The Treasure beneath Convolutional Layers: Cross-convolutional-layer Pooling for Image Classification

Lingqiao Liu[1]*, Chunhua Shen[1,2]*, Anton van den Hengel[1,2]*
The University of Adelaide, Australia[1] The Australian Centre for Robotic Vision[2]
e-mail: `Lingqiao.Liu@adelaide.edu.au`

Abstract

A number of recent studies have shown that a Deep Convolutional Neural Network (DCNN) pretrained on a large dataset can be adopted as a universal image descriptor, and that doing so leads to impressive performance at a range of image classification tasks. Most of these studies, if not all, adopt activations of the fully-connected layer of a DCNN as the image or region representation and it is believed that convolutional layer activations are less discriminative.

This paper, however, advocates that if used appropriately, convolutional layer activations constitute a powerful image representation. This is achieved by adopting a new technique proposed in this paper called cross-convolutional-layer pooling. More specifically, it extracts subarrays of feature maps of one convolutional layer as local features, and pools the extracted features with the guidance of the feature maps of the successive convolutional layer. Compared with existing methods that apply DCNNs in the similar local feature setting, the proposed method avoids the input image style mismatching issue which is usually encountered when applying fully connected layer activations to describe local regions. Also, the proposed method is easier to implement since it is codebook free and does not have any tuning parameters. By applying our method to four popular visual classification tasks, it is demonstrated that the proposed method can achieve comparable or in some cases significantly better performance than existing fully-connected layer based image representations.

1. Introduction

Recently, Deep Convolutional Neural Networks (DCNNs) have attracted a lot of attention in visual recognition, largely due to their performance [1]. It has been discovered that the activation of a DCNN pretrained on a large dataset, such as ImageNet [2], can be employed as a universal image representation, and applying this representation to many visual classification problems delivers impressive performance [3, 4]. This discovery quickly sparked significant interest, and inspired a number of extensions, including [5, 6]. A fundamental issue with this kinds of methods is how to generate an image representation from a pretrained DCNN. Most current solutions, if not all, take activations of the fully connected layer as the image representation. In contrast, activations of convolutional layers are rarely used and some studies [7, 8] have reported that directly using convolutional layer activations as image features produces inferior performance.

In this paper, however, we advocate that convolutional layer activations form a powerful image representation if they are used appropriately. We propose a new method called cross-convolutional layer pooling (or cross layer pooling for short) to derive discriminative features from from convolutional layers. This new technique relies on two crucial components: (1) we utilize convolutional layer activations in a 'local feature' setting in which subarrays of convolutional layer activations are extracted as region descriptors. (2) we pool extracted local features by using activations from two successive convolutional layers.

The first component is motivated by recent work [5, 6, 9] which has shown that DCNN activations are not translation invariant and that it is beneficial to extract fully connected layer activations from a DCNN to describe local regions and create the image representation by pooling multiple regional DCNN activations. Our method steps further to use subarrays of convolutional layer activations, that is, parts of CNN convolutional activations as regional descriptors. Compared with previous work [5, 6], our method avoids the image style mismatching issue which is commonly encountered in existing methods. More specifically, existing methods [5, 6, 9] essentially apply a network trained for representing an image to represent a local region. Thus, the image styles at the test stage do not match those of the training stage. This mismatching may degrade the discriminative power of DCNN activations. In contrast, our method uses

*This work was in part funded by the Data to Decisions Cooperative Research Centre, Australia, and Australian Research Council grants FT120100969 and LP130100156.

the whole image as the network input at both training and testing stages.

The second component is motivated by the parts-based pooling method [10] which was originally proposed for fine-grained image classification. This method creates one pooling channel for each detected part region with the final image representation obtained by concatenating pooling results from multiple channels. We generalize this idea into the context of DCNNs and avoid the need for predefined parts annotation. More specifically, we deem the feature map of each filter in a convolutional layer as the detection response map of a part detector and apply the feature map to weight regional descriptors extracted from previous convolutional layer in the pooling process. The final image representation is obtained by concatenating pooling results from multiple channels with each channel corresponding to one feature map. Note that unlike existing regional-DCNN based methods [5, 6], the proposed method does not need any additional dictionary learning and encoding steps at both training and testing stages. To further reduce the memory usage in storing image representations, we also experiment with a coarse 'feature sign quantization' compression scheme and show that the discriminative power of the proposed representation can be largely maintained after compression.

We conduct extensive experiments on four datasets covering four popular visual classification tasks, that is, scene classification, fine-grained object classification, generic object classification and attribute classification. Experimental results suggest that the proposed method can achieve comparable and in some cases significantly better performance than competitive methods.

Preliminary: Our network structure and model parameters are identical to those in [1], that is, we have five convolutional layers and two fully connected layers. We use conv-1, conv-2, conv-3, conv-4, conv-5, fc-6, fc-7 to denote them respectively. At each convolutional layer, multiple filters are applied and it results in multiple feature maps, one for each filter. In this paper, we use the term 'feature map' to indicate the convolutional result (after applying the ReLU) of one filter and the term 'convolutional layer activations' to indicate feature maps of all filters in a convolutional layer.

2. Existing ways to create image representations from a pretrained DCNN

In literature, there are two major ways of using a pretrained DCNN to create image representations for image classification: (1) directly applying a pretrained DCNN to the input image and extracting its activations as the image representation; (2) applying a pretrained DCNN to the subregions of the input image and aggregating activations from multiple regions as the image representation.

Usually, the first way takes the whole image as the input to a pretrained DCNN and extracts the fc-6/fc-7 activations as the image-level representation. To make the network better adapted to a given task, fine-tuning sometimes is applied. Also, to make this kind of method more robust to image transforms, averaging activations from several jittered versions of the original image, e.g. several slightly shifted versions of the input image, has been employed to obtain better classification performance [4].

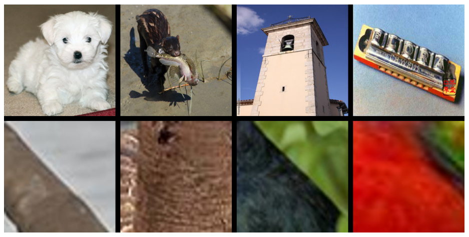
Figure 2: This figure demonstrates the image style mismatch issue when using fully-connected layer activations as regional descriptors. Top row: input images that a DCNN 'sees' at the training stage. Bottom row: input images that a DCNN 'sees' at the test stage.

DCNNs can also be applied to extract local features. It is suggested that DCNN activations are not invariant to a large amount of translation [5] and the performance will be degraded if input images are not well aligned. To handle this issue, it has been suggested to sample multiple regions from an input image and use one DCNN, called regional-DCNN in this scenario, to describe each region. The final image representation is aggregated from activations of those regional-DCNNs [5]. In [5], another layer of unsupervised encoding is employed to create the image-level representation [5, 6]. It is shown that for many visual tasks [5, 6] this kind of method lead to better performance than directly extracting DCNN activations as global features.

One common factor in the above methods is that they all use fully-connnected layer activations as features. The convolutional layer activations are not usually employed and preliminary studies [7, 8] have suggested that the convolutional layer activations have weaker discriminative power than activations of the fully-connected layer.

In image detection, the use of convolutional layers has been recently explored [11, 12]. In these works, the candidate object representations are extracted from the convolutional layer by either directly pooling convolutional feature maps [12] or pooling them by using a spatial pyramid [11].

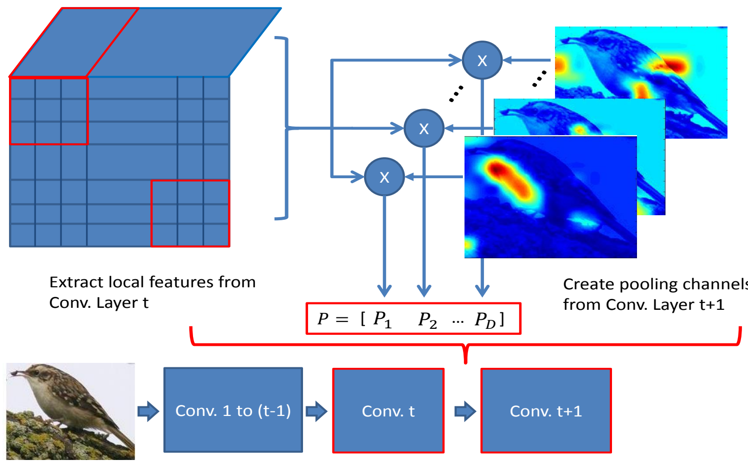
Extract local features from Conv. Layer t
Create pooling channels from Conv. Layer t+1
$P = [P_1 \; P_2 \; \cdots \; P_D]$
Conv. 1 to (t-1)
Conv. t
Conv. t+1

Figure 1: The overview of the proposed method.

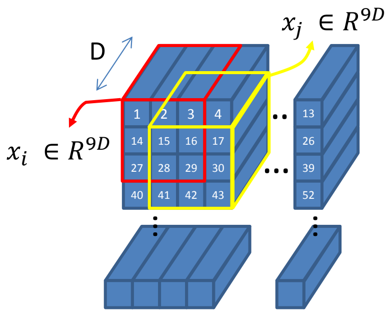

Figure 3: A depiction of the process of extracting local features from a convolutional layer.

3. Proposed Method

3.1. Convolutional layers vs. fully-connected layers

One major difference between convolutional and fully-connected layer activations is that the former is embedded with rich spatial information while the latter is not. The convolutional layer activations can be formulated as a tensor of the size $H \times W \times D$, where H, W denote the height and width of each feature map and D denotes the number of feature maps. Essentially, the convolutional layer divides the input image into $H \times W$ regions and uses D-dimensional feature maps (filter responses) to describe the visual pattern within each region. Thus, convolutional layer activations can be viewed as a 2-D array of D-dimensional *local features* with each one describing a local region. For the sake of clarity, we name each of the $H \times W$ regions as a **spatial unit**, and the D-dimensional feature maps corresponding to a spatial unit as the **feature vector in a spatial unit**. The fully-connected layer takes the convolutional layer activations as the network input and transforms them into a feature vector representing the whole image. Spatial information is lost through this process, meaning that the feature vector corresponding to a particular spatial area cannot be recovered from the activations of the subsequent fully-connected layer.

As mentioned in section 2, DCNNs can also be applied to image patches, to extract local features, as a means of compensating for the fact that they are not translation invariant. This approach has a significant disadvantage, however, in as much as the network will then be applied to patches, that have significantly different statistics to the whole images on which the network was trained. This is because, when applied as a regional feature transform, a DCNN is essentially used to describe local visual patterns which correspond to small parts of objects rather than the whole images used for training. Figure 2 shows some training images from the ImageNet dataset and a set of resized local regions. As can be seen, although they all have the same image size, their appearance and level of detail are quite different. Thus, blindly applying fully-connected layer activations as local features introduces a significant input image style mismatch which could potentially undermine the discriminative power of DCNN activations.

Our proposal for avoiding the aforementioned drawback is to extract multiple regional descriptors from *a single DCNN applied to a whole image*. We realize this idea by leveraging the spatial information within convolutional layers. More specifically, in convolutional layers, we can easily locate a subset of activations which correspond to a local region. These subsets of activations correspond to a set of subarrays of convolutional layer activations and we use them as local features. Figure 3 demonstrates the extraction of such local features. For example, we can first extract D-dimensional feature vectors from regions $1, 2, 3, 14, 15, 16, 27, 28, 29$ and concatenate them into a $9 \times D$-dimensional feature vector and then shift one unit

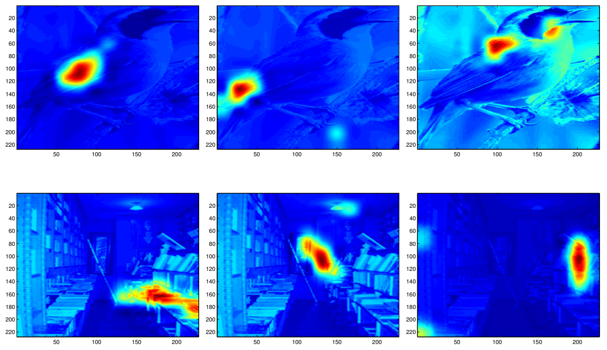

Figure 4: Visualizing of some feature maps extracted from the 5th layer of a DCNN.

along the horizontal direction to extract features from regions $2, 3, 4, 15, 16, 17, 28, 29, 30$. After scanning all the 13×13 feature maps we obtain 121 (omitting boundary spatial units) $(9 \times D)$-dimensional local features.

It is clear that in the proposed method the input of the DCNN is still a whole image rather than local regions. Thus the input image style mismatch issue is avoided. Note that in our method, we extract regional features from multiple spatial units and concatenate the corresponding feature vectors. This is as opposed to the approach in [12] (although developed for a different application) which treats the feature vector from one spatial unit as the local feature. We find that the use of feature vectors from multiple spatial units can significantly boost classification performance. This is because the feature vector from a single spatial unit may not be descriptive enough to characterize the visual pattern within a local region.

3.2. Cross-convolutional-layer Pooling

After extracting local features from a convolutional layer, one can directly perform traditional max-pooling or sum-pooling to obtain the image-level representation. In this section, we propose an alternative pooling method which can significantly improve classification performance. The proposed method is inspired by the parts-based pooling strategy [10] used in fine-grained image classification. In this strategy, multiple regions-of-interest (ROI) are first detected, with each corresponding to one human-specified object part, e.g. the tails of birds. Then local features falling into each ROI are then pooled together to obtain a pooled feature vector. Given D object parts, this strategy creates D different pooled feature vectors and these vectors are concatenated together to form the image representation. It has been shown that this simple strategy achieves significantly better performance than blindly pooling all local features together. Formally, the pooled feature from the kth ROI, denoted as $\mathbf{P}_k^t$, can be calculated by the following equation (let's consider sum-pooling in this case):

$$\mathbf{P}_k^t = \sum_{i=1} \mathbf{x}_i I_{i,k}, \tag{1}$$

where $\mathbf{x}_i$ denotes the ith local feature and $I_{i,k}$ is a binary *indicator map* indicating whether $\mathbf{x}_i$ falls into the kth ROI. We can also generalize $I_{i,k}$ to real values with its value indicating the 'membership' of a local feature to a ROI. Essentially, each indicator map defines a pooling channel and the image representation is the concatenation of pooling results from multiple channels.

However, in a general image classification task, there is no human-specified parts annotation, and even for many fine-grained image classification tasks the annotation and detection of these parts are usually non-trivial. To handle this situation, in this paper, we propose to use feature maps of the $(t+1)$th convolutional layer as D_{t+1} indicator maps. By doing so, D_{t+1} pooling channels are created for the local features extracted from the t-th convolutional layer. We call this method cross-convolutional-layer pooling or cross-layer pooling in short. The use of feature maps as indicator maps is motivated by the observation that a feature map of a deep convolutional layer is usually sparse and indicates some semantically meaningful regions[1]. This observation is illustrated in Figure 4. In Figure 4, we choose two images taken from two datasets, Birds-200 [13] and MIT-67 [14]. We randomly sample some feature maps from 256 feature maps in conv5 and overlay them on the original images for better visualization. As can be seen from Figure 4, the activated regions of the sampled feature map (highlighted in warm color) are actually semantically meaningful. For example, the activated region in top-left corner of Figure 4 corresponds to the wing-part of a bird. Thus, the filter of a convolutional layer works as a part detector and its feature map serves a similar role as the part region indicator map. Certainly, compared with the parts detector learned from human-specified part annotations, the filter of a convolutional layer is usually not directly task-relevant. However, the discriminative power of our image representation can benefit from combining a much larger number of indicator maps, e.g. 256 as opposed to 20-30 (the number of parts usually defined by human), which is akin to applying bagging to boost the performance of multiple weak classifiers.

Formally, the image representation extracted from cross-layer pooling can be expressed as follows:

$$\mathbf{P}^t = [\mathbf{P}_1^{t\top}, \mathbf{P}_2^{t\top}, \cdots, \mathbf{P}_k^{t\top}, \cdots, \mathbf{P}_{D_{t+1}}^{t\top}]^\top$$
$$\text{where,}\quad \mathbf{P}_k^t = \sum_{i=1}^{N_t} \mathbf{x}_i^t a_{i,k}^{t+1}, \tag{2}$$

[1]Note that similar observation has also been made in [7].

where $\mathbf{P}^t$ denotes the pooled feature for the t-th convolutional layer, which is calculated by concatenating the pooled feature of each pooling channel $\mathbf{P}_k^t, k = 1, \cdots, D_{t+1}$. $\mathbf{x}_i^t$ denotes the i-th local feature in the t-th convolutional layer. Note that feature maps of the $(t+1)$-th convolutional layer are obtained by convolving the feature maps of the t-th convolutional layer with a $m \times n$-sized kernel. So if we extract local features $\mathbf{x}_i^t$ from each $m \times n$ spatial unit in the t-th convolutional layer then each $\mathbf{x}_i^t$ naturally corresponds to a spatial unit in the $(t+1)$-th convolutional layer. Let us denote the feature vector in this spatial unit as $\mathbf{a}_i^{t+1} \in \mathbb{R}^{D_{t+1}}$ and the value at its k-th dimension as $a_{i,k}^{t+1}$. Then we use $a_{i,k}^{t+1}$ to weight local feature $\mathbf{x}_i^t$ in the k-th pooling channel.

Implementation Details: In our implementation, we perform PCA on $\mathbf{x}_i^t$ to reduce the dimensionality of $\mathbf{P}^t$. Also, we apply power normalization to $\mathbf{P}^t$, that is, we use $\hat{\mathbf{P}}^t = \text{sign}(\mathbf{P}^t)\sqrt{|\mathbf{P}^t|}$ as the image representation to further improve performance. We also tried directly using $\text{sign}(\mathbf{P}^t)$ as an image representation, that is, we coarsely quantize $\mathbf{P}^t$ into $\{-1, 1, 0\}$ according to the feature sign of $\mathbf{P}^t$. A similar strategy has been previously applied to convolutional features in [8] and it is reported to produce worse performance. However, to our surprise, our experiments show that this operation does not significantly decrease the performance of our cross-layer pooling representation. This observation allows us to simply use 2-bits to represent each feature dimension which significantly reduces the memory requirement for storing image representations. Please refer to section 4.3.3 for a more detailed discussion.

4. Experiments

We evaluate the proposed method on four datasets: MIT indoor scene-67 (MIT-67 in short) [14], Caltech-UCSD Birds-200-2011 [13] (Birds-200 in short), PASCAL VOC 2007 [15] (PASCAL-07 in short) and H3D Human Attributes dataset [16] (H3D in short). These four datasets cover several popular topics in image classification, that is, scene classification, fine-grained object classification, generic object classification and attribute classification. Previous studies [3, 4] have shown that using activations from the fully-connected layer of a pretrained DCNN leads to surprisingly good performance on those datasets. Here, in our experiments, we further compare different ways of extracting image representations from a pretrained DCNN. We organized our experiments into two parts, the first compares the proposed method against other competitive methods and the second examines the impact of various components of our method.

4.1. Experimental protocol

We compare the proposed method against three baselines, they are: (1) directly using fully-connected layer activations for the whole image (CNN-Global); (2) averaging fully-connected layer activations from several transformed versions of an input image. Following [3, 4], we transform the input image by cropping its four corners and middle regions as well as by creating their mirrored versions; (3) the method in [6]. It extracts fully-connected layer CNN activations from multiple regions in an image and encodes them using sparse coding based Fisher vector encoding (R-CNN SCFV). Since R-CNN SCFV has demonstrated superior performance to the MOP method in [5], we do not include MOP in our comparison. To make fair comparison, we reimplement all three baseline methods.

Table 1: Comparison of results on MIT-67. The lower part of this table lists some results reported in the literature. The proposed methods are marked with *.

Methods	Accuracy	Remark
CNN-Global	57.9%	-
CNN-Jitter	61.1%	-
R-CNN SCFV [6]	68.2%	-
*CL-45	64.6%	-
*CL-45F	65.8%	-
*CL-45C	68.8%	-
*CL + CNN-Global	70.0%	-
*CL + CNN-Jitter	**71.5%**	-
Fine-tuning [4]	66.0%	fine-tunning on MIT-67
MOP-CNN [5]	68.9%	three scales
VLAD level2 [5]	65.5%	single scale
CNN-SVM [3]	58.4%	-
FV+DMS [17]	63.2%	-
DPM [18]	37.6%	-

For all methods, we adopt the pretrained Alex net [1] provided in the caffe [19] package to extract CNN activations. We experiment with two resolutions for extracting convolutional features. The first sets the size of the 4th and 5th convolutional layers to be 13×13 spatial units, which is the default option in the Caffe implementation. We also tried a finer spatial resolution which uses 26×26 spatial units (we choose 26×13 spatial resolution for H3D because most images in H3D have greater height than width).

In the first part of our experiments, we report the results obtained using the 4th and 5th convolutional layer since this achieves the best performance. We denote our methods as CL-45, CL-45F, CL-45C, corresponding to the settings of applying our method to the default spatial resolution, to finer resolution and combining representations from two different resolutions, respectively. We also describe a similar experiment on the 3-4th layer of a DCNN in the second part of experiments and denote them as CL-34, CL-34F and CL-34C respectively. To reduce the dimensionality of the image representations we perform PCA on local features extracted from convolutional layers and reduce their dimensionality to 500 before cross-layer pooling. In practice, we find that reducing to higher dimensionality only slightly increases the performance. We use libsvm [20] as the SVM solver and use precomputed linear kernels as inputs. This is because the calculation of linear kernels/Gram matrices can be easily implemented in parallel. When feature dimensionality is high the kernel matrix computation actually occupies most of computational time. Thus it is appropriate to use parallel computing to accelerate this process.

4.2. Performance evaluation

4.2.1 Classification Result

Scene classification: MIT-67. MIT-67 is a commonly used benchmark for evaluating scene classification algorithms, it contains 6700 images with 67 indoor scene categories. Following the standard setting, we use 80 images in each category for training and 20 images for testing. The results are shown in Table 1. It can be seen that all the variations of our method (methods with '*' mark in Table 1) outperforms the methods that use DCNN activations as global features (CNN-Global and CNN-Jitter). This clearly demonstrates the advantage of using DCNN convolutional activations as local features. We can also see that the performance obtained by combining CL-45 and CL-45F, denoted as CL-45C, has already achieved the same performance as the regional-CNN based methods (R-CNN SCFV and MOP-CNN). Moreover, combining with the global-CNN representation, our method can obtain further performance gain. By combining CL-45C with CNN-Jitter, our method, denoted as CL+CNN-Global and CL+CNN-Jitter respectively, achieves impressive classification accuracy 71.5%.

Fine-grained image classification: Birds-200. Birds-200 is the most popular dataset in fine-grained image classification research. It contains 11788 images with 200 different bird species. This dataset provides ground-truth annotations of bounding boxes and parts of birds, e.g. the head and the tail, on both the training set and the test set. In this experiment, we just use the bounding box annotation. The results are shown in Table 2. As can be seen, the proposed method performs especially well on this dataset. Even CL-45 achieves 72.4% classification accuracy, a 6% improvement over the performance of R-CNN SCFV which, as far as we know, is the best performance obtained in the literature when no parts information is utilized. Combining with CL-45F, our performance can be improved to 73.5%. This is quite close to the best performance obtained from the method that relies on strong parts annotation. Another interesting observation is that for this dataset, CL-45 significantly outperforms CL-45F, which is in contrast to the case for MIT-67. This suggests that the optimal resolution of spatial units may vary from dataset to dataset.

Object classification: PASCAL-2007. PASCAL VOC 2007 has 9963 images with 20 object categories. The task is to predict the presence of each object in each image. Note that most object categories in PASCAL-2007 are also included in ImageNet. So ImageNet can be seen as a super-set of PASCAL-2007. The results on this dataset are shown in Table 3. From Table 3, we can see that the best performance of our method (CL + CNN-Jitter) achieves comparable performance to the state-of-the-art. Also, using only features extracted from a convolutional layer, our method CL-45C outperforms the CNN-Global and CNN-Jitter which use DCNNs to extract global image features. However, our CL-45C does not outperform R-CNN and our best performing method CL + CNN-Jitter does not achieve significant performance improvement as what it has achieved in MIT-67 and Birds-200. This is probably due to the fact that the 1000 categories in the ImageNet training set included the 20 categories in PASCAL-2007. Thus the fully-connected layer actually contains some classifier-level information and implicitly utilizes more training data from ImageNet. For this reason, using fully-connected layer activations can be more helpful for this dataset.

Attribute Classification: H3D. In recent years, attributes of objects, which are semantic or abstract qualities of objects and can be shared by many categories, have gained increasing attention due to their potential application in zero/one-shot learning and image retrieval [27, 28]. In this experiment, we evaluate the proposed method on the task of predicting attributes of humans. We use the H3D dataset [16] which defines 9 attributes for a subset of 'person' images from PASCAL VOC 2007. The results are shown in Table 4. Again, our method shows quite promising results. Merely using information from a convolutional layer, our approach achieved 77.3% classification accuracy which

Table 2: Comparison of results on Birds-200. Note that the method with "use parts" mark requires parts annotations and detection while our methods do not employ these annotations so they are not directly comparable with us.

Methods	Accuracy	Remark
CNN-Global	59.2%	no parts.
CNN-Jitter	60.5%	no parts
R-CNN SCFV [6]	66.4%	no parts
*CL-45	72.4%	no parts
*CL-45F	68.4%	no parts
*CL-45C	**73.5%**	no parts
*CL + CNN-Global	72.4%	no parts
*CL + CNN-Jitter	73%	no parts
GlobalCNN-FT [4]	66.4 %	no parts, fine tunning
Parts-RCNN-FT [21]	76.37 %	use parts, fine tunning
Parts-RCNN [21]	68.7 %	use parts, no fine tunning
CNNaug-SVM [3]	61.8%	-
CNN-SVM [3]	53.3%	CNN global
DPD+CNN [22]	65.0%	use parts
DPD [23]	51.0%	-

Table 3: Comparison of results on PASCAL VOC 2007.

Methods	mAP	Remark
CNN-Global	71.7%	-
CNN-Jitter	75.0%	-
R-CNN SCFV [6]	76.9%	-
*CL-45	72.6%	-
*CL-45F	71.3%	-
*CL-45C	75.0%	-
*CL + CNN-Global	76.5%	-
*CL + CNN-Jitter	**77.8%**	-
CNNaug-SVM [3]	77.2%	with augmented data
CNN-SVM [3]	73.9%	no augmented data
NUS [24]	70.5%	-
GHM [25]	64.7%	-
AGS [26]	71.1%	-

Table 4: Comparison of results on the Human attribute dataset.

Methods	mAP	Remark
CNN-Global	74.1%	-
CNN-Jitter	74.6%	-
R-CNN SCFV [6]	73.1%	-
*CL-45	75.3%	-
*CL-45F	70.7%	-
*CL-45C	77.3%	-
*CL + CNN-Global	78.1%	-
*CL + CNN-Jitter	**78.3%**	-
PANDA [29]	78.9	needs poselet annotation
CNN-FT [4]	73.8	CNN-Global, fine tunning
CNNaug-SVM [3]	73.0%	with augmented data
CNN-SVM [3]	70.8%	no augmented data
DPD [24]	69.9%	-

outperforms R-CNN SCFV by 4%. By combining with CNN-Jitter, our method becomes comparable to PANDA [29] which needs complicated poselet annotations and detection.

4.2.2 Computational cost

To give an intuitive idea of the computational cost incurred by our method, we report the average time spent on extracting image representations of various methods in Table 5. As can be seen, the computational cost of our method is comparable to that of CNN-Global and CNN-Jitter. This is quite impressive given that our method achieves significantly better performance than these two methods. SCFV, however, requires much more computational time[2]. Note that our speed evaluation is based on our naive MATLAB implementation, and our method may be further accelerated by a C++ or GPU implementation.

Table 5: Average time used for extracting an image representation for different methods. The time can be break down into two parts, time spend on extracting CNN features and time spend on performing pooling.

Method	CNN Extraction	Pooling	Total
*CL-45	0.45s	0.14s	0.6s
*CL-45F	1.3s	0.27s	1.6s
*CL-45C	1.75s	0.41s	2.2s
CNN-Global	0.4s	0s	0.4s
CNN-Jitter	1.8s	0s	1.8s
R-CNN SCFV [6]	19s	0.3s	19.3s

Table 6: Comparison of results obtained by using different convolutional layers.

Method	MIT-67	Birds200	PASCAL07	H3D
CL-34	61.7%	64.6%	66.3%	74.7%
CL-34F	61.4%	61.4%	64.9%	70.4%
CL-34C	64.1%	66.8%	68.5%	75.9%
CL-45C	**68.8%**	**73.5%**	**75.0%**	**77.3%**

4.3. Analysis of components of our method

From the above experiments, the advantage of using the proposed method has been clearly demonstrated. In this section we further examine the effect of various components in our method.

4.3.1 Using different convolutional layers

First, we are interested to examine the performance of using convolutional layers other than the 4th and 5th convolutional layers. We experiment with the 3th and 4th convolutional layers and report the resulting performance in Table 6. From the result we can see that using 4-5th layers achieves superior performance over using the 3-4th layers. This is not surprising since it has been observed that the deeper the convolutional layer, the better discriminative power[7].

4.3.2 Comparison of different pooling schemes

The cross-layer pooling is an essential component in our method. In this experiment, we compare it against other possible alternative pooling approaches, they are: directly

[2]A recent study shows that R-CNN based encoding methods such as [5] might be speeded up by treating the fully connected layer as a convolutional layer [30]. However, it still requires more 'CNN extraction' time than ours since beside the same convolutional feature extraction step as required by our method it also requires additional convolution operations to extract the fully connected layer activations.

Table 7: Comparison of results obtained by using different pooling schemes.

Method	MIT-67	Birds200	PASCAL07	H3D
Direct Max	42.6%	52.7%	48.0%	61.1%
Direct Sum	48.4%	49.0%	51.3%	66.4%
SPP [11]	56.3%	59.5%	67.3%	73.1%
SCFV [6]	61.9%	64.7%	69.0%	**76.5%**
CL-single	**65.8%**	**72.4%**	**72.6%**	75.3%

performing sum-pooling with power normalization (Direct Sum) and max-pooling (Direct Max), using spatial pyramid pooling as suggested in [11] (SPP), applying the SCFV encoding [6] to encode extracted local features and perform pooling (SCFV). To simplify the comparison, we only report results on the best performing single resolution setting for each dataset, that is, CL-45F for MIT-67 and CL-45 for the remaining three datasets. We perform those pooling methods on local features (from 3×3 spaital units) extracted from the 4th convolutional layer since the proposed method can be seen as a way to pool local features from this layer. The results are shown in Table 7. As can be seen, the proposed cross-layer pooling significantly outperforms directly applying max-pooling or sum-pooling or even spatial-pyramid pooling. By applying another layer of encoding on local features before pooling, the classification accuracy can be greatly boosted. However, in most cases, its performance is still much inferior to the proposed method, as seen in cases of MIT-67, PASCAL-07 and Birds-200. The only exception is the result on H3D, where SCFV performs slightly better than our method. However, it needs additional codebook learning and encoding computation while our method does not. Considering this computational benefit and superior performance in most cases, cross-layer pooling is clearly preferable to the other methods.

4.3.3 Feature sign quantization

Finally, we demonstrate the effect of applying a feature sign quantization to the pooled feature. Feature sign quantization quantizes a feature to 1 if it is positive, -1 if it is negative and 0 if it equals to 0. In other words, we use 2 bits to represent each dimension of the pooled feature vector. This scheme greatly saves the memory usage. Similar to the above experiment setting, we only report the result on the best performed single resolution setting for each dataset. The results are shown in Table 8. Surprisingly, this coarse quantization scheme does not degrade the performance too much, in three datasets, MIT-67, PASCAL-07 and H3D, it achieves almost the same performance as the original feature. Note that similar quantization scheme has been also explored in [8], however it reports signficant performance drop if applied to convolutional layer feature. For example, in the Table 7 of [8], by binarizing conv-5, the performance drops around 5%. In contrast, our representation seems to be less sensitive to this coarse quantization.

Table 8: Results obtained by using feature sign quantization.

Dataset	Feature sign quantization	Original
MIT-67	65.2%	65.8%
Birds-200	71.1%	72.4%
PASCAL07	71.2%	71.3%
H3D	75.4%	75.3%

5. Conclusion

In this paper, we proposed a new method called cross-convolutional layer pooling to create image representations from the convolutional activations of a pretrained CNN. Through extensive experiments we have shown that this method enjoys good classification performance and at low computational cost. Our discovery suggests that if used appropriately, convolutional layers of a pretrained CNN contains very useful information and can be turned into a powerful image representation.

In our future work, we will further explore this idea by training a convolutional neural network with the cross-layer pooling module as one of its layers.

References

[1] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1106–1114.

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2009.

[3] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: an astounding baseline for recognition," *Proc. Workshop of IEEE Conf. Comp. Vis. Patt. Recogn.*, 2014.

[4] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson, "From generic to specific deep representations for visual recognition," http://arxiv.org/abs/1406.5774, 2014.

[5] Y. Gong, L. Wang, R. Guo, and S. Lazebnik, "Multi-scale orderless pooling of deep convolutional activation features," *Proc. Eur. Conf. Comp. Vis.*, 2014.

[6] L. Liu, C. Shen, L. Wang, A. van den Hengel, and C. Wang, "Encoding high dimensional local features by sparse coding based fisher vectors," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014.

[7] M. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comp. Vis.*, 2014.

[8] P. Agrawal, R. Girshick, and J. Malik, "Analyzing the performance of multilayer neural networks for object recognition," in *Proc. Eur. Conf. Comp. Vis.*, 2014.

[9] Y. Li, L. Liu, C. Shen, and A. van den Hengel, "Midlevel deep pattern mining," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2015.

[10] T. D. Ning Zhang, Ryan Farrell, "Pose pooling kernels for sub-category recognition," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2012.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *Proc. Eur. Conf. Comp. Vis.*, 2014.

[12] W. Y. Zou, X. Wang, M. Sun, and Y. Lin, "Generic object detection with dense neural patterns and regionlets," in *Proc. British Machine Vis. Conf.*, 2014.

[13] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona, "Caltech-UCSD birds 200," Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

[14] A. Quattoni and A. Torralba, "Recognizing indoor scenes," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2009, pp. 413–420.

[15] M. Everingham, L. Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *Int. J. Comput. Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[16] L. Bourdev, S. Maji, and J. Malik, "Describing people: Poselet-based attribute classification," in *Proc. IEEE Int. Conf. Comp. Vis.*, 2011.

[17] C. Doersch, A. Gupta, and A. A. Efros, "Mid-level visual element discovery as discriminative mode seeking," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013.

[18] M. Pandey and S. Lazebnik, "Scene recognition and weakly supervised object localization with deformable part-based models," in *Proc. IEEE Int. Conf. Comp. Vis.*, 2011, pp. 1307–1314.

[19] Y. Jia, "Caffe," 2014, http://mloss.org/software/view/539/.

[20] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM T. Intelligent Systems & Technology*, vol. 2, pp. 27:1–27:27, 2011.

[21] N. Zhang, J. Donahue, R. Girshick, and T. Darrell, "Part-based R-CNNs for fine-grained category detection," in *Proc. Eur. Conf. Comp. Vis.*, 2014.

[22] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "DeCAF: A deep convolutional activation feature for generic visual recognition," in *Proc. Int. Conf. Mach. Learn.*, 2014.

[23] N. Zhang, R. Farrell, F. Iandola, and T. Darrell, "Deformable part descriptors for fine-grained recognition and attribute prediction," in *Proc. IEEE Int. Conf. Comp. Vis.*, December 2013.

[24] Z. Song, Q. Chen, Z. Huang, Y. Hua, and S. Yan, "Contextualizing object detection and classification.," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2011.

[25] Q. Chen, Z. Song, Y. Hua, Z. Huang, and S. Yan, "Hierarchical matching with side information for image classification.," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2012, pp. 3426–3433.

[26] J. Dong, W. Xia, Q. Chen, J. Feng, Z. Huang, and S. Yan, "Subcategory-aware object classification," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2013, pp. 827–834.

[27] D. Parikh and K. Grauman, "Relative attributes," in *Proc. IEEE Int. Conf. Comp. Vis.*, 2011.

[28] A. Kovashka, D. Parikh, and K. Grauman, "Whittlesearch: Image search with relative attribute feedback," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2012.

[29] N. Zhang, M. Paluri, M. Ranzato, T. Darrell, and L. Bourdev, "PANDA: Pose aligned networks for deep attribute modeling," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, 2014.

[30] D. Yoo, S. Park, J. Lee, and I. S. Kweon, "Fisher kernel for deep neural activations," http://arxiv.org/abs/1412.1628, 2014.