Analyzing Filters Toward Efficient ConvNet

Takumi Kobayashi
National Institute of Advanced Industrial Science and Technology, Japan
takumi.kobayashi@aist.go.jp

Abstract

Deep convolutional neural network (ConvNet) is a promising approach for high-performance image classification. The behavior of ConvNet is analyzed mainly based on the neuron activations, such as by visualizing them. In this paper, in contrast to the activations, we focus on filters which are main components of ConvNets. Through analyzing two types of filters at convolution and fully-connected (FC) layers, respectively, on various pre-trained ConvNets, we present the methods to efficiently reformulate the filters, contributing to improving both memory size and classification performance of the ConvNets. They render the filter bases formulated in a parameter-free form as well as the efficient representation for the FC layer. The experimental results on image classification show that the methods are favorably applied to improve various ConvNets, including ResNet, trained on ImageNet with exhibiting high transferability on the other datasets.

1. Introduction

In computer vision fields, deep convolutional neural network (ConvNet) [33, 30, 8] is successfully applied to such as image classification with producing state-of-the-art performance. A large number of convolution layers are stacked in the deep ConvNet to extract effective image features and back-propagation is applied in an end-to-end manner to train a huge number of parameters, *i.e.*, convolution filters, contained in those layers. For effectively training the networks, large-scale annotated data (ImageNet [12]) is exploited and some techniques such as stochastic gradient descent [47, 67], rectified linear unit (ReLU) [40], DropOut [51] and BatchNormalization [24], are proposed while the network architectures are also steadily improved [30, 8, 20, 63].

In the other direction from enthusiastically improving performance of ConvNets, there are some works toward analyzing ConvNets [68, 36, 39, 14, 3]. Most of them analyze *neuron activations* in the ConvNets such as by visualizing them for understanding how the ConvNets work on images. Though the filters in the first convolution layer are obviously visible, it is difficult to see how the second and the later layers operate on an input image domain. Thus, they are visualized in a pseudo manner using exemplars via tracking the high neuron activations [68] through DeConvNet [69], inverting them by means of back-propagation [36, 39] or estimating inverse function [14]. Those methods are useful for visually evaluating the invariance achieved by ConvNets and [68] guides us to slightly modify the network architecture for improving classification performance. Differently from such visualization, the neuron activations are statistically analyzed in [3] to derive feature representation based on a simple statistical model.

In this study, we analyze *filters* of ConvNets toward improving their efficiency, rather than analyzing the behavior of ConvNets. It is thus in contrast to the above-mentioned previous works focusing on the neuron activations. The ConvNets are composed of convolution, fully-connected (FC) and classification (multilayer perceptron: MLP) layers, where the fully-connected layer links the convolution layers with the MLP classifier. Note that we distinguish the fully-connected and MLP layers both of which may have been grouped and referred to as 'fully-connected' layers in the other studies; the fully-connected layer that we mention in this paper is `fc6` while we call `fc7` and `fc8` by MLP layers, if we follow the conventional naming of layers [30, 8, 50]. We can see that the filters in the convolution and FC layers have spatial dimensions larger than 1×1, exhibiting structural patterns through learning ConvNets, and thus we analyze the *spatial* structure in the *trained* filters.

The main contributions of this work are three-fold: (1) Through analyzing the filters sampled from various pre-trained ConvNets, we present the methods to reformulate both convolution and FC filters, providing parameter-free basis filters and efficient representation for FC layer, to improve ConvNets in terms of memory consumption and classification performance. (2) We conducted thorough experiments on an ImageNet classification task to validate the methods on various types of ConvNets which contain convolution as well as FC layers, with demonstrating the favorable transferability of the improved ConvNet onto the other

2575-7075/18 $31.00 © 2018 IEEE
DOI 10.1109/CVPR.2018.00589

IEEE computer society

datasets. (3) We also show that by similarly analyzing the convolution filters of ResNet [20], the methods are favorably applied even to the ResNet which excludes FC layer.

2. Analysis of convolution filters

We first analyze convolution filters which are main building blocks in ConvNets. As in most ConvNets, we assume isometric convolution filters denoted by $\boldsymbol{W} \in \mathbb{R}^{s\times s\times C}$ with the filter size s and the number of input channel C; here, we further assume a filter of odd size $s=2r+1$ which is a common shape in ConvNets. The filters are applied to extract *spatial* characteristics as well as correlations among input channels from the feature maps including an input image at the first layer. According to the long history of image processing/analysis, we can naturally suppose that the filters are related to *derivatives* [15]. And, considering convolution operation, it is also necessary to consider *rotation* of the filters in disregard of spatial translation.

2.1. Basis filters

The above discussion inspires us to employ steerable filters [16] for representing the convolution filters. They are advantageous in describing any rotated filters by a linear combination of a few steerable filters and are quite efficient in the case of Gaussian derivative filters [16]. Thus, our goal in this section is to find out the general form of *basis* filters that describe any convolution filters in ConvNets. It is also practically beneficial since the number of parameters in ConvNets are reduced through efficiently re-parameterizing convolution filters by means of the bases.

It, however, is hard to determine such bases for arbitrary convolution filters $\boldsymbol{W}$ of any depth C. Thus, we decompose (slice) the filter $\boldsymbol{W}$ into $\{\boldsymbol{w}_c\}_{c=1}^{C}$ such that $\boldsymbol{W} * \boldsymbol{Z} = \sum_{c=1}^{C} \boldsymbol{w}_c * \boldsymbol{z}_c$ where $*$ indicates a convolution operator and $\boldsymbol{w}_c \in \mathbb{R}^{s\times s}$ and $\boldsymbol{z}_c \in \mathbb{R}^{W\times H}$ are the c-th slices of the filter $\boldsymbol{W}$ and the input feature map $\boldsymbol{Z} \in \mathbb{R}^{W\times H\times C}$ along the channel dimension, respectively. We formulate the general bases for the spatial filters $\boldsymbol{w}_c$ of which 2D shape is shared across any channel-wise convolution filters. Suppose we are given B basis filters $\{\boldsymbol{b}_i\}_{i=1}^{B}$, the convolution filter $\boldsymbol{w}_c$ can be formulated by

$$\boldsymbol{w}_c = \sum_{i=1}^{B} \alpha_{ic}\boldsymbol{b}_i \;\Rightarrow\; \boldsymbol{W} * \boldsymbol{Z} = \sum_{i,c=1}^{B,C} \alpha_{ic}\boldsymbol{b}_i * \boldsymbol{z}_c \quad (1)$$

where $\{\alpha_{ic}\}_{i=1,c=1}^{B,C}$ are the coefficients of the bases to construct the filter $\boldsymbol{W}$. Considering that there are D filters $\{\boldsymbol{W}^{(j)}\}_{j=1}^{D}$ for D outputs and the bases are shared across them, (1) shows that the number of (learnable) parameters for the convolution filter is reduced into BCD from s^2CD.

For efficient representation, we can impose the constraint of *orthonormality* on the basis filters, and thus the *orthonormal steerable filters* could be basis for convolution filters. For example, the orthonormal steerable filters of up to 2nd order are analytically given as

$$\boldsymbol{b}^{[0]}(x,y) \propto e^{\frac{-1}{2\sigma^2}(x^2+y^2)}, \quad (2)$$
$$\boldsymbol{b}_0^{[1]}(x,y) \propto y e^{\frac{-1}{2\sigma^2}(x^2+y^2)},\ \boldsymbol{b}_1^{[1]}(x,y) \propto x e^{\frac{-1}{2\sigma^2}(x^2+y^2)},$$
$$\boldsymbol{b}_0^{[2]}(x,y) \propto (x^2+y^2-\sigma^2) e^{\frac{-1}{2\sigma^2}(x^2+y^2)},$$
$$\boldsymbol{b}_1^{[2]}(x,y) \propto (x^2-y^2) e^{\frac{-1}{2\sigma^2}(x^2+y^2)},\ \boldsymbol{b}_2^{[2]}(x,y) \propto xy e^{\frac{-1}{2\sigma^2}(x^2+y^2)},$$

where the superscript index in $\boldsymbol{b}$ indicates the order of derivatives and σ is the standard deviation of Gaussian envelope function, which are two parameters in the bases. The bases can be practically computed via Gram-Schmidt orthonormalization; for the practical algorithm, refer to the supplementary material. In Fig. 1, our basis filters (2) of 3×3 with $\sigma=1$ are compared to the actual bases computed by applying singular value decomposition (SVD) to the 3×3 convolution filters in the pre-trained VGG-vd-16 ConvNet[1] [50]. We can find well correspondence between them, which demonstrates feasibility to employ the orthonormal steerable filters for the bases of convolution filters.

While the filter size is designed by users, we then perform analysis for specifying two parameters of the derivative order N and the envelope size σ in the bases, both of which would be dependent on the filter size r. Note that we hereafter regard the N-th order bases as all the orthonormal steerable basis filters of up to N-th derivative order; the N-th order bases contain $\frac{1}{2}(N+1)(N+2)$ basis filters [16].

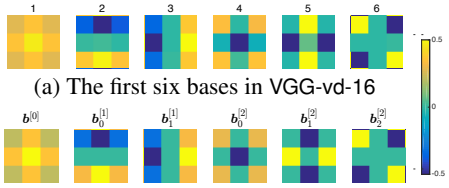
To find the relationship between those parameters (N,σ) and the filter size (r), we define the following criterion to evaluate how well the bases explain the optimized convolution filters in the pre-trained ConvNets;

$$\tilde{\epsilon}(\boldsymbol{W};\mathcal{B}) = 1 - \frac{\sum_c \| \sum_{\boldsymbol{b}\in\mathcal{B}} \mathtt{vec}(\boldsymbol{b})\mathtt{vec}(\boldsymbol{b})^\top \mathtt{vec}(\boldsymbol{w}_c)\|_2^2}{\sum_c \|\mathtt{vec}(\boldsymbol{w}_c)\|_2^2},$$
$$\epsilon(\mathcal{W};\mathcal{B}) = \frac{1}{|\mathcal{W}|}\sum_{\boldsymbol{W}\in\mathcal{W}} \tilde{\epsilon}(\boldsymbol{W};\mathcal{B}), \quad (3)$$

where $\mathtt{vec}$ is an operator to vectorize a filter (matrix), $\mathcal{B}=\{\boldsymbol{b}_i\}_{i=1}^{B}$ and $\mathcal{W}=\{\boldsymbol{W}^{(j)}\}_{j=1}^{D}$ are the banks of bases and convolution filters in the target convolution layer, respectively. It should be noted that the number of basis filter $|\mathcal{B}|$ depends only on the derivative order N, $|\mathcal{B}| = \frac{1}{2}(N+1)(N+2)$. $\tilde{\epsilon}$ measures reconstruction error by the bases $\mathcal{B}$ in comparison to the total filter energy $\sum_c \|\mathtt{vec}(\boldsymbol{w}_c)\|_2^2$, and ϵ averages them across the convolution filters $\mathcal{W}$ at the target convolution layer; it is further averaged across all convolution layers and various ConvNets. Thus, the smaller ϵ means that $\mathcal{B}$ fits the convolution filters well.

We draw the optimized convolution filters from 12 pre-trained ConvNets, all of which are trained on ImageNet dataset [12] and downloaded from [1]; they are listed in the supplementary material. Fig. 2a shows the results by fitting

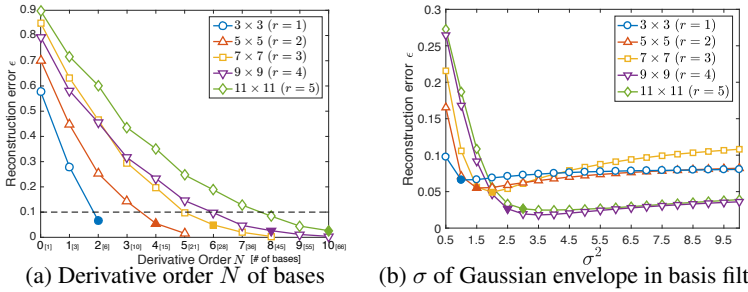
[1] All the convolution filters in VGG-vd-16 are in the shape of 3×3.



(a) The first six bases in VGG-vd-16

(b) Orthonormal steerable filters (2)

Figure 1. Comparison between (a) the 3×3 basis filters computed from the pre-trained VGG-vd-16 [50] and (b) ours by (2) of $\sigma = 1$. Weights are shown in pseudo colors.



(a) Derivative order N of bases (b) σ of Gaussian envelope in basis filter

Figure 2. Fitting the proposed orthonormal steerable bases to the convolution filters in the pre-trained ConvNets [1]. The results at the filled markers are used to establish Analysis 1.

to them the bases of various order N with $\sigma^2 = \frac{1}{2}(r+1)$ where r indicates the filter reach size, *i.e.*, $s = 2r+1$. The higher order produces larger number of bases, obviously decreasing error ϵ. Thus, based on $\epsilon < 0.1$ which means over 90% reconstruction by the bases, we can find that the derivative order of $N = 2r$ gives sufficiently favorable reconstruction. On the other hand, Fig. 2b shows the results for various σ in the bases of $N = 2r$, implying that there are minima. By seeing the minima on respective filter sizes, we roughly[2] derive the relationship of $\sigma^2 = \frac{1}{2}(r+1)$. As a result, our analysis can be summarized as follows:

Analysis 1: for convolution filters

For the convolution filter of $(2r+1)\times(2r+1)$ size, its bases are defined by the *orthonormal steerable filters* of up to order $\boxed{N = 2r}$ and the variance $\boxed{\sigma^2 = \frac{1}{2}(r+1)}$, producing $(r+1)(2r+1)$ basis filters.

2.2. Quantitative evaluation

We quantitatively evaluate Analysis 1 by applying it to the ConvNets based on VGG-M [8]. The ConvNets are re-parameterized by introducing the filter bases into the convolution layers as in (1) and are trained *from scratch* on ILSVRC2014 training dataset. We investigate the order N of the bases that directly affects the number of basis filters, while the effect of σ is limited to smoothing of filters.

In Table 1a, the basis set produced by Analysis 1 is compared to the original convolution filters without bases, the random orthonormal bases in which the numbers of basis filters are all the same as ours, and the steerable filter without orthogonalization [16]. Note that the original convolution filters implicitly assume the full-rank bases. By applying Analysis 1, the parameters for the convolution filters are significantly reduced by $\frac{(r+1)(2r+1)}{(2r+1)^2} \approx \frac{1}{2}$ while keeping the performance as well as outperforming the other bases.

The detailed comparison is also shown in Table 1b∼d by increasing/decreasing the order N of bases on the respective convolution sizes. Obviously, reduction of basis order degrades the performance, while less improvement is found even by increasing the order; actually the performances are slightly deteriorated, except for the case of 3×3 filter.

These results validate Analysis 1 of convolution filters from the viewpoint of classification performance, though it is derived from the approximation of the pre-trained filters.

[2]The 9×9 filter appears only in one layer, causing the observation of low confidence; see details of the ConvNets in the supplementary material.

2.3. Discussion and Related Works

By applying Analysis 1, convolution filters of any sizes are described by the linear combinations (1) of the *predefined* basis filters whose number is also pre-set. Thereby, the number of parameters to be learned is significantly reduced from s^2CD of whole filters $\{\boldsymbol{W}^{(j)}\}_{j=1}^{D}$ to BCD of coefficients where $B = (2r+1)(r+1) = s\frac{s+1}{2}$ is the number of basis filters. Thus, Analysis 1 roughly halves the parameter size (memory size) of any convolution layers in ConvNets. In the forward/backward-propagation, computation at the convolution layer can be decomposed into the convolutions by the fixed bases and their linear combination (implemented by 1×1 convolution) as in network-in-network model [35]. The off-the-shelf ConvNet learner is also applicable by reconstructing the convolution filters from the bases and their coefficients in negligible computation cost.

Basis (prefixed) filters are found in some works. ScatteringNet [49] is built upon the pre-defined wavelet filter banks which are cascaded with nonlinear operations to form the pre-fixed network. In [43], the pre-fixed Gabor filters are employed to extract orientation-based features only at the first layer, without considering the basis filters nor their (linear) combination. In contrast to those methods, we employ *orthonormal* steerable filters as bases and *learns* the coefficient weights for those pre-defined bases in an end-to-end manner. As the pre-defined basis filters reconstruct the convolution filters with the learnt weights, our method is generally applicable to various ConvNets without changing the network architecture as shown in Sec. 4. In [25, 10], basis filters are introduced to make ConvNets steerable in terms of certain types of transformation groups, while we leverage them to analyze the convolution filters toward improving efficiency of the ConvNets (Fig. 2 and Sec. 4.3). Our work is closely connected to [26] which directly employs Gaussian derivative filters with parameters to be tuned by users.

Table 1. Performance comparison on filter bases of various orders. The top-5 error rates (%) are measured by 10-crop testing [30] on ILSVRC2014 validation set. Numbers in parentheses indicate the number of basis filter, and the underlined ones mean the full-rank (s^2).

(a) VGG-M

Bases	Convolution filter size 7×7 1st layer	5 × 5 2nd layer	3 × 3 3∼5th layer	Error
original	- [49]	- [25]	- [9]	12.96
random	- [28]	- [15]	- [6]	13.65
steerable [16] w/o orthogonality	6 [28]	4 [15]	2 [6]	13.64
Analysis 1	**6 [28]**	**4 [15]**	**2 [6]**	**13.19**

(b) VGG-M

Bases	7×7 1st layer	5 × 5 2nd layer	3 × 3 3∼5th layer	Error
Analysis 1	**6 [28]**	**4 [15]**	**2 [6]**	**13.19**
steerable	*5 [21]*	4 [15]	2 [6]	13.34
steerable	*7 [36]*	4 [15]	2 [6]	13.26
steerable	6 [28]	*3 [10]*	2 [6]	13.40
steerable	6 [28]	*5 [21]*	2 [6]	13.26
steerable	6 [28]	4 [15]	*1 [3]*	13.93
steerable	6 [28]	4 [15]	*3 [9]*	13.12

(c) VGG-$M_{9\times9}$

Bases	9×9 1st layer	5 × 5 2nd layer	3 × 3 3∼5th layer	Error
steerable	*7 [36]*	4 [15]	2 [6]	13.27
Analysis 1	**8 [45]**	**4 [15]**	**2 [6]**	**13.21**
steerable	*9 [55]*	4 [15]	2 [6]	13.28

(d) VGG-$M_{11\times11}$

Bases	11×11 1st layer	5 × 5 2nd layer	3 × 3 3∼5th layer	Error
steerable	*9 [55]*	4 [15]	2 [6]	13.30
Analysis 1	**10 [66]**	**4 [15]**	**2 [6]**	**13.07**
steerable	*11 [78]*	4 [15]	2 [6]	13.26

However, through our analysis (Sec. 2.1), we provide the parameter-free basis filters only dependent on the filter size (Analysis 1). And, they are different in *orthogonality* which is necessary for effective learning as shown in Table 1a. In addition, it is noteworthy that our work covers the FC filter (Sec. 3) in contrast to the previous works [26, 25, 10] which only consider convolution filters.

From the viewpoint of filter decomposition, our method is related to the works for slimming ConvNets via decomposing convolution filters [27, 32, 13, 60, 59]. It, however, is totally different from them in that those methods decompose the trained filters a *posteriori* after learning the ConvNets and thus require re-training such as by fine-tuning, while the proposed Analysis 1 operates on convolution filters a *priori* before end-to-end learning. Therefore, those posterior methods would be applicable to the ConvNets that are learned according to Analysis 1 for further reducing the parameter size such as by decomposing the coefficients $\{\alpha_{ic}^{(j)}\}_{i=1,c=1,j=1}^{B,C,D}$ of the bases.

3. Analysis of fully-connected filters

The filter at the fully-connected (FC) layer, so-called `fc6`, works on a feature map in a different way from the convolution filters. Namely, the FC filter sees whole input without convolution and is applied mainly to compress the dimensionality by processing the feature map globally, while the convolution filter extracts local spatial characteristics. Considering that the FC filter operates on the whole spatial domain, a legacy image processing technique for image compression inspires us to employ the *discrete cosine transform (DCT)* [18] for representing the filter in an orthonormal manner; this is different motivation from the case of convolution filters based on derivatives (Sec. 2).

3.1. Basis filters

As is the case with convolution filters (Sec. 2.1), we consider the bases for the 2D spatial filter into which the FC filter is sliced along the channel dimension. Suppose the input feature map is defined on $\mathbb{R}^{S\times S\times C}$ of the spatial extent S, and the DCT bases are mathematically written by

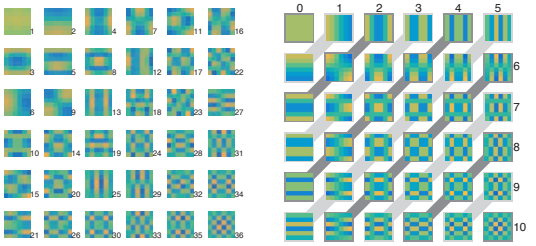
$$\boldsymbol{b}^{[m,n]}(x,y) = \beta_m\beta_n \cos\left\{\frac{m\pi}{2S}(2x+1)\right\} \cos\left\{\frac{n\pi}{2S}(2y+1)\right\}, \tag{4}$$

where $(x,y) \in \{0,\cdots,S-1\}\times\{0,\cdots,S-1\}$ and $\beta_0 = \frac{1}{\sqrt{S}}$, $\beta_i = \sqrt{\frac{2}{S}}$ for $0 < i \leq S-1$. The order N of the DCT bases is defined by $N = n+m$ for $0 \leq m,n \leq S-1$ as shown in Fig. 3b, and we regard the N-th order DCT bases to contain all the bases (4) of up to N-th order; $\mathcal{B} = \{\boldsymbol{b}^{[m,n]}\}_{m+n\leq N}$. N is the only parameter in the bases.

Similarly to Fig. 1, the actual bases at the FC layer are computed by applying SVD to the FC filter in the pre-trained VGG-M [8] and are depicted in Fig. 3a, exhibiting similarity to the DCT bases in Fig. 3b. Then, the relationship between the order N and the input size S is investigated over the pre-trained ConvNets of AlexNet [30], Caffe-reference [28] and VGG-S/F/M [8] for $S=6$ and VGG-vd-16/19 [50] for $S=7$. The reconstruction error ϵ (3) by the DCT bases is shown in Fig. 4. Note that the number of bases even in the same DCT order are different according to the spatial size S. As in Sec. 2.1, based on $\epsilon < 0.1$, the order can be determined as $N = S-1$.

3.2. Quantitative evaluation

The preliminary analysis result for DCT bases, $N = S$–1, is quantitatively evaluated in terms of classification performance. On ImageNet dataset, we train from scratch the ConvNet with Analysis 1 of VGG-M [8] containing the FC layer of $S=6$ to which we can apply the DCT bases of the order $N \in \{0,\cdots,10\}$. The performance results are shown in Fig. 5, leading to the following three findings: 1) the performance is kept at $N = S-1 = 5$ halving the number of parameters in the FC layer, 2) the lower-order bases give better performance and especially the best performance is achieved by $N=1$, and 3) even the 0-th order basis exhibits almost the same performance as the original one ($N=10$) while significantly reducing the number of parameters by $\frac{1}{S^2} = \frac{1}{36}$. Note that the 0-th order basis is just the uniform weighting (Fig. 3b) corresponding to *average-pooling*. Based on these results, we can say that the fully-



(a) The bases in VGG-M (b) DCT bases (4) on 6×6

Figure 3. Comparison between (a) the FC basis filters computed from the pre-trained ConvNet (VGG-M [8]) and (b) the DCT bases (4). The numbers in (b) indicate the DCT orders. Weights are shown in pseudo colors.

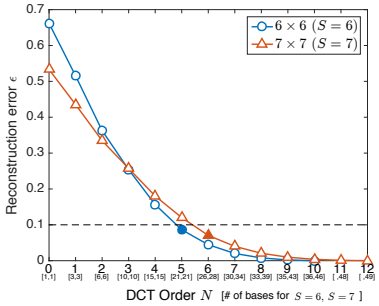


Figure 4. Reconstruction errors by fitting DCT bases to the FC filters.

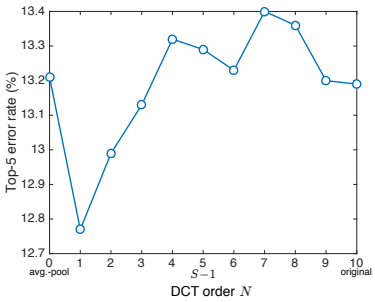


Figure 5. Classification performance by VGG-M of various DCT orders.

connected layer is redundantly parameterized, and in particular, the third finding further inspires us to reformulate the fully-connected layer as follows.

3.3. BoW-based representation for FC layer

Let $\boldsymbol{z}_{xy} \in \mathbb{R}^C$ be the C-dimensional feature vector at (x, y) position in the input feature map $\boldsymbol{Z} \in \mathbb{R}^{S\times S\times C}$. The fully-connected layer followed by the ReLU layer is mathematically written by, for one output channel,

$$\texttt{ReLU}\left[\sum_{x=1,y=1}^{S,S} \boldsymbol{w}_{xy}^\top \boldsymbol{z}_{xy} + \rho\right] \tag{5}$$

where $\texttt{ReLU}(x) \triangleq \max[0, x]$ and $\boldsymbol{w}_{xy}$ is the C-dimensional filter vector at (x, y) in the FC filter $\boldsymbol{W} \in \mathbb{R}^{S\times S\times C}$. By applying the 0-th order DCT basis which is equivalent to average-pooling followed by linear projection via $\boldsymbol{\alpha} \in \mathbb{R}^C$, the FC layer (5) is transformed into

$$\texttt{ReLU}\left[\frac{1}{S^2}\sum_{x=1,y=1}^{S,S} \boldsymbol{\alpha}^\top \boldsymbol{z}_{xy} + \rho\right]. \tag{6}$$

This formulation simply aggregates the score of local descriptor (feature) $\boldsymbol{z}_{xy}$ projected onto the vector $\boldsymbol{\alpha}$ which comprises the coefficients of the 0-th order DCT basis across channels. Such a process on the local descriptors belongs to the framework of bag-of-words (BoW) [11] which aggregates the code weight, *i.e.*, non-negative similarity score, of the visual words assigned to the local descriptors. Inspired by the BoW approach, (6) is modified into

$$\frac{1}{S^2}\sum_{x=1,y=1}^{S,S} \texttt{ReLU}[\boldsymbol{\alpha}^\top \boldsymbol{z}_{xy} + \rho]. \tag{7}$$

In (7), we regard $\boldsymbol{\alpha}$ as the visual word vector and the non-negative similarity score (code weight) to $\boldsymbol{\alpha}$ is measured by dot-product and thresholding (ReLU) with the bias ρ. The code weights at the local descriptors $\boldsymbol{z}_{xy}$ are average-pooled over the input map of $S\times S$ as in the standard BoW. Though (7) is simply implemented by swapping the sequence of ReLU and average-pooling in (6) as shown in Fig. 7b, it is noteworthy that (7) is theoretically founded on the BoW model which delivers an effective feature representation of medium level with favorable transferability (Sec. 4.2).

Following the progress of BoW [57, 23], we can apply *max*-pooling [6] to (7) instead of average-pooling:

$$\max_{x,y} \texttt{ReLU}[\tilde{\boldsymbol{w}}^\top \boldsymbol{z}_{xy} + \rho] = \texttt{ReLU}[\max_{x,y}\{\tilde{\boldsymbol{w}}^\top \boldsymbol{z}_{xy} + \rho\}]. \tag{8}$$

The max-pooling woks well on the hand-crafted local descriptors with sparse code weights [57, 23]. (8) is easily implemented by using the max-pooling layer.

In consequence, based on the findings in Sec. 3.2, we can conclude our analysis on FC filter as follows:

Analysis 2: for fully-connected filters

▷The bases of the FC filter are given by the *DCT bases* (4) of up to order $\boxed{N=1}$ producing 3 basis filters, or
▷$\boxed{\text{BoW-based FC layer (Sec. 3.3)}}$ should be applied.

3.4. Related Works

Some ConvNets [35, 52, 53, 20] heuristically replace the FC layer with average-pooling. On the other hand, we naturally introduce the average-pooling based on the analysis of the FC layer from the *frequency* viewpoint by DCT (Sec. 3.2), and then derive the BoW-based representation (7, 8). In contrast to the classification score pooling in [35], our model (7) produces effective BoW-based feature representation which can be extended via max-pooling (8) and is well transferable (Sec. 4.2). It is connected to but different from the feature-pooling in [52, 53, 20]; Sec. 4.3 shows that it favorably works to improve ResNet [20].

While in [9] the neuron activations at the intermediate convolution layer, such as `conv5`, are fed into Fisher kernel in the framework of transferring ConvNet features, we embed the BoW-based coding into ConvNets which is trained in an end-to-end manner. In NetVLAD [4], VLAD layer is similarly embedded and learned for place recognition. Our BoW-based layer (7, 8) is much simpler with lower dimensionality and is advantageous in transferring features.

As to FC layer decomposition, we introduce the DCT bases *a priori* while Fast R-CNN [17] decompose the FC filter *a posteriori* via SVD for accelerating detection.

4. Experimental Results

4.1. Image classification on ImageNet dataset

We apply Analysis 1&2 to various ConvNets of AlexNet [30] and VGG-S/M/F [8] as well as the deeper ConvNets of VGG-vd-16/19 [50]. Note that the ConvNets are re-parameterized by applying the proposed methods to the convolution layers as well as the fully-connected layer (Fig. 7). All the ConvNets of both the original and our models are trained *from scratch* on ILSVRC2014 training dataset[3]. We implemented them by using MatConvNet toolbox [55], following the learning procedure provided in the toolbox; the details are shown in the supplementary material. Performance is measured on ILSVRC2014 validation set and we report top-5 error rates by 10-crop testing [30] on a resized input image whose minimum side has 256 pixels.

Table 2 shows the performance results for the moderately deep ConvNets. Analysis 1 of convolution filters efficiently works to reduce the parameter size of the ConvNet while keeping the performance; see the reduction of convolution parameters in Fig. 6a. On AlexNet and VGG-F which produce rather higher error rates, the performance gap between ours and the original one is slightly larger than those for the more sophisticated ConvNets of VGG-M/S. The gap, however, is getting smaller as the ConvNets are improved so as to produce better performance, as described in the later.

As to the fully-connected filter, Analysis 2 regarding DCT bases of $N = 1$ effectively works, and it is noteworthy that the BoW representation (Sec. 3.3) contributes to further performance improvement; the max-pooling in (8) largely improves the performance especially on the simpler ConvNets of AlexNet and VGG-F. While the BoW representation is shift-invariant due to the global pooling in (7, 8), the FC layer using the DCT bases of $N=1$ would be effective such as for position-sensitive regression in YOLO [46]. The memory size of the ConvNets equipped with the BoW layer is significantly reduced as shown in Fig. 6a. While all these ConvNets contain the same MLP classifier of the fixed size, the other layers enjoy significant memory reduction due to the filter bases by Analysis 1 for the convolution layers and the BoW-based representation by Analysis 2 for the FC layer. The number of parameter at the FC layer is reduced to CD of word vectors $\{\boldsymbol{\alpha}^{(j)}\}_{j=1}^{D}$ from S^2CD where S is the spatial size of the input feature map at `fc6`.

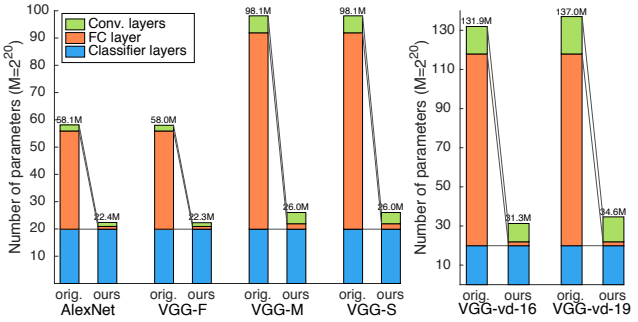
As to the spatial size S, as shown in Fig. 7a, the final convolution layer (`conv5`) is actually followed by local max-pooling in those ConvNets, mainly for reducing the size S. On the other hand, the proposed BoW layer (Fig. 7b) is not dependent on the spatial dimensions (S) as it is marginalized out via global avg/max-pooling. Thus, it is possible to simply remove the local pooling layer for *enlarging* the spatial size S. Increasing the number of local descriptors makes the statistical operation such as average and max in the pooling more stable. Such *dense*-BoW layer is applied to the ConvNets by simply removing the local pooling layer after `conv5` as shown in Fig. 7c, and the performance results are shown in Table 2. Even such a simple modification can further improve the performance on all the ConvNets. And, the above-mentioned performance gap between the original convolution and that of Analysis 1 on AlexNet and VGG-F is reduced under the dense-BoW setting. As a result, we conclude that on these ConvNets, Analysis 1 of convolution filters and the dense-BoW layer by Analysis 2 are effective in terms of both parameter size and performance.

The number of output channels in the BoW layer is regarded as that of visual words in the hand-crafted BoW framework, and its effect on the classification performance is shown in Fig. 8 where Analysis 1 and dense-BoW with max-pooling are applied to VGG-M. Note that the BoW layer is followed by the MLP classifier of which hidden layer comprises 4,096 neuron nodes for the 1,000-class soft-

[3]We confirmed that the original ConvNets of AlexNet∼ResNet trained by ourselves in Table 2&3&5 produce similar (or a bit better) performance to those provided by the authors in the 10-crop testing protocol.

Table 2. Top-5 error rates (%) on ILSVRC2014 validation set by the moderately deep ConvNets, following 10-crop testing [30].

Conv. Analysis 1	FC Analysis 2	ConvNet models AlexNet	VGG-F	VGG-M	VGG-S
-	-	16.58	16.36	12.96	13.19
✓	-	16.96	16.81	13.19	13.43
✓	DCT ($N=1$)	16.57	16.52	12.77	12.84
✓	BoW (avg-pool)	15.50	15.46	11.95	11.59
✓	BoW (max-pool)	14.58	14.84	11.95	11.60
✓	dense-BoW (avg)	14.70	14.98	11.95	11.57
✓	dense-BoW (max)	14.08	14.41	11.74	11.59
-	dense-BoW (max)	14.07	14.20	11.58	11.26

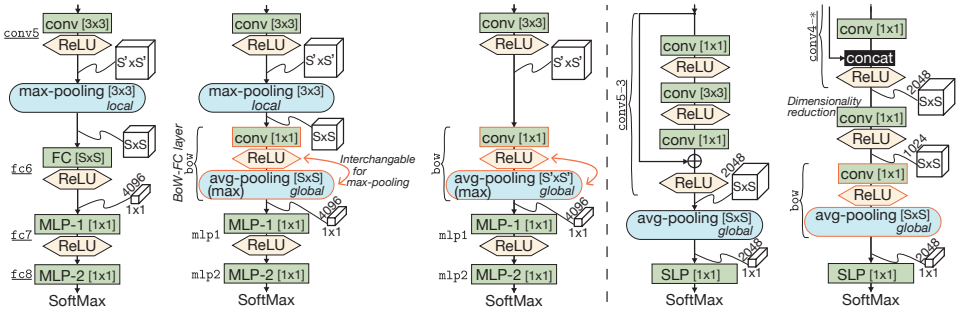


(a) Moderately deep ConvNets (b) Deeper ConvNets

Figure 6. The number of parameters in the ConvNets. We apply Analysis 1 and BoW layer (Analysis 2) to the ConvNets.

Table 3. Top-5 error rates (%) by the deeper ConvNets.

Conv. Analysis 1	FC Analysis 2	VGG-vd-16	VGG-vd-19
-	-	7.88	7.97
✓	-	8.24	8.03
✓	BoW (avg-pool)	7.13	7.03
✓	BoW (max-pool)	7.33	7.27
✓	dense-BoW (avg)	7.10	7.06



(a) ConvNet (b) ConvNet with BoW (c) dense-BoW (d) ResNet (e) ResNet-BoW

Figure 7. The BoW layer (Sec. 3.3) embedded in the ordinary ConvNets [30, 8, 50] and ResNet [20]. The conventional layer names are underlined, and the cuboids stand for data shape in feed-forwarding. The multi-/single-layered perceptron (MLP/SLP) is implemented by 1×1 convolution.

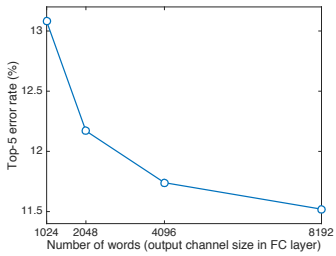


Figure 8. Classification performances on various number of words (output channels) in the BoW layer. The default number is 4,096.

max. Based on the trade-off between parameter size and the performance, it is favorable to employ 4,096 words, which corresponds to the original setting in VGG-M.

The proposed method is then applied to the deeper ConvNets of VGG-vd-16/19 [50]. The performance results shown in Table 3 and Fig. 6b demonstrate the effectiveness of the proposed methods in these deeper models. The max-pooling in the BoW layer, however, does not contribute to performance improvement, even degrading performance, in contrast to the case of moderately deep ConvNets (Table 2). The max-pooling operates *globally* on the input feature map, unlike the commonly used *local* max-pooling in ConvNets. Therefore, the *global* max-pooling at the FC layer makes back-propagation very sparse, which hampers end-to-end learning in the deeper (complex) models. This is also seen in Table 2 where the effectiveness of max-pooling is less on VGG-M/S of plenty parameters, compared to the simple ConvNets of AlexNet and VGG-F. And, the dense-BoW by removing local pooling after `conv5` is not so effective in VGG-vd-16/19. In the deeper models, the local descriptors produced by `conv5` are heavily overlapped in terms of the spatial receptive fields, and thus the local pooling less affects such highly *smoothed* local descriptors.

4.2. Transferability

Next, we evaluate the transferability [5, 38, 45] of the ConvNet which is pre-trained based on Analysis 1&2. We apply to various types of datasets the ConvNet of VGG-vd-16 which is frequently employed as a generic image feature extractor instead of hand-crafted ones. The original VGG-vd-16 is compared with ours to which Analysis 1 and the dense-BoW (avg-pool) layer by Analysis 2 are applied, without fine-tuning for fairly evaluating the transferability (generality) of the ConvNets pre-trained on ILSVRC2014 training dataset. Note that our VGG-vd-16 consumes significantly lower memory, less than $\frac{1}{4}$ of the original one, as shown in Fig. 6b. The image features are extracted by applying the pre-trained ConvNet in a convolution manner to a rescaled image which has 256 pixels on the minimum side, and then are max-pooled over the image region. The neuron activations at the intermediate layer, `fc6/7` for the original VGG-vd-16 and `bow`/`mlp1` for ours, are employed to produce 4,096-dimensional features; `bow` and `mlp1` are counterparts to `fc6` and `fc7`, respectively, as shown in Fig. 7. The features are finally classified by linear SVM [54] and classification accuracies are measured according to the standard protocol provided in the respective datasets; on Caltech256, we draw 60 training samples on each class, and for details of the other datasets, refer to the respective papers.

The performance results are shown in Table 4. The proposed method improves the performance of the original VGG-vd-16, being close to the state-of-the-art performances. It should be noted that our ConvNet produces high performance at the `bow` layer on all the datasets, exhibiting favorable transferability to various tasks. In contrast, the performance by the original one is dependent on the combination of the layer types (`fc6` or `fc7`) and target datasets, according to the similarity between the target task and the ImageNet classification as discussed in [5]. Note that most of the state-of-the-art methods are built on the pre-trained ConvNets, while we simply apply single VGG-vd-16 pre-trained on ImageNet to an image of single-scale. Thus, our ConvNet would contribute to further improvement in those methods, which is out of our focus in this paper.

4.3. Improving ResNet [20]

Finally, we analyze the deep ConvNet of ResNet [20] which stacks plenty of 3×3 convolution layers with single-layered perceptron (SLP) classifier. We focus on the 3×3 convolution and analyze them in the same manner as in Sec. 2.1 based on the reconstruction error ϵ by the six basis filters of $N = 2$. In Fig. 9, we can find that (1) the filters in the last block (`conv5`) exhibit quite less reconstruction errors, while (2) larger errors are produced at the other blocks, in comparison to the reference model of VGG-vd-19. The first finding suggests that the `conv5` block results in so simple filters as to be eliminated as redundant one. Thus, according to Analysis 2, we replace `conv5` (and

Table 4. Classification accuracy (%) on the various datasets of middle scale. The ConvNet of VGG-vd-16 pre-trained on ImageNet is applied without fine-tuning to extract image features, followed by a linear SVM. We also show the performance reported in the other papers.

		object classification		scene classification			fine-grained classification				others		
		VOC2007 [2]	Caltech256 [19]	Scene15 [31]	MIT67 [44]	SUN397 [61]	Bird200 [56]	Car196 [29]	Pet37 [42]	Flower102 [41]	FMD [48]	Event8 [34]	Action40 [66]
Orig.	fc6	87.63	82.98	93.35	75.66	58.12	68.13	82.69	89.64	93.03	83.57	98.40	75.88
	fc7	88.82	83.88	93.03	74.55	57.88	65.56	77.90	90.06	92.20	84.76	97.64	76.65
Ours	bow	88.85	**85.83**	**93.49**	**79.12**	**62.53**	**77.60**	**86.99**	91.06	**96.72**	**85.42**	98.33	**79.56**
	mlp1	**89.32**	85.52	93.35	78.62	60.32	71.03	81.98	**91.50**	94.94	85.06	**98.61**	78.67
Others		77.7 [38]	86.1 [7]	95.18 [21]	86.04 [21]	70.17 [21]	85.1 [37]	92.5 [58]	93.45 [3]	96.40 [3]	82.4 [9]	96.13 [71]	80.9 [7]
		73.9 [45]	85.06 [3]	92.9 [65]	81.0 [9]	62.97 [3]	76.6 [22]	86.3 [64]	90.03 [62]	94.8 [7]	81.6 [36]	95.42 [70]	72.03 [70]

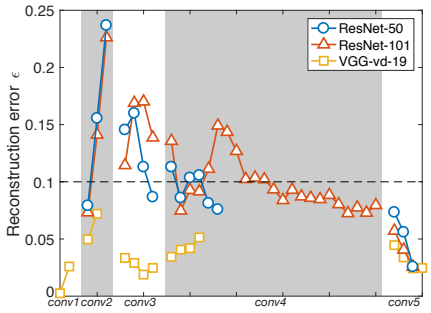


Figure 9. Reconstruction error ϵ (3) at respective 3×3 convolution layers in ResNet [20]; we show those of VGG-vd-19 for reference.

Table 5. Top-5 error rates (%) by ResNet-50 [20]. Numbers in parentheses indicate the number of parameters in the ConvNets. "same/half-dim" means the same/half number of input channels of 5×5 convolution compared to that of 3×3 convolution; the detailed architectures are shown in the supplementary material.

	conv2∼3	conv4	conv5	Error	Param.
a)	orig.	orig.	orig.	6.35	[24.35M]
b)	orig.	orig.	BoW	6.23	[14.09M]
c)	Analysis 1:3×3		orig.	6.42	[20.75M]
d)	Analysis 1:3×3		BoW	6.31	[12.74M]
e)	Analysis 1:3×3	Analysis 1:5×5 (same-dim.)	BoW	6.01	[16.12M]
f)	Analysis 1:3×3	Analysis 1:$\{3\times3, 5\times5$ (half-dim.)$\}$	BoW	5.71	[16.59M]
g)	Analysis 1:$\{3\times3, 5\times5$ (half-dim.)$\}$		BoW	5.50	[17.15M]
h)	Analysis 1:$\{3\times3, 5\times5$ (same-dim.)$\}$		BoW	5.22	[23.92M]

subsequent average-pooling layer) with the proposed BoW representation (7), as shown in Fig. 7e; in order to increase the discriminativity of the local descriptor fed into the BoW, the last *sum*-connection at conv4 is replaced with *concatenation* and it is followed by dimensionality reduction via $1\times1\times2048\times1024$ convolution like PCA in BoW.

Then, the second finding suggests applying the filters of the higher order ($N > 2$) at the blocks of conv2∼4. According to Analysis 1, the higher-order derivatives can be realized by the larger-sized filters and we apply the 5×5 filter size, producing 15 basis filters of the order $N = 4$. The 5×5 filters are either replaced with 3×3 filters or embedded in the additional residual path to that of 3×3. We feed the 5×5 convolution a feature map of the same or half number of channel, compared to the 3×3 convolution; refer to the supplementary material for the detailed architecture.

The above two modifications are evaluated on ResNet-50 in Table 5. The BoW representation significantly reduces the parameter size while slightly improving the classification performance; combination (*d*) of Analysis 1 for conv2∼4 and Analysis 2 for conv5 halves the parameter size, maintaining the same performance as the original (*a*). This validates the first finding regarding the redundancy of conv5, which may be due to the preceding deep convolution layers that sufficiently characterize the image features. The second modification also improves the classification performance even by replacing the 3×3 filters with 5×5 only at the conv4 block (*e*). The performance is further improved by introducing both the 3×3 and 5×5 filters into residual paths at the blocks of conv2∼4 (*f*∼*h*). We finally achieve the performance (*h*) of 5.22% [23.92M] which is favorably competitive with ResNeXt-50 [63], 5.36%[4] [23.93M]. These results show that analyzing filters based on our bases (Sec. 2.1, Fig. 9) is useful to improve the ConvNet model and the proposed Analysis 1&2 work well for improving ResNet [20]; we would improve ResNeXt [63] in a similar way since the ResNeXt and ResNet share large portion of the network architecture.

5. Conclusion

We have proposed the methods to reformulate filters for improving ConvNets. Two types of methods are presented through analyzing filters at convolution and fully-connected (FC) layers, respectively, on various pre-trained ConvNets. We give light on the redundant parameterization of the filters, and introduce the filter bases of lower ranks by means of the orthonormal steerable filters and discrete cosine transform. The analysis on the FC layer further endows us with the BoW-based representation. The proposed methods can improve the existing ConvNets in terms of memory size and classification performance while keeping the overall network architecture. The experimental results on image classification show that they work effectively in various types of ConvNets, including ResNet, on ImageNet with exhibiting favorable transferability on the other dataset.

[4] We applied the pre-trained model of ResNeXt-50 (32x4d) [63] with 10-crop testing. And, our ResNet-101 with the configuration (*f*) produces 4.88% [39.75M] compared to 4.96% [42.26M] of ResNeXt-101 (32x4d).

References

[1] MatConvNet pre-trained models. http://www.vlfeat.org/matconvnet/pretrained/. Accessed: 2017-1-19.

[2] The PASCAL Visual Object Classes Challenge 2007 (VOC2007). http://www.pascal-network.org/challenges/VOC/voc2007/index.html.

[3] J. W. A. Y. AL. Xie, L. Zheng and Q. Tian. Interactive : Inter-layer activeness propagation. In *CVPR*, pages 270–279, 2016.

[4] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *CVPR*, pages 5297–5307, 2016.

[5] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson. Factors of transferability for a generic convnet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(9):1790–1802, 2016.

[6] Y. Boureau, J. Ponce, and Y. LeCun. A theoretical analysis of feature pooling in visual recognition. In *ICML*, pages 111–118, 2010.

[7] S. Cai, L. Zhang, and W. Z. X. Feng. A probabilistic collaborative representation based approach for pattern classification. In *CVPR*, pages 2950–2959, 2016.

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

[9] M. Cimpoi, S. Maji, and A. Vedaldi. Deep filter banks for texture recognition and segmentation. In *CVPR*, pages 3828–3836, 2015.

[10] T. S. Cohen and M. Welling. Steerable cnns. *arXiv,* 1612.08498, 2016.

[11] G. Csurka, C. Bray, C. Dance, and L. Fan. Visual categorization with bags of keypoints. In *ECCV Workshop on Statistical Learning in Computer Vision*, pages 1–22, 2004.

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009.

[13] E. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *NIPS*, pages 1269–1277, 2014.

[14] A. Dosovitskiy and T. Brox. Inverting visual representations with convolutional networks. In *CVPR*, pages 4829–4837, 2016.

[15] L. M. Florack, B. M. ter Haar Romeny, J. J. Koenderink, and M. A. Viergever. Scale and the differential structure of images. *Image and Vision Computing*, 10(6):376–388, 1992.

[16] W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(9):891–906, 1991.

[17] R. Girshick. Fast r-cnn. In *CVPR*, pages 1440–1448, 2015.

[18] R. Gonzalez and R. Woods. *Digital Image Processing*. Prentice Hall, 2007.

[19] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, Caltech, 2007.

[20] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[21] L. Herranz, S. Jiang, and X. Li. Scene recognition with cnns: objects, scales and dataset bias. In *CVPR*, pages 571–579, 2016.

[22] S. Huang, Z. Xu, D. Tao, and Y. Zhang. Part-stacked cnn for fine-grained visual categorization. In *CVPR*, pages 1173–1181, 2016.

[23] Y. Huang, K. Huang, Y. Yu, and T. Tan. Salient coding for image classification. In *CVPR*, pages 1753–1760, 2011.

[24] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Journal of Machine Learning Research*, 37:448–456, 2015.

[25] J.-H. Jacobsen, B. de Brabandere, and A. W. Smeulders. Dynamic steerable blocks in deep residual networks. In *BMVC*, pages 1–13, 2017.

[26] J.-H. Jacobsen, J. van Gemert, Z. Lou, and A. W. M. Smeulders. Structured receptive fields in cnns. In *CVPR*, pages 2610–2619, 2016.

[27] M. Jaderberg, A. Vedaldi, and A. Zisserman. Speeding up convolutional neural networks with low rank expansions. In *BMVC*, 2014.

[28] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv,* 1408.5093, 2014.

[29] J. Krause, M. Stark, J. Deng, and L. Fei-Fei. 3d object representations for fine-grained categorization. In *ICCV Workshop*, 2013.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.

[31] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2169–2178, 2006.

[32] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. In *ICLR*, 2015.

[33] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[34] L.-J. Li and L. Fei-Fei. What, where and who? classifying events by scene and object recognition. In *ICCV*, 2007.

[35] M. Lin, Q. Chen, and S. Yan. Network in network. *CoRR,* abs/1312.4400, 2013.

[36] T.-Y. Lin and S. Maji. Visualizing and understanding deep texture representations. In *CVPR*, pages 2791–2799, 2016.

[37] T.-Y. Lin, A. RoyChowdhury, and S. Maji. Bilinear models for fine-grained visual recognition fine-grained visual recognition. In *ICCV*, pages 1449–1457, 2015.

[38] I. L. M. Oquab, L. Bottou and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*, pages 1717–1724, 2014.

[39] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *CVPR*, pages 5188–5196, 2015.

[40] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814, 2010.

[41] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, 2008.

[42] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar. Cats and dogs. In *CVPR*, pages 3498–3505, 2012.

[43] J. A. Pérez-Carrasco, B. Zhao, C. Serrano, B. Acha, T. Serrano-Gotarredona, S. Cheng, and B. Linares-Barranco. Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing – application to feedforward convnets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2706–2719, 2013.

[44] A. Quattoni and A. Torralba. Recognizing indoor scenes. In *CVPR*, pages 413–420, 2009.

[45] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *CVPR Workshop*, pages 512–519, 2014.

[46] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, pages 779–788, 2016.

[47] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[48] L. Sharan, R. Rosenholtz, and E. Adelson. Material perception: What can you see in a brief glance? *Journal of Vision*, 9(8):784, 2009.

[49] L. Sifre and S. Mallat. Rotation, scaling and deformation invariant scattering for texture discrimination. In *CVPR*, pages 1233–1240, 2013.

[50] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[51] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout : A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[52] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, pages 1–9, 2015.

[53] C. Szegedy, V. Vanhoucke, S. Ioffe, and J. Shlens. Rethinking the inception architecture for computer vision. *arXiv,* 1512.00567, 2015.

[54] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[55] A. Vedaldi and K. Lenc. MatConvNet – convolutional neural networks for matlab. In *ACM MM*, 2015.

[56] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[57] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-constrained linear coding for image classification. In *CVPR*, pages 3360–3367, 2010.

[58] Y. Wang, J. Choi, V. I. Morariu, and L. S. Davis. Mining discriminative triplets of patches for fine-grained classification. In *CVPR*, pages 1163–1172, 2016.

[59] Z. Wang, Z. Deng, and S. Wang. Accelerating convolutional neural networks with dominant convolutional kernel and knowledge pre-regression. In *ECCV*, pages 533–548, 2016.

[60] K. H. X. Zhang, J. Zou and J. Sun. Accelerating very deep convolutional networks for classification and detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(10):1943–1955, 2016.

[61] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010.

[62] L. Xie, R. Hong, B. Zhang, and Q. Tian. Image classification and retrieval are one. In *International Conference on Multimedia Retrieval (ICMR)*, pages 3–10, 2015.

[63] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *CVPR*, pages 5987–5995, 2016.

[64] S. Xie, T. Yang, X. Wang, and Y. Lin. Hyper-class augmented and regularized deep learning for fine-grained image classification. In *CVPR*, pages 2645–2654, 2015.

[65] S. Yang and D. Ramanan. Multi-scale recognition with dag-cnns. In *ICCV*, pages 1215–1223, 2015.

[66] B. Yao, X. Jiang, A. Khosla, A. Lin, L. Guibas, and L. Fei-Fei. Human action recognition by learning bases of action attributes and parts. In *ICCV*, 2011.

[67] M. D. Zeiler. Adadelta: An adaptive learning rate method. *arXiv,* 1212.5701, 2012.

[68] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, pages 818–833, 2014.

[69] M. D. Zeiler, G. W. Taylor, and R. Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *ICCV*, pages 2018–2025, 2011.

[70] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *CVPR*, pages 2921–2929, 2016.

[71] B. Zhou, A. Khosla, A. Lapedriza, A. Torralba, and A. Oliva. Places: An image database for deep scene understanding. *arXiv,* 1610.02055, 2016.