Cross-Convolutional-Layer Pooling for Image Recognition

Lingqiao Liu, Chunhua Shen, and Anton van den Hengel

Abstract—Recent studies have shown that a Deep Convolutional Neural Network (DCNN) trained on a large image dataset can be used as a universal image descriptor and that doing so leads to impressive performance for a variety of image recognition tasks. Most of these studies adopt activations from a single DCNN layer, usually a fully-connected layer, as the image representation. In this paper, we proposed a novel way to extract image representations from two consecutive convolutional layers: one layer is used for local feature extraction and the other serves as guidance to pool the extracted features. By taking different viewpoints of convolutional layers, we further develop two schemes to realize this idea. The first directly uses convolutional layers from a DCNN. The second applies the pre-trained CNN on densely sampled image regions and treats the fully-connected activations of each image region as a convolutional layer's feature activations. We then train another convolutional layer on top of that as the pooling-guidance convolutional layer. By applying our method to three popular visual classification tasks, we find that our first scheme tends to perform better on applications which demand strong discrimination on lower-level visual patterns while the latter excels in cases that require discrimination on category-level patterns. Overall, the proposed method achieves superior performance over existing approaches for extracting image representations from a DCNN. In addition, we apply cross-layer pooling to the problem of image retrieval and propose schemes to reduce the computational cost. Experimental results suggest that the proposed method achieves promising results for the image retrieval task.

Index Terms—Convolutional networks, deep learning, pooling, fine-grained object recognition

1 Introduction

Recently, Deep Convolutional Neural Networks (DCNNs) have attracted much research attention in visual recognition, largely due to their excellent performance [1]. It has been discovered that the activation of a DCNN trained on a large dataset, such as ImageNet [2], can be employed as a universal image descriptor, and applying this descriptor to many visual classification and retrieval problems delivers impressive performance [3], [4], [5]. This discovery quickly sparked significant interest and inspired many extensions, including [6], [7]. A fundamental issue with these kinds of methods is how to generate an image representation from a pre-trained DCNN. Most current solutions take activations of a single DCNN layer, usually the fully-connected layer, as the image representation.

In this paper, we show that we can build a powerful image representation using the activations from two consecutive convolutional layers. We name our method cross-convolutional layer pooling (or cross-layer pooling for short). This new method relies on two crucial components: (1) we extract local features from one convolutional layer (2) we pool extracted local features by using activations from its successive convolutional layer as guidance.

The first component is motivated by recent work [6], [7], [8] 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. In this paper, we view those regional CNN activations as a newly added convolutional layer (named as the augmented convolutional layer as discussed in Section 3.1). Inspired by this view, we also extract local features from the original convolutional layers of the pre-trained CNN.

The second component is motivated by the parts-based pooling method [9] which was originally proposed for fine-grained image classification. This method creates one pooling channel for each detected part region while the final image representation is obtained by concatenating pooling results from multiple channels. We generalize this idea to DCNNs and avoid the need for annotating predefined parts. More specifically, we deem the feature map of each filter in a convolutional layer as the detection score map of a part detector and apply the feature map to weight regional descriptors extracted from the 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 in contrast to existing regional-DCNN based methods [6], [7], the proposed method does not require additional dictionary learning and encoding steps at either the training or testing stage once the convolutional layer activations become available. To further reduce the memory use 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.

Besides image classification, we explore the use of cross-layer pooling for image retrieval. To overcome the high computational cost of the direct implementation of cross-layer pooling, we propose to employ feature binarization and adaptive pooling channel selection schemes to reduce the computational cost.

We conduct extensive experiments on three popular visual classification datasets, and three popular image retrieval datasets. Experimental results suggest that the proposed method achieves significantly better performance than competitive methods in most cases. Further ablation studies provide insight into the importance of various components of our approach.

A preliminary version of this paper has been published in [10]. In this paper, we have made a significant extension. The major differences are threefold.

1) We view the scheme of extracting fully connected CNN activations at densely sampled regions as a newly added convolutional layer and perform cross-layer pooling at that level. This extension makes our method more widely applicable.
2) We apply cross-layer pooling to image retrieval tasks and propose new schemes to reduce the computational cost.
3) We have conducted more experiments to validate the proposed method, including experiments with a better CNN model and new ablation studies.

Preliminary. Two network structures are considered in this paper. One is the AlexNet [1] and another one is the VGG very deep (VGGVD in short) network [11]. Both networks are composed by the cascade of convolutional layers and fully connected layers. 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 Related Work

In the literature, there are two primary methods for using a pre-trained DCNN to create an image representation: (1) directly feeding the whole image into a pre-trained DCNN and extracting its

• *The authors are with the School of Computer Science, The University of Adelaide, Adelaide, SA 5005, Australia.*
E-mail: {lingqiao.liu, chunhua.shen, anton.vandenhengel}@adelaide.edu.au.

Manuscript received 8 Oct. 2015; revised 9 Oct. 2016; accepted 22 Oct. 2016. Date of publication 8 Dec. 2016; date of current version 10 Oct. 2017.
Recommended for acceptance by S. Lazebnik.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPAMI.2016.2637921

0162-8828 © 2016 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

activations; (2) applying the pre-trained DCNN to subregions of the input image and aggregating activations from multiple regions as the image representation. Usually, the first method extracts the last few fully-connected layer activations as the image-level representation. Fine-tuning is sometimes applied to make the network better adapted to a given task. 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].

DCNNs can also be applied to extract local features. It has been suggested that DCNN activations are not invariant to a large amount of translation [6] and that 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 [6]. In [6], another layer of unsupervised encoding is employed to create the image-level representation [6], [7]. In [12], discriminative patterns are mined from those regional activations for classification. It is shown that for many visual tasks [6], [7] this approach leads to better performance than directly extracting DCNN activations as global features.

One common factor in the above methods is that they all use fully-connected layer activations as features. Some recent studies explore the use of convolutional layers to extract image representations. For example, the work in [13] applies Fisher vector pooling to the local features extracted from a convolutional layer to create image representations for texture classification. The work in [14] uses pooled convolutional activations for object detection. The authors in [5] demonstrated that the pooled convolutional feature are well suited to the image retrieval task. The work in [15] is probably most relevant to our work. As mentioned in [15] itself, their approach is an extension of the method proposed in [10] (the preliminary version of this paper) for fine-grained image classification. It uses a similar strategy as ours to combine the convolutional feature activations from two DCNNs and jointly fine-tune all of the parameters in an end-to-end fashion.

3 Proposed Method

3.1 Convolutional Layers, Fully Connected Layers and Notations

The convolutional layer in a CNN is embedded with rich spatial information. Its activations can be formulated as a tensor of the size $H \times W \times D$, where H and W denote the height and width of each feature map and D denotes the number of feature maps. These activations can alternatively be viewed as an array of D-dimensional local features extracted at $H \times W$ locations. In this paper, we denote each of the $H \times W$ locations as a *spatial unit*, and the D-dimensional feature maps corresponding to a location as the *feature vector for a spatial unit*.

The fully-connected layer can be seen as a convolutional layer with the receptive field as the whole image. In recent literature [6], [8], [16], the activations from a fully connected layer are often used as a descriptor for image regions rather than the whole image. As pointed out in [17], if the regions are sampled from the input image over a dense grid, such a descriptor extraction process can also be viewed as applying a convolutional layer. For the sake of clarity, in this paper we refer such a convolutional layer as the "augmented convolutional layer" or the AConv layer for short and the convolutional layer of the original pre-trained CNN as the OConv layer.

3.2 Cross-Convolutional-Layer Pooling

In [6], [7], it has been shown that applying an additional pooling operation on the local features extracted from multiple image regions can significantly boost classification performance. Motivated by these methods, we can design a specific pooling layer and apply it to the local features extracted from a convolutional layer which can be either the OConv layer or the AConv layer.

Various pooling methods could be chosen to aggregate the local features, e.g., max-pooling, sum-pooling or Fisher vector based pooling [18]. In this section, we propose an alternative pooling method which significantly improves classification performance.

The proposed method is inspired by the parts-based pooling strategy [9] 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 directly pooling all local features together. Formally, the pooled feature from the kth ROI of the tth layer, which we denote by $\mathbf{P}_k^t$, 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 an 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 are no human-specified part annotations, and even for many fine-grained image classification tasks the annotation and detection of these parts are typically 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 tth convolutional layer. We call this method cross-convolutional-layer pooling or cross-layer pooling for 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 tends to be selective of higher-level visual concepts, as has also been observed in [19]. This observation is illustrated in Fig. 3. In Fig. 3, we choose three images taken from the Birds-200 [20] dataset. We sample three feature maps from 256 feature maps in conv5 and overlay them on the original images for better visualization. As can be seen from Fig. 3, the activated regions of the sampled feature map (highlighted in warm colors) are semantically meaningful. For example, the activated region in the first row tends to localize at the head region of a bird while the feature map shown in the second row exhibits high values around the claw area. 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).

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

$$\begin{aligned} &\mathbf{P}^t = [\mathbf{P}_1^{t\top}, \mathbf{P}_2^{t\top}, \ldots, \mathbf{P}_k^{t\top}, \ldots, \mathbf{P}^{t\top}_{D_{t+1}}]^\top \\ &\text{where,} \quad \mathbf{P}_k^t = \sum_{i=1}^{N_t} \mathbf{x}_i^t x_{i,k}^{t+1}, \end{aligned} \tag{2}$$

where $\mathbf{x}_i^t \in \mathbb{R}^{D_t}$ is the ith local feature (filter responses) in the tth convolutional layer. N^t is the total number of local features at the tth layer. $x_{i,k}^{t+1} \in R$ is the activation value of the ith spatial unit and

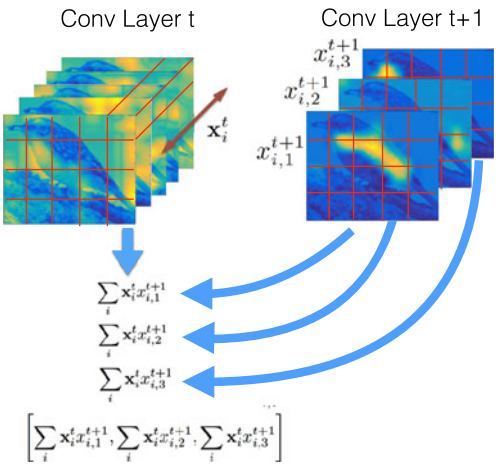

Fig. 1. Illustration of the proposed method. Our method performs the pooling operation by using two consecutive convolutional layers. Local features $\mathbf{x}_i^t$ are extracted from the tth convolutional layer, and each of the feature maps at the $(t+1)$th convolutional layer is used to perform weighted sum-pooling for $\mathbf{x}_i^t$. The concatenation of all pooling results is used as the image representation.

the kth filter at the $(t+1)$th convolutional layer. Thus $\mathbf{P}_k^t$ represents a weighted sum-pooling of $\mathbf{x}_i^t$ with the weight defined by $x_{i,k}^{t+1}$. In total, there are D^{t+1} sets of weights since there are D^{t+1} filters at the $(t+1)$th convolutional layer. The final image representation $\mathbf{P}^t$ will be the concatenation of all $\mathbf{P}_k^t, k = 1, \ldots, D_{t+1}$ and thus its dimensionality will be $D^{t+1}D^t$. A demonstration of the proposed cross-layer pooling scheme is shown in Fig. 1 Note that here we assume that there is a correspondence between the ith local feature at the tth layer $\mathbf{x}_i^t$ and the ith feature activations at the $(t+1)$th layer. This correspondence can be easily established if the consecutive convolutional layers have the same spatial unit layout. For example, the last two convolutional layers in the Alex net both have a 13×13 spatial unit layout and we can deem that feature maps at the same spatial unit across two layers are corresponding.

Another way of interpreting Equ. (2) is that the image representation is a sum over outer products of corresponding features in two layers. This operation is similar to calculating Gram matrices which have been applied in computer vision [21], [22], [23]. The difference is that our cross-layer pooling calculates the outer product across different layers and thus can be seen as an extension of the Gram matrices based representation.

3.3 Augmented Convolutional Layers versus Original Convolutional Layers

To implement cross-layer pooling, one needs to specify two convolutional layers. In practice, these two convolutional layers can either be chosen from the AConv layer or the OConv layer.[1] But which type of convolutional layers performs better? We show empirically below that the best performing layer type depends on the recognition task to which it will be applied.

For the AConv layer, its convolutional filters are the fully-connected layer parameters of the original CNN. So the AConv layer encodes the higher-level visual concept, e.g., object-category-level information. Thus, if the target problem involves identification of object-category-level patterns, e.g., to classify whether a "car" occurs in the image, then the AConv layer should be used, and its activations can be seen as being similar to object bank detectors [24]. Note that even if the problem does not directly involve the identification of an object that appears in the network training task, e.g., the 1,000 categories in the ImageNet subset, category-level pattern detection may be still beneficial. For example, for scene classification, the occurrence of an object, such as a bed, can be a strong indicator of the scene class "bedroom".

1. It is also possible to choose one OConv layer and one AConv layer to perform cross-layer pooling. We discuss this possibility in Section 4.2.1.

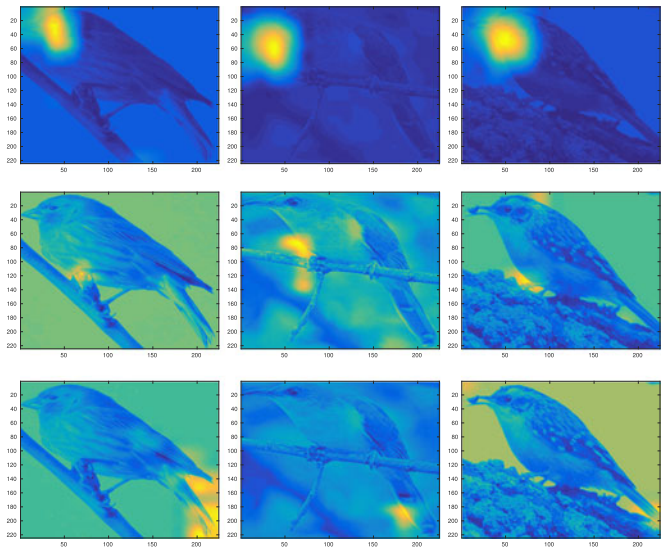

Fig. 2. This figure demonstrates the image style mismatch issue when using fully-connected layer activations as regional descriptors. Top row: input images that a DCNN 'sees' at the training stage. Bottom row: input images that a DCNN 'sees' at the test stage.

Compared with the AConv layer, the OConv layer captures lower-level visual patterns. Thus for target applications which require strong discriminative power to identify lower-level visual patterns, e.g., specific textures in a fine-grained image classification problem, using the OConv layer can transfer across domain more easily and lead to better performance than using the AConv layer. It should be noted that most commonly used pre-trained CNNs are trained on image classification datasets such as ImageNet [2]. Thus, if we use an AConv layer to describe low-level patterns, the input images of those pre-trained networks will be very different to that in the target domain. Fig. 2 shows an example in this case. The top row of Fig. 2 shows some images from the ImageNet [2] dataset while the bottom row shows some regions corresponding to the low level visual patterns from the images in Birds-200 [20] dataset. As can be seen, the appearance and the level of detail are quite different between the two rows. Thus, applying the AConv layer to describe lower-level visual patterns will introduce a significant input image style mismatch which could potentially undermine the discriminative power of DCNN activations.

3.4 Implementation Details

PCA. In our implementation, we perform PCA on $\mathbf{x}_i^t$ to reduce the dimensionality of $\mathbf{P}^t$ to 2,000 dimensions for the AConv layer. For the OConv layer, we still perform PCA to de-correlate the local features but without performing dimensionality reduction. We

Fig. 3. Visualization of feature maps extracted from the conv5-4 layer of the VGG Net. Three feature maps and their activations on three different images are shown. Each row represents the feature map corresponding to the same filter. Warmer color indicates higher activation values.

empirically find that this leads to slightly better performance than using the uncorrected OConv local features.

Normalization. Since the number of activated spatial units at the guidance convolutional layer can be different for different pooling channels. The pooling vector derived from different channels may have a different energy. Thus, in our implementation we ℓ_2 normalize the pooled coding vector for each channel. After that, 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.

Feature Sign Quantization. Besides the aforementioned image representation, 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 $\mathbf{P}^t$.

Adding a New Convolutional Layer for the AConv Layer. One issue when using the AConv layer for cross-layer pooling is that we need to find two consecutive AConv layers. These two layers can be obtained by using two consecutive fully-connected layers in the original DCNN. However, since the fully-connected layers in most commonly used DCNN models have a very large number of output neurons, e.g., 4,096 or 1,000. Directly performing cross-layer pooling on those AConv layers will result in a very high dimensional image representation. To solve this issue, we only utilize one fully connected layer from the original DCNN as one AConv layer, and stack another newly added convolutional layer on top of it with a much smaller number of filters, e.g., 100. Then we train the new convolutional layer on the target dataset. The network architecture of our implementation is as follows: a max-pooling layer is applied to pool the activations of the newly added convolutional layer and the pooled result is feed into a logistic regression layer. The negative entropy loss is then utilized to train the new convolutional layer.

3.5 Application to Image Retrieval

Recently, it has been discovered that the pooled convolutional layer activations can form a good image representation for image retrieval [5]. Inspired by the success of the work in [5], we apply our cross-layer pooling method to image retrieval. Since cross-layer pooling creates multiple pooling channels with each pooling channel capturing one type of visual pattern, and the pooling result of each channel is normalized, an image representation created by cross-layer pooling can depict various aspects of visual patterns within the image in a balanced way. In comparison, the representation generated by the direct pooling method in [5] may be dominated by patterns that occur more frequently within the image.

Directly applying cross-layer pooling for image retrieval will incur a high computational cost due to the high dimensionality of the generated image representations. For example, if there are M pooling channels, the computational cost can be M times higher than the method in [5]. To handle this drawback, in this paper we propose two strategies. The first strategy is to binarize the cross-layer pooling feature. This is inspired by the observation (which will be experimentally demonstrated in Section 4.4) that keeping the sign of feature values[2] does not significantly impact the discriminative power of cross-layer pooling. The second strategy is to adaptively select a small number, say k, of pooling channels for *each query image* and then only retain the features pooled from the selected channels in both the query and the reference image to perform the similarity comparison. Formally, for such a scheme the image similarity between a query image and a reference image is calculated via

$$s(\mathbf{I}_q, \mathbf{I}) = \sum_{k \in \mathcal{S}} \mathbf{P}_{q,k}^T \mathbf{P}_{x,k}, \tag{3}$$

where $\mathbf{I}_q$ and $\mathbf{I}$ are the query image and a reference image. In the original cross-layer pooling, both the query image and the reference image are represented by D subvectors which are pooled from each pooling channel. We use $\mathbf{P}_{q,k}$ and $\mathbf{P}_{x,k}$ denote the kth subvectors of the query and a reference image respectively. For the original cross-layer pooling approach, the comparison should be made over all D subvectors. Here in Equ. (3), only subvectors whose indices fall within a small subset $\mathcal{S}$ ($|\mathcal{S}| = k \ll D$) are compared. Thus, the computational cost can be greatly reduced in comparison with the naive implementation of cross-layer pooling for image retrieval. In this paper, we construct the set $\mathcal{S}$ by selecting channels (feature maps of a convolutional layer) with top k average activations. By applying this criterion, the convolutional feature maps with less significant activations will be discarded. Thus, for this operation, one additional benefit besides a reduction in the computational cost is that it might suppress the noise patterns and therefore potentially improve retrieval performance. Note that besides selecting channels with top k activation values, other criteria can be applied. For example, if the retrieval task is to find a specific object type such as sculpture, cloth, an importance weight for each pooling channel can be learned by using images which contain the object-of-interest as positive training samples and random images as negative training samples.

4 Experiments

We have organized our experiments into three parts. The first evaluates the proposed cross-layer pooling method for the image classification application. In the second part, we conduct ablation studies and demonstrate the impact of the various components of our method. In the third part, we evaluate the proposed method on image retrieval tasks. The focus of the third part is to evaluate whether the proposed cross-layer pooling leads to better performance than the method in [5] which also uses a convolutional layer pooling strategy for image retrieval.

4.1 Image Classification Experiments

4.1.1 Experimental Protocol

We evaluate the proposed method on three datasets: MIT indoor scene-67 (MIT-67 in short) [28], Caltech-UCSD Birds-200-2011 [20] (Birds-200 in short) and PASCAL VOC 2007 [29] (PASCAL-07 in short) for image classification. These three datasets cover several popular topics in image classification, that is, scene classification, fine-grained object classification and generic object classification.

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 (CNN-Jitter); (3) applying the sparse coding based Fisher vector encoding method [7] on the local feature extracted from the convolutional layer (in both schemes) (SCFV). Since R-CNN SCFV has demonstrated superior performance to the MOP method in [6], we do not include MOP in our comparison. To make a fair comparison, we reimplement all three baseline methods. We also apply PCA, ℓ_2 normalization and power normalization to SCFV and ℓ_2 normalization to CNN-Global and CNN-Jitter (We find that using PCA and/or power normalization makes little difference to the result of CNN-Global and CNN-Jitter).

Two CNN models are adopted throughout our experiment: the Alex net [1] and the VGG very-deep 19 layer network [11]. Two different types of convolutional layers are used, the original convolutional layer (denoted as OConv) and the augmented convolutional layer (denoted as AConv). As discussed in Section 3.1, the latter can be implemented by applying DCNN on a set of image regions which are extracted on a dense grid. In our implementation, we first resize

2. As will be discussed in the second strategy, some channels will be discarded during feature similarity calculation, which is equivalent to setting the feature values within the discarded channels to 0. In this view, there are still three possible values, 1,−1,0, in the resulting image representation.

TABLE 1
Comparison of Results on MIT-67

Methods	Accuracy	Remark/Setting
CNN-Global	57.9%	AlexNet
CNN-Jitter	61.1%	AlexNet
SCFV [7]	59.2%	AlexNet, OConv
CrossLayer (proposed)	63.0%	AlexNet, OConv
SCFV [7]	**68.2%**	AlexNet, AConv
CrossLayer (proposed)	**68.2%**	AlexNet, AConv
CNN-Global	68.2%	VGGNet
CNN-Jitter	70.2%	VGGNet
SCFV [7]	73.5%	VGGNet, OConv
CrossLayer (proposed)	**74.4%**	VGGNet, OConv
SCFV [7]	76.4%	VGGNet, AConv
CrossLayer (proposed)	**78.2%**	VGGNet, AConv
Fine-tuning [25]	69.8%	fine-tunning with the VGGNet
Fine-tuning [4]	66.0%	fine-tunning with the AlexNet
MOP-CNN [6]	68.9%	three scales
VLAD level2 [6]	65.5%	single scale
CNN-SVM [3]	58.4%	-
FV+DMS [26]	63.2%	-
DPM [27]	37.6%	-
DeepTexture [13]	**81.7%**	7 scales
Texture Synthesis [21]	75.0%	using the Gram matrix on fc18 layer (VGG net) for classification

The lower part of this table lists some results reported in the literature.

TABLE 2
Comparison of Results on Birds-200

Methods	Accuracy	Remark
CNN-Global	59.2%	no parts. AlexNet
CNN-Jitter	60.5%	no parts. AlexNet
SCFV [7]	64.2%	no parts, AlexNet, OConv
CrossLayer	**73.3%**	no parts, AlexNet, OConv
SCFV [7]	66.4%	no parts, AlexNet, AConv
CrossLayer	71.7%	no parts, AlexNet, AConv
CNN-Global	62.5%	no parts. VGGNet
CNN-Jitter	63.6%	no parts. VGGNet
SCFV [7]	73.7%	no parts, VGGNet, OConv
CrossLayer	**77.0%**	no parts, VGGNet, OConv
SCFV [7]	66.2%	no parts, VGGNet, AConv
CrossLayer	69.4%	no parts, VGGNet, AConv
Fine-tuning [15]	76.4%	no parts, fine tunning, VGGNet
Fine-tuning [4]	66.4 %	no parts, fine tunning, AlexNet
Parts-RCNN-FT [31]	76.37%	use parts, fine tunning
Parts-RCNN [31]	68.7%	use parts, no fine tunning
CNNaug-SVM [3]	61.8%	-
CNN-SVM [3]	53.3%	CNN global
DPD+CNN [32]	65.0%	use parts
DPD [33]	51.0%	-
Bilinear CNN [15]	77.9%	Two networks
Bilinear CNN [15]	**81.9%**	Two networks, fine-tuning
Texture Synthesis [21]	67.3	using the Gram matrix on conv5-4 layer (VGG net) for classification

Note that the method annotated with "use parts" requires parts annotations and detection while our methods do not employ these annotations so they are not directly comparable with our method. Also, the fine-tuning result in [15] is achieved with a different configuration while our method achieves 80 percent with the same configuration.

the input image to 512×512 pixels and then extract image regions with the size 128×128 at a step size of 32 pixels. For OConv layers, we report the results obtained using the 4th and 5th convolutional layers for the Alex net and the conv5-3 and conv5-4 convolutional layers for the VGGVD net since those settings achieve the best performance. We also explore the use of other convolutional layers in the second part of our experiment. For the AConv layer, we extract local features from the fc6 layer in the Alex net and the first fully-connected layer in the VGGVD net respectively. Then we stack a new convolutional layer with 100 filters on top of them and train the newly added layer on the target dataset. This new layer is trained with the learning rate 0.01 by using 50 epochs. No data augmentation is used in this training step.

We use libsvm [30] 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 the computational time. Thus it is appropriate to use parallel computing to accelerate this process.

4.1.2 Classification Results

Scene Classification: MIT-67.MIT-67 is a commonly used benchmark for evaluating scene classification algorithms. It contains 6,700 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 the proposed cross-layer pooling achieves the overall best performance in most settings. The best performance is achieved by using cross-layer pooling and the AConv layer: this setting produces 68.2 percent classification accuracy for the Alex net and 78.2 percent for the VGGVD net. Also, it is clear that extracting local features from the AConv layer, as has been done in SCFV and CrossLayer, achieves significant performance increase in comparison with global CNN features, i.e., Global and Jitter. Finally, the use of the VGGVD net further boosts the classification performance by a large margin.

By comparing the performance reported from the literature, we can see that the proposed method surpasses most state-of-the-art methods. The only exception is the result in [13]. However, its method is very close to our SCFV (with OConv) baseline, and its good performance is largely due to their brute force multiple-scale strategy (they have utilized seven scales while we only use a single scale).

Fine-Grained Image Classification: Birds-200.Birds-200 is the most popular dataset in fine-grained image classification research. It contains 11,788 images with 200 different bird species. This dataset provides ground-truth annotations of bounding boxes and parts of birds such as the head and the tail, on both the training set and the test set. In this experiment, we only use the bounding box annotation. The results are shown in Table 2. As can be seen, the cross-layer pooling achieves the best classification performance: 77.0 percent when the VGGVD net is used. Also, using the original convolutional layer achieves much better performance than the use of the AConv layers. For both the Alex net and the VGGVD net, the best performance is achieved by using features from the OConv layer. The underlying reason can be well explained by Section 3.3, that is, the discriminative information of birds species usually lies at small regions and it will be more appropriate to extract features from original convolutional layers due to the image style mismatch issue discussed in Section 3.3.

Our best performance is among the best for the dataset. The work in [15] reports higher classification accuracy than us. However, it relies on a fine-tuning step on two different networks and it adopts some different experimental settings, e.g., their convolutional layers have a different number of spatial units to ours[3] (28 × 28 as oppose to 14 × 14 in our experiments); it performs decision score calibration on the SVM while we just use the standard one-versus-the-rest SVM.

Object Classification: PASCAL-2007.PASCAL VOC 2007 contains 9,963 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 which is the

3. When the same spatial units configuration is used, our cross-layer pooling achieves 80 percent classification accuracy which is closer to the result in [15]. Note that we only use one network and do not apply fine-tuning and score calibration.

TABLE 3
Comparison of Results on PASCAL VOC 2007

Methods	mAP	Remark
CNN-Global	71.7%	AlexNet
CNN-Jitter	75.0%	AlexNet
SCFV [7]	66.8%	AlexNet, OConv
CrossLayer	71.3%	AlexNet, OConv
SCFV [7]	76.9%	AlexNet, AConv
CrossLayer	**79.1%**	AlexNet, AConv
CNN-Global	83.6%	VGGNet
CNN-Jitter	84.5%	VGGNet
SCFV [7]	82.9%	VGGNet, OConv
CrossLayer	84.1%	VGGNet, OConv
SCFV [7]	85.1%	VGGNet, AConv
CrossLayer	**87.1%**	VGGNet, AConv
Fine-tuning [34]	**90.1%**	VGGNet fine-tuning
Fine-tuning [35]	82.4%	CNN-S fine tuning
CNNaug-SVM [3]	77.2%	with augmented data
CNN-SVM [3]	73.9%	no augmented data
NUS [36]	70.5%	-
GHM [37]	64.7%	-
AGS [38]	71.1%	-
Texture Synthesis [21]	84.7%	using the Gram matrix on fc18 layer (VGG net) for classification

training set of the Alex net and the VGGVD net. So ImageNet can be seen as a superset of PASCAL-2007. The results on this dataset are shown in Table 3. From Table 3, we can see that again the best performance is achieved by using cross-layer pooling and the VGGVD net. Not surprisingly, the AConv layer performs better than the OConv layer in this dataset because the training categories of the DCNN overlaps with PASCAL-2007 and the AConv layer contains this category-level information. The per-class performance of three best comparing methods, that is, CNN jitter with the VGGVD net, SCFV with the AConv layer from the VGGVD net and cross-layer pooling with the AConv layer from the VGGVD net, is shown in Table 4. As seen, the proposed cross-layer pooling achieves the best performance in most classes.

4.2 Ablation Study

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.

4.2.1 Using Different Convolutional Layers

First, we are interested in examining the performance of using convolutional layers other than the 4th and 5th convolutional layers in the Alex net and the conv5-2 and conv5-3 convolutional layers in the VGGVD net. We investigate the performance of using the 3rd and 4th convolutional layers for the Alex net and the conv5-2 and conv5-3 convolutional layers in the VGGVD net. The results are shown in Table 6. From the results, we can see that using 4-5th layers (conv5-3-4th) layers achieves superior performance over using the 3-4th layers (conv5-2-3th) layers. This is consistent with the observation in [19] that the deeper the convolutional layer, the better discriminative power it has.

TABLE 5
Comparison of Results Obtained by Using Cross-Layer Pooling with Fully-Connected Layers

Method	MIT-67	Birds200	PASCAL07
FC-18, FC-20 (VGGNet)	77.0%	63.2%	85.9%
conv5-3, FC-18 (VGGNet)	74.1%	68.5%	84.5%

TABLE 6
Comparison of Results Obtained by Using Different Convolutional Layers

Method	MIT-67	Birds200	PASCAL07
CL-3-4 (AlexNet)	59.4%	63.9%	66.3%
CL-4-5 (AlexNet)	**63.0%**	**73.5%**	**71.3%**
CL-conv5-2-3 (VGGNet)	73.7%	**77.0%**	82.2%
CL-conv5-3-4 (VGGNet)	**74.4%**	**77.0%**	**84.1%**

As discussed in Section 3.1, the process of extracting fully-connected layer activations from multiple local regions can be viewed as applying a special convolutional layer. Thus it is possible to perform cross-layer pooling on two fully-connected layers. For AConv layers, a new fully-connected layer is stacked and retrained. Certainly, it is also possible to directly use two fully-connected layers in an existing CNN without introducing new layers, but the computational cost can be higher due to the high-dimensionality of the resulting representation, e.g., $4{,}096\times1{,}000$. Also, it is possible to perform cross-layer pooling on an original convolutional layer and a fully-connected layer in the above setting. In such cases, multiple spatial units in a convolutional layer will correspond to one fully-connected layer output, to apply cross-layer pooling we can either flatten activations from multiple spatial units into a long vector or using the pooled activation from multiple spatial units. In this paper, we use the latter approach since it produces lower dimensional image representations.

In this section we conduct an experimental evaluation of the above two approaches to performing cross-layer pooling. Specifically, for the first approach, denoted as FC-FC cross-layer pooling, we use the activations from the first fully-connected layer (4,096 dimensions) and the last fully-connected layer (1,000 dimensions) in the VGG net to perform cross-layer pooling. PCA is applied to reduce the dimensionality of the first fully-connected layer activations to 2,000; for the second approach, denoted as FC-Conv cross-layer pooling, we use the activations from conv5-4 and the first fully-connected layer in VGG net and apply sum-pooling (followed by using square-root post-processing) to pool the activations of conv5-4. Thus the dimensionality of the representation obtained from cross-layer pooling will be $512\times4{,}096$. The performance of these two methods is shown in Table 5. From Table 5 we make the following two observations:

TABLE 4
Comparison of Results on Pascal VOC 2007 for Each of 20 Classes

	TV	train	sofa	sheep	plant	person	mbike	horse	dog	table
Global Jitter (VGGVD)	81.9	96.3	72.9	86.1	61.6	95.2	89.3	91.5	90.9	79.7
SCFV (VGGVD)	82.3	95.7	78.9	84.0	62.6	95.8	89.1	94.1	90.4	82.2
CrossLayer (VGGVD)	**85.3**	96.7	**80.1**	**87.6**	**64.6**	**96.7**	**91.7**	94.3	**93.0**	82.1
	cow	chair	cat	car	bus	bottle	boat	bird	bike	areo
Global Jitter (VGGVD)	77.8	67.4	92.1	91.2	85.2	56.6	92.8	92.9	90.4	97.1
SCFV (VGGVD)	79.8	66.5	92.4	91.5	86.1	59.6	90.7	93.2	90.9	96.6
CrossLayer (VGGVD)	**83.3**	**70.2**	**93.9**	**92.8**	**89.4**	59.0	**93.8**	**94.7**	**94.0**	97.8

The classes on which cross-layer pooling achieves significantly better performance are labeled with bold font.

TABLE 7
The Impact of PCA, ℓ_2 Normalization and Power Normalization

PCA	ℓ_2 normalization	power normalization	Result
Yes	Yes	Yes	**78.2%**
Yes	Yes	-	76.4%
Yes	-	Yes	77.1%
-	Yes	Yes	72.6%
Yes	-	-	69.7%
-	Yes	-	74.6%
-	-	Yes	73.2%
-	-	-	69.7%

(1) by cross-referencing the performance in Tables 1, 2, and 3, we observe that FC-FC cross-layer pooling achieves similar performance to cross-layer pooling using the AConv layer, but marginally worse. This may suggest that the good performance of the AConv layer based cross-layer pooling mainly comes from the cross-layer pooling strategy, although the re-trained new convolutional layer can further boost classification performance. (2) FC-Conv cross-layer pooling achieves better performance than FC-FC cross-layer pooling on Birds200 but it is inferior to FC-FC cross-layer pooling on MIT-67 and PASCAL-07. Considering that the pooling operation of FC-Conv cross-layer pooling is performed on the OConv layer, this observation is consistent with the conclusion in Tables 1 and 3, that is, cross-layer pooling on AConv layers leads to better performance than pooling with OConv layers for MIT-67 and PASCAL-07.

4.2.2 The Impact of PCA and Normalization

In our implementation, we have applied three operations to obtain the final image representation, that is, performing PCA on the local feature, performing ℓ_2 normalization on each pooled coding vector and power normalization. In this section, we investigate the impact of those three operations. We conduct our experiment on MIT-67 with the AConv layer features and test the performance under various settings of those three operations. Table 7 shows the results. From Table 7, we observe some interesting phenomena: (1) The three operations have a big impact on the performance. If none of them is applied, the performance drops significantly. (2) Applying either ℓ_2 normalization or power normalization leads to similar performance improvement. (3) Applying PCA with normalization, ℓ_2 normalization or power normalization or both of them, can lead to further performance improvement. (4) The best performance is obtained by applying all three operations together.

TABLE 8
Comparison of Alternative Pooling Methods

Method	Conv5-3	Conv5-4	Conv5-3 + Conv5-4
max-pooling, SPM level0	57.3/67.7/75.5	60.7/69.0/80.5	60.6/55.8/77.8
max-pooling, SPM level1	66.3/67.6/76.8	67.7/70.2/81.6	66.0/59.1/78.9
max-pooling, SPM level2	68.0/67.6/76.8	69.3/69.9/81.9	67.9/61.5/79.8
sum-sqrt-pooling, SPM level0	66.4/66.4/77.7	68.2/69.3/81.7	69.9/65.8/80.9
sum-sqrt-pooling, SPM level1	68.0/64.1/77.5	70.0/69.8/81.7	70.7/63.6/80.6
sum-sqrt-pooling, SPM level2	69.0/64.1/77.4	70.3/69.8/81.9	70.9/64.1/80.7

Two pooling methods, max-pooling and sum-pooling (with square-root post-processing) are applied. Different levels of spatial pyramid [40] configuration are applied for both pooling methods. Level 0: pooling over the whole image (1×1); level 1: 1×1+2×2; level 2: 1×1+2×2+4×4; Those pooling methods are applied to the conv5-3 layer, the conv5-4 layer and the concatenation of conv5-3 and conv5-4 layers of the VGG network. The experimental comparison is made on MIT67, Birds200, and PASCAL07 three datasets. Results are reported in the order MIT67/Birds200/PASCAL07. For reference, cross-layer pooling on conv5-3 and conv5-4 achieves classification accuracy 74.4/77.0/84.1, which is higher than the results of all the alternative pooling methods.

TABLE 9
Results Obtained by Using Feature Sign Quantization

Dataset	Feature sign quantization	Original
MIT-67 (VGG, Aconv)	77.9%	78.2%
Birds-200 (VGG, OConv)	76.5%	77.0%
PASCAL07 (VGG, AConv)	85.1%	87.0%

4.2.3 Feature Sign Quantization

As has been discussed in Section 3.4, feature sign quantization is a promising strategy to reduce the memory cost of cross-layer pooling. Here we demonstrate the effect of applying feature sign quantization. Feature sign quantization quantizes a feature to 1 if it is positive, -1 if it is negative and 0 if it equals 0. In other words, we use 2 bits to represent each dimension of the pooled feature vector. This scheme greatly reduces the memory use. Here we only report the result on the best performing setting for each dataset. The results are shown in Table 9. As can be seen, this coarse quantization scheme does not degrade the performance much, and for two datasets, MIT-67 and Birds-200, it achieves almost the same performance as the original feature. Note that a similar quantization scheme has been also explored in [39], however there is caused a significant performance drop if applied to convolutional layer features. For example, in the Table 7 of [39], by binarizing conv-5, the performance drops around 5 percent. In contrast, our representation seems to be less sensitive to this coarse quantization.

4.3 Comparison with Alternative Pooling Methods

Finally, we compare several alternative pooling strategies against cross-layer pooling. The baseline method that we compare against directly pools convolutional layers. There are many possible variants of such an approach, however, which we characterize according to three criteria:

- Pooling methods. We consider both sum-pooling with square-root post-processing (which is better than direct sum-pooling) and max-pooling.
- Spatial pyramids [40]. Three spatial pyramid partitions, that is, the 1×1 (level 0), $1 \times 1 + 2 \times 2$ (level 1) and $1 \times 1 + 2 \times 2 + 4 \times 4$ (level3) are considered.
- Pooling layers. The conv5-3 layer, conv5-4 layer and the concatenation of conv5-3 and conv5-4 layers of the VGG net are considered.

The classification results on MIT67, Birds200 and PASCAL07 are reported in Table 8. As can be seen, the best performance achieved by tuning those pooling strategies is 69.0 percent on MIT67, 64.1 percent on Birds200 and 77.4 percent on PASCAL07. In comparison, the cross-layer pooling counterpart achieves 74.4, 77.0 and 84.1 percent on MIT67, Birds200 and PASCAL07 respectively, which is much better than the traditional pooling approaches.

Another possible variation of cross-layer pooling is to change the channel pooling method from sum-pooling to max-pooling. We also tried this setting and achieved 72.8 percent on MIT67, 71.6 percent on Birds200 and 85.2 percent on PASCAL07. As can be seen, its performance is worse than sum-pooling. Thus we suggest using sum-pooling as the default pooling method for cross-layer pooling.

4.4 Experiments for Image Retrieval

For image retrieval, we evaluate cross-layer pooling on the Oxford5K [41], Holiday [42] and Sculpture6K [43] datasets. These three datasets represent several common scenarios in image retrieval. The objects-of-interest in Oxford5K are buildings which have rich texture patterns. For the Holiday dataset, the to-be-retrieved images are more general scenes and objects. The sculpture6K dataset focuses on sculptures which have relatively smooth surfaces. We adopt a similar setting to [5] to evaluate performance

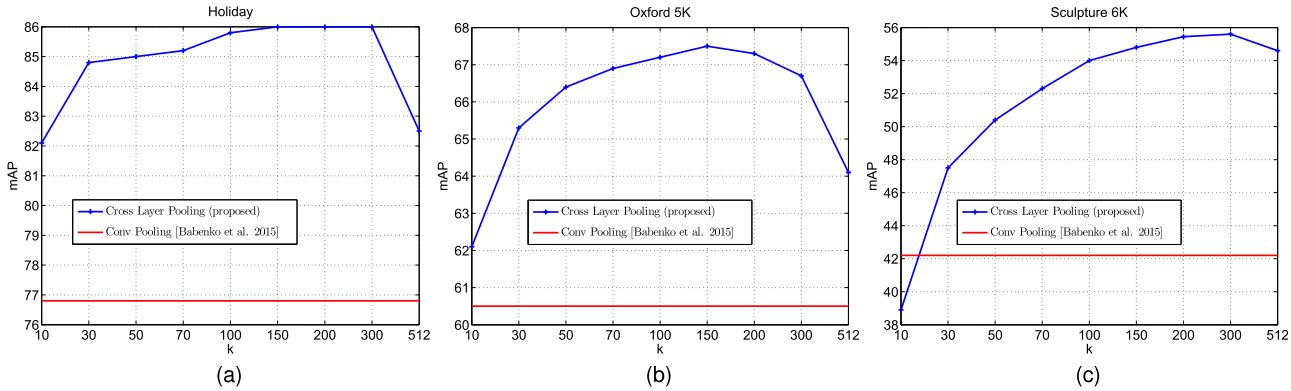

Fig. 4. Performance of cross-layer pooling on image retrieval. For our method, the feature maps of the conv5-4 layer with top k average activations are selected as pooling channels.

by directly using the query image without the object bounding box. We re-implemented the baseline method of [5] by strictly following its experimental protocol. Specifically, we apply PCA whitening and calculate the PCA projection matrix from external datasets. For Oxford5K, we learn the PCA projection matrix from Paris6K; for Holiday, we learn the PCA projection matrix from 5 K Flickr images (although those 5 K Flickr images might not be same as the one used in [5]); for Sculpture, we also calculate the PCA projection matrix from the same 5 K Flickr images. The VGG net is used in this experiment, the baseline in [5] is performed on the conv5-4 layer since it leads to the best performance. Our approach is applied on conv5-3 and conv5-4 layers. We use binarized cross-layer pooling vectors and select the pooling channels with top-k largest average activation value on conv5-4. Different k are evaluated and the comparison of cross-layer pooling and the baseline method in [5] is shown in Fig. 4. From Fig. 4, it is clear that cross-layer pooling performs much better than the direct convolutional layer pooling baseline [5] once a sufficiently large k is chosen. Selecting $k = 50$ is typically sufficient to achieve good performance. Considering that all the features are binarized, the computational cost is still reasonable despite the fact that the dimensionality is higher than that of the baseline method. Note that if we directly binarize the feature obtained from the convolutional layer pooling baseline, this leads to significant performance drop, while our method does not. Also, it is interesting to discover that when k becomes too large, that is, when we are close to using all of the available pooling channels, the retrieval performance will start to drop. This is probably because including channels with small average activation values tend to introduce more noise during retrieval.

5 Conclusion

We have proposed a new method termed cross-convolutional layer pooling to create image representations from the activations of two consecutive convolutional layers of a pre-trained CNN. We realize this idea on two types of implementations of convolutional layers and show that these two different implementations are particularly well suited to different recognition tasks. Also, we propose a variation on the cross-convolutional layer pooling approach for the image retrieval task. By conducting experiments on popular image classification datasets and image retrieval datasets, we show that the proposed method leads to superior performance over various existing methods of using a pre-trained DCNNs to extract image representations.

Acknowledgments

This work was in part supported by the Data to Decisions Cooperative Research Centre; and Australian Research Council Future Fellowship (FT120100969), and Australian Research Council projects DP160103710, and LP130100156. C. Shen is the corresponding author.

References

[1] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Advances 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. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.
[3] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: An astounding baseline for recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshop*, 2014, pp. 512–519.
[4] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson, "From generic to specific deep representations for visual recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshop*, 2015, pp. 36–45.
[5] A. Babenko and V. S. Lempitsky, "Aggregating deep convolutional features for image retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1269–1277.
[6] Y. Gong, L. Wang, R. Guo, and S. Lazebnik, "Multi-scale orderless pooling of deep convolutional activation features," in *Proc. 13th Eur. Conf. Comput. Vis.*, 2014, pp. 392–407.
[7] 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. 27th Int. Conf. Advances Neural Inf. Process. Syst.*, 2014, pp. 1143–1151.
[8] Y. Li, L. Liu, C. Shen, and A. van den Hengel, "Mid-level deep pattern mining," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 971–980.
[9] T. Darrell, N. Zhang, and R. Farrell, "Pose pooling kernels for sub-category recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3665–3672.
[10] L. Liu, C. Shen, and A. van den Hengel, "The treasure beneath convolutional layers: Cross-convolutional-layer pooling for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 4749–4757.
[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Representations*, 2014.
[12] Y. Li, L. Liu, C. Shen, and A. van den Hengel, "Mid-level deep pattern mining," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 971–980.
[13] M. Cimpoi, S. Maji, and A. Vedaldi, "Deep filter banks for texture recognition and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3828–3836.
[14] R. B. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1440–1448.
[15] T. Lin, A. RoyChowdhury, and S. Maji, "Bilinear CNN models for fine-grained visual recognition," *Proc. IEEE Int. Conf. Comp. Vis.*, 2015.
[16] D. Yoo, S. Park, J. Lee, and I. S. Kweon, "Fisher kernel for deep neural activations," in *arXiv preprint*, 2014.
[17] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3431–3440.
[18] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the Fisher kernel for large-scale image classification," in *Proc. 11th Eur. Conf. Comput. Vis.*, 2010, pp. 143–156.
[19] M. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. 13th Eur. Conf. Comput. Vis.*, 2014, pp. 818–833.
[20] P. Welinder, et al., "Caltech-UCSD birds 200," California Inst. Technol., Pasadena, CA, USA, Tech. Rep. CNS-TR-2010–001, 2010.
[21] L. A. Gatys, A. S. Ecker, and M. Bethge, "Texture synthesis using convolutional neural networks," in *Proc. Advances Neural Inf. Process. Syst.*, May 2015, pp. 262–270.

[22] I. Ustyuzhaninov, W. Brendel, L. A. Gatys, and M. Bethge, "Texture synthesis using shallow convolutional networks with random filters," *arXiv*, 2016.
[23] L. A. Gatys, A. S. Ecker, and M. Bethge, "Image style transfer using convolutional neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2414–2423.
[24] L.-J. Li, H. Su, E. P. Xing, and F.-F. Li, "Object bank: A high-level image representation for scene classification and semantic feature sparsification," in *Proc. Advances Neural Inf. Process. Syst.*, 2010, pp. 1378–1386.
[25] Y. Li, L. Liu, C. Shen, and A. van den Hengel, "Mining mid-level visual patterns with deep CNN activations," *Int. J. Comput. Vis.*, pp. 1–21, 2016.
[26] C. Doersch, A. Gupta, and A. A. Efros, "Mid-level visual element discovery as discriminative mode seeking," in *Proc. Advances Neural Inf. Process. Syst.*, 2013, pp. 494–502.
[27] M. Pandey and S. Lazebnik, "Scene recognition and weakly supervised object localization with deformable part-based models," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 1307–1314.
[28] A. Quattoni and A. Torralba, "Recognizing indoor scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 413–420.
[29] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "PASCAL visual object classes challenge 2007 (VOC2007) results," (2007). [Online]. Available: http://host.robots.ox.ac.uk/pascal/VOC/
[30] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, pp. 27:1–27:27, 2011. [Online]. Available: http://www.csie.ntu.edu.tw/ cjlin/libsvm
[31] N. Zhang, J. Donahue, R. Girshick, and T. Darrell, "Part-based R-CNNs for fine-grained category detection," in *Proc. 13th Eur. Conf. Comput. Vis.*, 2014, pp. 834–849.
[32] J. Donahue, et al., "DeCAF: A deep convolutional activation feature for generic visual recognition," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 647–655.
[33] N. Zhang, R. Farrell, F. Iandola, and T. Darrell, "Deformable part descriptors for fine-grained recognition and attribute prediction," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 729–736.
[34] R.-W. Zhao, J. Li, Y. Chen, J.-M. Liu, Y.-G. Jiang, and X. Xue, "Regional gating neural networks for multi-label image classification," in *Proc. Brit. Mach. Vis. Conf.*, 2016.
[35] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," in *Proc. Brit. Mach. Vis. Conf.*, 2014.
[36] Z. Song, Q. Chen, Z. Huang, Y. Hua, and S. Yan, "Contextualizing object detection and classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2011, pp. 1585–1592.
[37] Q. Chen, Z. Song, Y. Hua, Z. Huang, and S. Yan, "Hierarchical matching with side information for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3426–3433.
[38] J. Dong, W. Xia, Q. Chen, J. Feng, Z. Huang, and S. Yan, "Subcategory-aware object classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2013, pp. 827–834.
[39] P. Agrawal, R. Girshick, and J. Malik, "Analyzing the performance of multilayer neural networks for object recognition," in *Proc. 13th Eur. Conf. Comput. Vis.*, 2014, pp. 329–344.
[40] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2006, pp. 2169–2178.
[41] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2007, pp. 1–8.
[42] M. Douze, H. Jegou, and C. Schmid, "Hamming embedding and weak geometry consistency for large scale image search," in *Proc. 10th Eur. Conf. Comput. Vis.*, 2008, pp. 304–317.
[43] R. Arandjelović and A. Zisserman, "Smooth object retrieval using a bag of boundaries," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2011, pp. 375–382.