPR (2009)
31. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional neural networks. arXiv:1311.2901 (2013)
32. Zhang, N., Paluri, M., Ranzato, M., Darrell, T., Bourdevr, L.: Panda: Pose aligned networks for deep attribute modeling. In: CVPR (2014)
33. Zou, W.Y., Wang, X., Sun, M., Lin, Y.: Generic object detection with dense neural patterns and regionlets. In: ArXiv:1404.4316 (2014)