Pattern Recognition 91 (2019) 34–46

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Deep convolutional network with locality and sparsity constraints for texture classification

Xingyuan Bu, Yuwei Wu*, Zhi Gao, Yunde Jia

Beijing Laboratory of Intelligent Information Technology, School of Computer Science, Beijing Institute of Technology (BIT), Beijing, 100081, China

ARTICLE INFO

Article history:
Received 19 April 2017
Revised 2 November 2018
Accepted 6 February 2019
Available online 16 February 2019

Keywords:
Deep convolutional feature
Sparse coding
Locality-aware
Texture classification

ABSTRACT

Recent studies have demonstrated advantages of the representations learned by Convolutional Neural Networks (CNNs) in providing an appealing paradigm for visual classification tasks. Most existing methods adopt activations from the last fully connected layer as the image representation. This paper advocates exploiting appropriately convolutional layer activations to constitute a powerful descriptor for texture classification under an end-to-end learning framework. The main component of our method is a new locality-aware coding layer conducted with the locality constraint, where the dictionary and the encoding representation are learned simultaneously. The layer is readily amenable to training via the backpropagation as the locality-aware coding process has an analytical solution. It is capable of capturing class-specific information which makes the learned convolutional features more robust. The resulting representation is particularly useful for texture classification. Comprehensive experiments on the DTD, FMD and KTH-T2b datasets show that our approach notably outperforms the state-of-the-art methods.

1. Introduction

Texture, as a visual cue describing the characteristic of many types of images, plays a significant role in analysis of the visual content from images. Texture classification is one of the fundamental problems in the computer vision community and has been a long-standing research topic with a wide variety of applications including face recognition [1], image retrieval [2], medicine analysis [3], and material classification [4]. Although texture classification is similar to other classification tasks, it presents distinct challenges. For instance, texture representation often suffers from the variations in scale, illumination, rotation, and the subtle difference in different texture patterns [5–7]. In case of the bubbly texture, since the scale of bubbles forms the different geometry structure, the bubbles show a challenge for consistency global information at different scales. Therefore, the fundamental problem of texture classification is to build a robust local representations and pool them in an orderless manner for the regular repetition texture [8].

To design a robust image representation which encodes the underlying characteristic texture structure, researchers have developed a number of discriminative and robust texture features, such as filter bank texton [9,10], image patch [4,11,12], and Local Binary Pattern (LBP) [13–17]. The recent years have witnessed significant advances of convolutional neural networks (CNNs). Driven by the emergence of large-scale data sets and fast development of computation power, CNNs have proven to perform remarkably well on a wide range of visual recognition tasks including texture classification [18–25]. Typical CNNs are often concatenations of multiple convolution layers followed by a couple of fully connected layers and a SoftMax classifier. It has been demonstrated that the last fully connected (FC) layer features can be employed as an universal image descriptor and suitable for linear classifiers such as SVM [26]. Different from hand-crafted features, the FC features learned by CNNs often possess rich high-level semantic information that can effectively distinguish the object of interest from the background. However, local characteristics of images may not well preserve at the FC layer, because spatial information is ignored when the fully connected layer transforms the convolutional layer activations into a feature vector representing the whole image. Moreover, the size of the input image should be fixed to be compatible with the FC layer, which makes the deep learning framework potentially less flexible. Additionally, texture is generally defined as a set of regularly repeating pattern elements along a plane [7]. It means that the FC feature, as a global descriptor, may not be suitable for modeling an orderless representation of the texture image [27,28].

Recently, the research trend has moved towards the features extracted from deep convolutional layers of CNNs which can be interpreted as local features describing particular image regions [28–32]. Two contemporaneous works introduced by Liu

* Corresponding author.
E-mail address: wuyuwei@bit.edu.cn (Y. Wu).

https://doi.org/10.1016/j.patcog.2019.02.003
0031-3203/

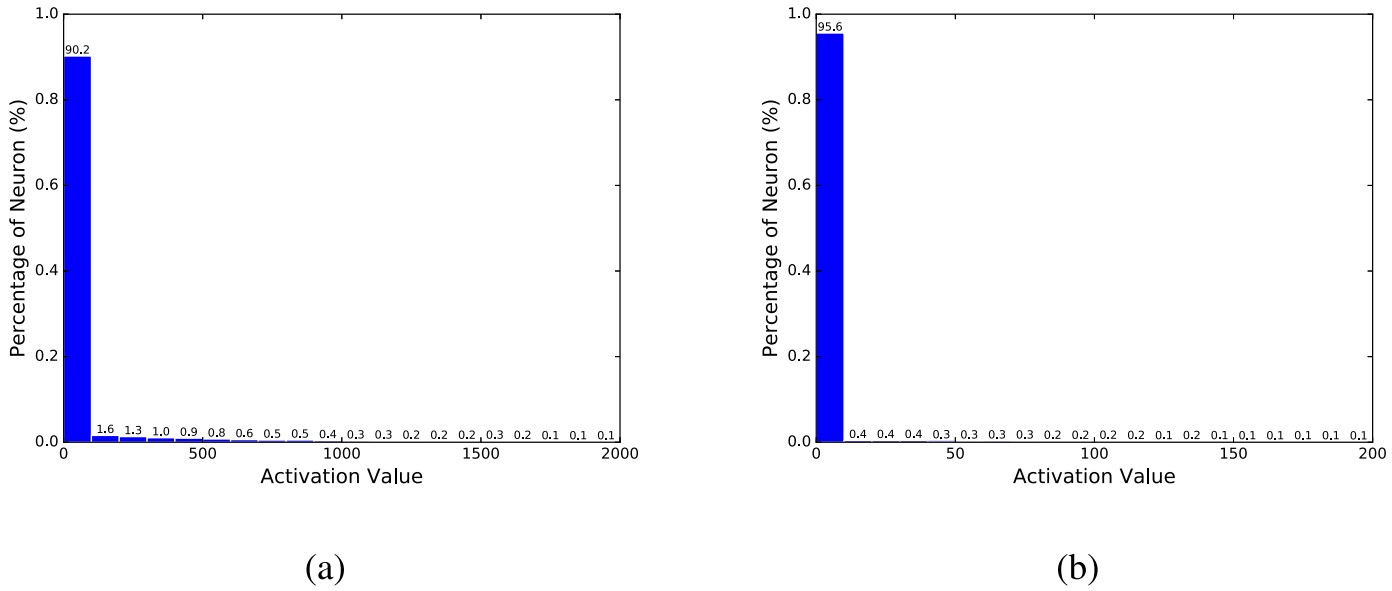

Fig. 1. The histograms of activation values in VGG-16 conv4_3 layer (a) and conv5_3 (b) (best viewed on high-resolution display).

et al. [33] and Babenko and Lempitsky [29] have demonstrated that the activations from the convolutional layers could be seen as a set of local features which can capture the visual representation related to objects. In addition, Long et al. [34] demonstrated that the output of earlier layer is actually preferable to obtain expressive region descriptors and inclined to encoding more informative details compared with the later layers. Therefore, the use of intermediate layer activations, i.e., deep convolutional features have recently achieved promising performance and attracted a lot of interests for visual classification. In this work, we focus on building a powerful image representation based on activations from convolutional layers with both the locality and sparsity constraints by using an end-to-end learning framework for texture classification.

Our motivation comes from the following observations. The *sparsity* has a great influence on good performance of CNNs [35,36] due to the rectifier linear unit (ReLU) activation functions and the Dropout operation [37]. As shown in Fig. 1, ReLU only keeps positive activations and switches off all the negative activations, resulting in most activations are small or zero [18]. Dropout randomly switches off activations in neural networks to shape a sparse network [37]. Co-adaptation disentangling, linear separability, and model combination are offered to explain the advantages of sparsity in CNNs [38–41]. *Locality* is another key property of the deep convolutional feature. The FC features from the same category are close to each other and far from the others, which shows the locality of convolutional features [42]. As illustrated in Fig. 2, we visualize the convolutional feature and the SIFT feature extracted from the FMD dataset by running the t-SNE algorithm [43]. The convolutional features presents semantic clustering, i.e., locality while the SIFT almost mixed up between different categories.

In particular, we introduce a locality-aware coding layer in CNNs with locality constraints, leveraged by the "Locality-constrained Linear Coding" [44,45] image representation. Our locality-aware coding acts as a pooling layer integrated on top of the convolutional layers and a class-specific dictionary, then generates a sparse locality-aware representation with respect to a distance constraint to improve the discriminative power of the convolutional feature. The distance constraint ensures that the encoded feature item has more weight when the corresponding dictionary atom is more similar to the original convolutional feature. Moreover, the coding layer learns an inherent dictionary and the encoding representation which enforces a class-specific tightness reconstruction based on the semantic clustering. As a consequence, the activation distribution for each encoded feature is highly peaked in each category. More importantly, the proposed locality-aware coding layer is differentiable and therefore can be solved by a closed-form solution. The gradient propagation is unblocked for the coding layer and the convolutional layer under an end-to-end framework. Overall, the layer is readily pluggable into any CNN architecture and amenable to training via standard backpropagation.

The rest of the paper is organized as follows: Section 2 introduces the related works about the CNN image representation and texture classification. Section 3 presents our network architecture including the locality-aware coding layer and the global representation layer in detail. Section 4 shows the experiment results on standard datasets and extensive diagnostic analysis. We conclude this work in Section 5.

2. Related work

In this work, we concentrate on building an orderless image representation using activations from convolutional layers for texture classification. We provide a literature review from two aspects, i.e., CNNs based image representations and texture classification.

2.1. Image representation using CNNs

Traditional methods typically obtain the image representation by aggregating the hand-crafted local features (e.g., SIFT) into a global image descriptor. Popular aggregation schemes include Bag of Words (BoW) [46], Vector of Locally Aggregated Descriptor (VLAD) [47] and Fisher Vector (FV) [48].

Recent works have witnessed significant advances of CNNs for a variety of classification tasks [21,23,30,49,50]. In the paradigm of CNNs, the most common way takes activations from the fully connected (FC) layer as an image representation, and applies this descriptor to visual classification. In order to obtain better performance, researchers focus on building deeper network [21–23], or changing the loss function and training strategy [51]. Babenko et al. [19] suggested that the feature emerging in the FC layer can serve as a global descriptor for image retrieval. Nevertheless, they only

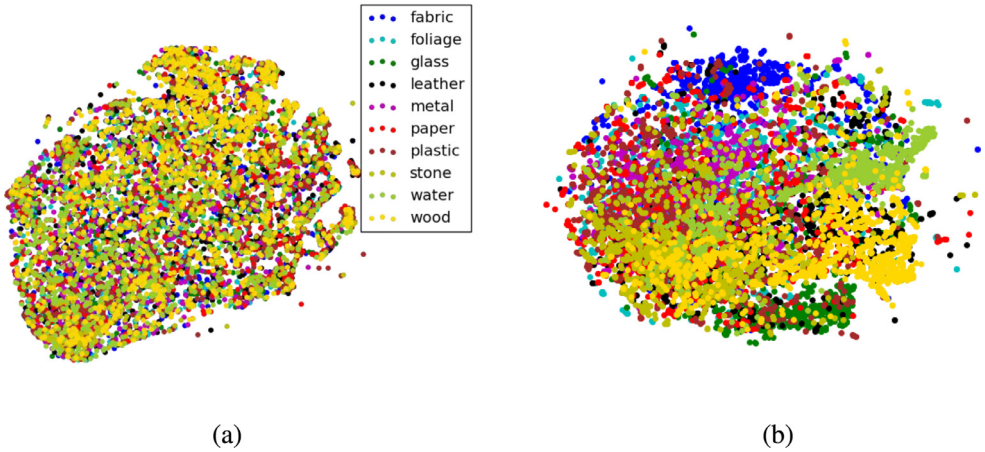

Fig. 2. Visualization of the convolutional feature and SIFT feature using the t-SNE algorithm. (a) shows the SIFT extracted from the FMD dataset. (b) depicts convolutional features. The SIFT almost mixed up in different category while the convolutional feature presents locality (best viewed on high-resolution display).

treated the CNN as a black-box feature extractor rather than studying the properties of CNN features from different layers. Gong et al. [52] introduced a multi-scale orderless pooling scheme to aggregate FC activations of local patches into a global feature using the VLAD. Similarly, Liu et al. [53] aggregated FC activations of local patches using Fisher Vector and sparse coding. One common factor in the above methods is that they all use the last FC layer features with the dimensionality of 4096 as global image descriptors and demonstrate their advantages over the traditional image descriptors. Although FC features trained towards assigning category labels to images is invariant to illumination and rotation to some extent, they suffer from the lack of details of local patterns.

Alternatively, CNNs can be used to extract local features which distribute in convolutional layers along the direction of channels. Zeiler et al. [54] showed that feature maps following the later convolutional layers encode both spatial and semantic information of the dominant attributes and semantic concepts. Joe et al. [31] used VLAD over the convolutional layer to obtain instance-level descriptors for fine-grained image retrieval. Ng et al. [55] aggregated convolutional layer activations using VLAD and achieved competitive performance. Tolias et al. [56] max pooled the activations of the last convolutional layer to represent each patch and achieved compelling performance for object retrieval. Those methods use off-the-shelf convolutional features but lose the important end-to-end training ability of CNNs, because their encoding components are not differentiable. In contrast, our method can jointly fine-tune all parameters in an end-to-end manner due to the differentiable encoding process, as validated in Section 3.

Liu et al. [32] built a powerful image representation using the activations from two consecutive convolutional layers. He et al. [57] introduced the Spatial Pyramid Pooling Network (SPP-Net) which solves the variable scales by spatial pyramid aggregation. Arandjelović et al. [50] developed a new generalized VLAD layer based on a weakly supervised ranking loss to learn parameters of the architecture in an end-to-end manner. These works [50,57] analyze the scheme of encoding features but lack of discussion about the convolutional feature itself. In this work, we not only encode the convolutional feature but also exploit the locality constraints within the convolutional feature via a locality-aware coding layer to build a discriminative image representation.

2.2. Texture classification

Texture classification is the basic problem in material recognition, biomedical image analysis, and content based image retrieval. It primarily consists of two critical subproblems: feature extraction and classifier designation [17]. An effective image representation is an essential element for texture classification, which encodes the underlying characteristic texture structure under variations in scale, viewpoint, and rotation. In the last decade, a number of discriminative and robust texture features have been proposed for texture classification. Traditional methods usually represent texture by pooling robust local features like filter bank texton [9,10], image patch [4,11,12], and Local Binary Pattern (LBP) [13–17]. In addition, the specialized texture descriptors [5,6,58] are also designed for texture classification under specific circumstances such as dynamic texture [59] and fine-grained material [5].

Driven by the emergence of large-scale data sets and fast development of computation power, features based on CNNs have been proven to outperform hand-crafted texture features. Gatys et al. [60] showed that the Gram matrix representation extracted from various layers of the VGG can be inverted for texture synthesis. Cimpoi et al. [8,28] proposed a so-called FV-CNN texture descriptor obtained by Fisher Vector pooling of a CNN filter bank response. They stated their FV-CNN is orderless because if we represent a pooling scheme as ϕ_p, and the set of local features as $\mathcal{F} = [f_1, f_2, \ldots, f_N]$, $f_i \in \mathbb{R}^K$, their final global feature $g = \phi_p(\mathcal{F})$ will remain unchanged even the order of $\mathcal{F}$ changes. They also announced the importance of orderless pooling for texture classification, that given a particular texture, appearance variations are statistically independent in the long range and therefore the ideal texture representations is the orderless pooled local descriptors. However, their FV-CNN only encodes off-the-shelf feature from the pre-trained CNN, i.e., it is not able to be trained in an end-to-end manner. Lin et al. [61] proposed a general orderless pooling architecture called the bilinear CNN that outperforms Fisher Vector since the gradients of the model can be easily computed allowing fine-tuning. They also conducted a systematic evaluation of recent CNN-based texture descriptors and explained the orderless feature captured by these descriptors [27]. Zhang et al. [62] presented a Deep-TEN network incorporating the dictionary learning and residual encoding into a single layer of CNNs, which learns the encoding parameters along with an inherent dictionary in a fully supervised manner. Different from the FV-CNN and Deep-TEN, our locality-aware coding layer is designed with locality constraints, where the dictionary and the encoding representation are all learned simultaneously.

3. Deep networks with sparsity and locality constraints

As mentioned in Section 1, the CNNs feature tends to perform sparsity and locality. These two characters are critical to good

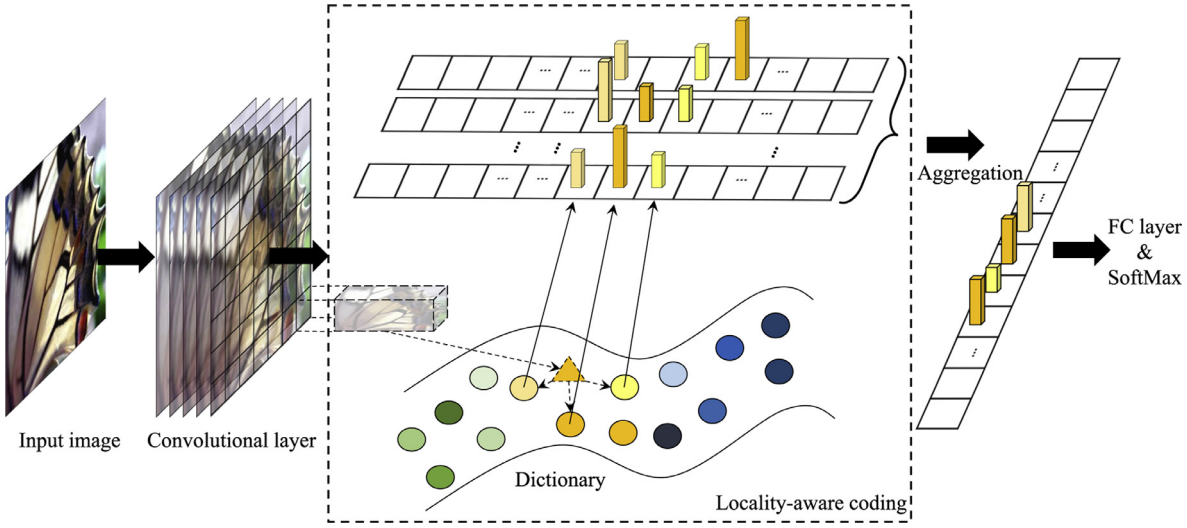

Fig. 3. The flowchart of our locality-aware coding in company with CNNs architecture. A standard CNN is firstly employed to extract the convolutional features, then these convolutional features are encoded respectively. The number of convolutional features is 7×7 using the last maxpooling layer of the VGG-16 network. The 7×7 features are encoded respectively with regard to a dictionary under the locality constraint that only the nearest atoms will be activated. At last, global maxpooling aggregates all encoded feature to produce a fixed-length feature followed by FC layer and loss function like SoftMax for texture classification.

performance of deep neural networks because the sparsity leads to co-adaptation disentangling, linear separability, and model combination, and the locality captures the essence of the category semantic distribution. However, deep neural networks learn these two characters by itself which is unreliable and unreasonable. In this work, we attempt to introduce a novel locality-aware coding layer to exploit the sparsity and locality, as illustrated in Fig. 3.

A locality-aware coding layer can be plugged into a standard CNN on top of the convolutional layers. In our approach, the dictionary learning and feature encoding are learned simultaneously in an end-to-end manner. During feature encoding, a locality constraint ensures that only the nearest atoms are activated, resulting in the encoded feature presents semantic clustering in a class-specific dictionary. Then a global maxpooling layer aggregates all encoded features for the followed FC layer and the classifier. Since the locality constraint is a ℓ_2-norm, the encoding process is differentiable. The locality-aware coding layer takes the gradients from the FC layer as input and then computes the gradients for the dictionary and the shallow convolutional layers. The gradients information is unblocked in the whole network architecture and thus the architecture is amenable to training via backpropagation.

3.1. The locality-aware coding layer

Let $X = \{x_i\}_{i=1}^N$ denote N convolutional features, where M is the dimensionality of a data point $x_i \in \mathbb{R}^{M\times 1}$. $D = [d_1, d_2, \ldots, d_K] \in \mathbb{R}^{M\times K}$ is a dictionary where each column represents an atom. $C = [c_1, c_2, \ldots, c_N] \in \mathbb{R}^{K\times N}$ is a coding matrix. The goal of sparse representation is to learn a dictionary and corresponding sparse codes such that each input local feature x_i can be well approximated by the dictionary D. The general formulation of the sparse representation is expressed as

$$\min_{c_i} \sum_{i=1}^{N} \|x_i - Dc_i\|_2 + \lambda \|c_i\|_1, \tag{1}$$

where $\|c_i\|_1$ is a ℓ_1 constraint term which enforces c_i to have a small number of nonzero elements. $\sum_{i=1}^N \|x_i - Dc_i\|_2$ measures the reconstruction error. λ is a weight parameter for making a trade-off between the sparsity and reconstruction error.

While a general sparse coding can be used to exploit the sparsity for CNNs, the locality of CNNs feature is not preserved during sparse coding operation. As discussed in [44,45], the locality implies the class-specific information when the dictionary is initialized under the category-wise K-means algorithm. Atoms in the dictionary near the input are more likely to be the same category than those far from the input. It is easy to prove that preserving the locality information during encoding will obtain a more discriminative feature. The locality constraint can be used to replace the sparsity constraint in Eq. (1), the locality-aware coding is formulated as

$$\begin{aligned} &\min_{c_i} \sum_{i=1}^{N} \|x_i - Dc_i\|_2 + \lambda \| p_i \odot c_i\|_2 \\ &\text{s.t.} \mathbf{1}^\top c_i = 1, \forall i, \end{aligned} \tag{2}$$

where $\mathbf{1}$ is a column vector of "1"s. The $p_i = [p_{i1}, p_{i2}, \ldots, p_{iK}] \in \mathbb{R}^K$ is the locality adaptor which adapts the encoded feature to the distance between input x_i and dictionary atoms $[d_1, d_2, \ldots, d_K]$. The locality adaptor p_{ik} is usually expected to be large when the input x_i is far from the dictionary atom d_k and be small when they are close to each other. So that the code c_i is affected by p_i through the element-wise multiplication $\odot$, then the locality constraint will force the code element c_{ik} to be zero if x_i is far from d_k. The shift-invariant constraint $\mathbf{1}^\top c_i = 1$ enforces the coding results to remain the same even if the origin of the data coordinate system is shifted, as proved in [44]. Locality-aware coding Eq. (2) would produce a sparse code c_i, since only a few dictionary atoms are near to the input x_i and have large weight, the rest of atoms are penalized and have small weights. We use ℓ_2-norm locality adaptor for a convenient purpose. It measures Euclidean distance between the input feature x_i and the kth dictionary atom d_k:

$$\begin{aligned} p_{ik} &= \|x_i - d_k\|_2 \\ &= (x_i - d_k)^\top (x_i - d_k). \end{aligned} \tag{3}$$

Hence, the diagonal matrix P_i whose principal diagonal elements are the entries of p_i can be expressed as

$$P_i = diag\left(\tilde{D}^\top \tilde{D}\right), \tag{4}$$

where $\tilde{D} = \left(x_i \mathbf{1}^\top - D\right)$.

Since the ℓ_0 or ℓ_1 norm in standard sparse coding are replaced by the locality constraint $\|p_i \odot c_i\|_2$, there is a closed-form solution

for Eq. (2), so that the network's forward transformation is

$$\begin{cases} \tilde{c}_i = \left(\tilde{D}^\top \tilde{D} + \lambda P_i^2\right)^{-1} \mathbf{1} \\ c_i = \dfrac{\tilde{c}_i}{\mathbf{1}^\top \tilde{c}_i} \end{cases}. \tag{5}$$

In order to carry out an end-to-end learning, we propose a locality-aware coding backpropagation algorithm. Suppose the loss function is $\mathcal{L}$, and given the gradient input $\frac{\partial \mathcal{L}}{\partial c_i}$, we firstly compute the partial derivatives of $\mathcal{L}$ with respect to $\tilde{c}_i$:

$$\frac{\partial \mathcal{L}}{\partial \tilde{c}_i} = A^\top \frac{\partial \mathcal{L}}{\partial c_i}, \tag{6}$$

where A is the derivative of c_i with respect to $\tilde{c}_i$, given by

$$A = \begin{bmatrix} \frac{dc_1}{d\tilde{c}_1} & \frac{dc_1}{d\tilde{c}_2} & \cdots & \frac{dc_1}{d\tilde{c}_K} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{dc_K}{d\tilde{c}_1} & \frac{dc_K}{d\tilde{c}_2} & \cdots & \frac{dc_K}{d\tilde{c}_K} \end{bmatrix}.$$

Each element in A is

$$\frac{dc_i}{d\tilde{c}_j} = \frac{\delta_{ij}\left(\mathbf{1}^T\tilde{c}\right) - \tilde{c}_i}{\left(\mathbf{1}^T\tilde{c}\right)^2},$$

where δ_{ij} is the indicator function:

$$\delta_{ij} = \begin{cases} 1, & if \quad i = j \\ 0, & if \quad i \neq j \end{cases}.$$

Then the gradients of $\mathcal{L}$ with respect to x_i and D are given by

$$\frac{\partial \mathcal{L}}{\partial x_i} = -2\tilde{D}(G + \lambda diag(PG + GP))\mathbf{1}, \tag{7}$$

$$\frac{\partial \mathcal{L}}{\partial D} = 2\tilde{D}(G + \lambda diag(PG + GP)), \tag{8}$$

where G is

$$G = \left(\tilde{D}^\top \tilde{D} + \lambda P_i^2\right)^{-1} \mathbf{1} \left(\frac{\partial \mathcal{L}}{\partial \tilde{c}_i}\right)^\top \left(\tilde{D}^\top \tilde{D} + \lambda P_i^2\right)^{-1}.$$

Once all $\frac{\partial \mathcal{L}}{\partial x_i}$ and $\frac{\partial \mathcal{L}}{\partial D}$ are obtained, the standard backpropagation can be easily employed to update the CNNs and the dictionary parameters.

3.2. Fast version of locality-aware coding

In the neural network, what the forward and backward operations need are the feature outputs and the gradient outputs for inputs and parameters. In our network, they are obtained by solving a linear system, as demonstrated as Eqs. (5), (7), (8), whose complexity depends on the number of the atoms K, namely $O(K^3)$. When K goes larger, the forward and backward operations will be time-consuming. Actually, the encoded feature of Eq. (5) is supposed to have limited significant values since only a few dictionary atoms are near to the input x_i and have large weights. We may only keep these significant values and set the other trivial values to zeros, hence get a much smaller linear system to reduce our computational complexity.

In practice, we can apply the k-NN method to choose the nearest atoms in the dictionary D for an input x_i, then the chosen atoms are retreated as a new dictionary, thus Eq. (2) can be reformulated as

$$\min_{c_i} \sum_{i=1}^{N} \|x_i - D_i c_i\|_2$$

$$\text{s.t.} \quad \mathbf{1}^\top c_i = 1, \forall i, \tag{9}$$

where $D_i \in \mathbb{R}^{M \times L}$ is the new dictionary containing L atoms chosen by the k-NN method. This local linear system can be solved by

$$\begin{cases} \tilde{c}_i = \left(\tilde{D}_i^\top \tilde{D}_i\right)^{-1} \mathbf{1} \\ c_i = \frac{\tilde{c}_i}{\mathbf{1}^\top \tilde{c}_i} \end{cases}, \tag{10}$$

where $\tilde{D}_i = \left(x_i \mathbf{1}^\top - D_i\right)$.

The subsampling from D (in Eq. (2)) to D_i (in Eq. (9)) might cause discontinuous derivative. To address this issue, we borrow the idea of maxpooling, in which the discontinuous derivative exists. We only keep the L max atoms. The maxpooling computes output $y_i = x_{pos}$, in which i is the output position, $R(i)$ is the position set of inputs in the sub-window over y_i, and

$$pos = argmax_{pos' \in R(i)} x_{pos'}.$$

The derivative of maxpooling with respect to the jth position x_j is

$$\frac{\partial y_i}{\partial x_j} = \begin{cases} 1, & if \quad j = pos \\ 0, & otherwise \end{cases}.$$

Similar to maxpooling, our approximate algorithm subsampling D to D_i can be expressed as

$$D_i = \{atom \mid atom = argmax_{atom' \in D} L\ \tilde{c}_{i,atom'}\},$$

where $argmaxL$ means that it takes top L rather than top 1, $\tilde{c}_{i,atom'}$ is the element in $\tilde{c}_i$ corresponding to $atom'$. The derivative $\frac{\partial \mathcal{L}}{\partial x_i}$ and $\frac{\partial \mathcal{L}}{\partial D_i}$ in Eq. (11) can be obtained using D_i

$$\begin{cases} \dfrac{\partial \mathcal{L}}{\partial x_i} = -2\tilde{D}_i H \mathbf{1} \\ \dfrac{\partial \mathcal{L}}{\partial D_i} = 2\tilde{D}_i H \end{cases}, \tag{11}$$

where H is

$$H = \left(\tilde{D}_i^\top \tilde{D}_i\right)^{-1} \mathbf{1} \left(\frac{\partial \mathcal{L}}{\partial \tilde{c}_i}\right)^\top \left(\tilde{D}_i^\top \tilde{D}_i\right)^{-1}.$$

Moreover, we can obtain the derivative of $\frac{\partial \mathcal{L}}{\partial D}$ for standard backpropagation in an atom-by-atom way as

$$\frac{\partial \mathcal{L}}{\partial D_{atom}} = \begin{cases} \frac{\partial \mathcal{L}}{\partial D_{i,atom}}, & if \quad atom \in D_i \\ 0, & otherwise \end{cases}, \tag{12}$$

where $atom$ is an atom in the dictionary.

With Eqs. (11) and (12), our approximate algorithm can be optimized using standard backpropagation. An remarkable advantage of our approximate algorithm is that it reduces the computational complexity from $O(K^3)$ to $O(K \times M + L^3)$. Since $L \ll K$, our network has the ability to be implemented in a fast way. If we measure the computational complexity via FLOPs, there is little increase from VGG $1.55E + 10$ FLOPs to our $1.56E + 10$ FLOPs, as shown in Table 1.

3.3. Global representation

Once all convolutional features encoded, default CNNs usually vectorize all convolutional features by concatenating them to bridge the different shape of the convolutional layer and the generalized linear classifier (FC layer), which has several disadvantages. First, the spatial structures in the convolutional layer are destroyed. The activations distributing in a three dimensional space are vectorized into a one dimensional vector regardless the activations position in the feature map and channel [63]. Second, the direct vectorization produces a large number of FC units, which makes CNNs substantial overfitting [18]. Third, the vectorization forces the input

Table 1
The FLOPs of our architecture. The "I", "O" and "K" represent "Input", "Output" and "Kernel" respectively. The subscript "W", "H" and "C" represent "Width", "Height" and "Channel", respectively. The main difference between our model and VGG-16 exists after the fifth maxpooling layer.

Layer	I_W	I_H	I_C	O_C	K_W	K_H	FLOPs
conv1_1	224	224	3	64	3	3	8.67E+07
conv1_2	224	224	64	64	3	3	1.85E+09
maxpooling	224	224	64	64	2	2	3.21E+06
conv2_1	112	112	64	128	3	3	9.25E+08
conv2_2	112	112	128	128	3	3	1.85E+09
maxpooling	112	112	128	128	2	2	1.61E+06
conv3_1	56	56	128	256	3	3	9.25E+08
conv3_2	56	56	256	256	3	3	1.85E+09
conv3_3	56	56	256	256	3	3	1.85E+09
maxpooling	56	56	256	256	2	2	8.03E+05
conv4_1	28	28	256	512	3	3	9.25E+08
conv4_2	28	28	512	512	3	3	1.85E+09
conv4_3	28	28	512	512	3	3	1.85E+09
maxpooling	28	28	512	512	2	2	4.01E+05
conv5_1	14	14	512	512	3	3	4.62E+08
conv5_2	14	14	512	512	3	3	4.62E+08
conv5_3	14	14	512	512	3	3	4.62E+08
maxpooling	14	14	512	512	2	2	1.00E+05
Locality aware coding	7	7	512	2048			2.06E+08
maxpooling	7	7	2048	2048	7	7	1.00E+05
FC6	1	1	2048	512	1	1	1.05E+06
FC7	1	1	512	47	1	1	2.41E+04
Total							**1.56E+10**

image to be fixed-size due to fixed-length FC layer. It is not natural for different scale images [57]. Fourth, as discussed in Introduction, the FC feature may not be suitable to model an orderless representation for the texture classification.

In order to overcome above shortcomings, we employ a new aggregation scheme, called global maxpooling, to replace traditional vectorization. The global maxpooling takes the max value of each channel instead of vectorizing convolutional layer. This aggregation scheme presents several attractive advantages:

- As we encode the convolutional feature with regard to a dictionary, the global maxpooling actually computes statistics over all activations within a channel, i.e., an atom. Hence, each atom is directly corresponding to the FC layer and further corresponding to the classification task in the later SoftMax layer. Global maxpooling is more meaningful and interpretable than the vectorization operation.
- As global maxpooling generates FC layer by taking the max values along with channels, the aggregation scheme is relevant to the number of atoms K instead of image size. We can input an image with any scale, which is not necessary to adapt to the deep network.
- Since the number of parameters is reduced, global maxpooling is able to mitigate the overfitting, which improves the networks generalization ability.
- Global maxpooling can build an orderless representation for the texture classification. We denote the set of local features as $\mathcal{F} = [f_1, f_2, \ldots, f_N]$, $f_i \in \mathbb{R}^K$. The ith element in our final global feature g can be obtained by $g_i = \max\left(f_{1,i}, f_{2,i}, \ldots, f_{N,i}\right)$, where $f_{j,i}$ is the ith element in the jth local features. We can see that global maxpooling is invariant to the permutation of input, that is, is orderless.

Clearly, the locality-aware coding and global maxpooling are trainable and can be optimized by the standard backpropagation. First, an image is fed into the CNNs to extract the convolutional feature. In order to implement the efficient forward and backward propagation, a fast version of locality-aware coding has been proposed, in which the convolutional feature is encoded in a differentiable process with regard to a class-specific dictionary. Finally, all encoded features are aggregated by global maxpooling for the classification task.

4. Experiments

4.1. Datasets and evaluation

We evaluate our method on three texture datasets, i.e., Describable Textures Dataset (DTD) [64], Flickr Material Dataset (FMD) [65] and KTH-TIPS-2b (KTH-T2b) [66]. Both DTD and FMD are collected in uncontrolled conditions, which are so-called "in the wild". Since the DTD assigns the label by psychological literature, its labels are not the category of the material or object but the feeling of human about an image. FMD's labels are the category of material, but the material isn't cropped from the background. KTH-T2b's images are collected under controlled pose and illumination with multi-scale. Fig. 4 illustrates these three texture datasets for our experiments. For all datasets, we follow the standard train-test protocol. On the DTD and FMD datasets, we randomly and evenly divide the dataset into train, valid, and test set for 10 splits. We use the train and valid set for training and report the Mean Accuracy on the test set across splits. On the KTH-T2b dataset, we train the model on one sample and test on the remaining three.

4.2. Implementation details

Our locality-aware coding based on the convolutional feature is from the VGG-16 [21] which is pre-trained on the ImageNet. We crop the VGG-16 at the last maxpooling layer to extract the convolutional features and insert our locality-aware coding layer and maxpooling layer following by two fully connected layers whose output dimension is 512 and the number of classes, respectively. Dropout is used on the first fully connected layer. Unless otherwise stated, we use the fast version of locality-aware coding with $K = 1024$ and $L = 5$ in our experiments. The dictionary is initialized by concatenating the results of K-means algorithm in different categories. We use SGD with a mini-batch size of 128. The

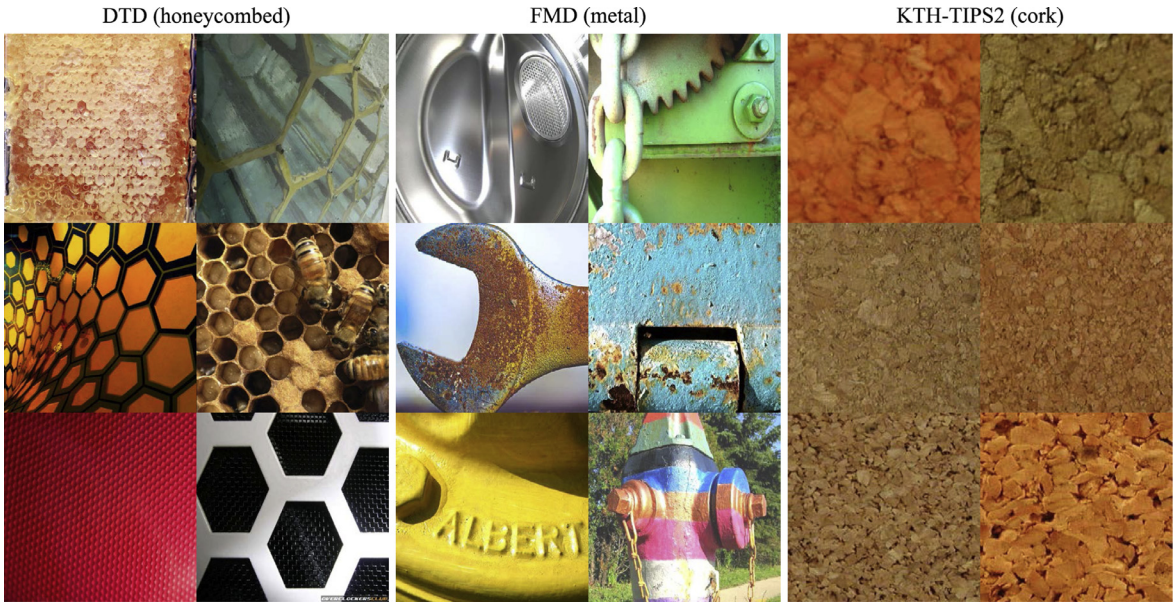

Fig. 4. The sample images of the DTD, FMD, and KTH-TIPS-2b datasets. The DTD texture dataset which is collected in the wild focuses on the prior to high-level semantic understanding rather than material. The FMD dataset focuses on the material without alignment and crop, so that the material is bothered by background. The KTH-TIPS-2b dataset addresses the problem of category-level material classification under the varying scale.

learning rate is started from 0.01 and divided by 10 when error plateaus. All the texture images are normalized by subtracting 123, 117, 104 for RGB channels, respectively. Following the standard method in [18], the images are augmented by randomly cropping 15 times for training and picking from the center and four corners for testing. Our network is running based on Torch on an Inter(R) i7-5930k CPU 3.5GHz, NVIDIA TitanX GPU and 32GB RAM.

4.3. Comparisons with state-of-the-art methods

To assess benefits of our method, we compare our method trained for texture classification against the other state-of-the-art approaches. Namely, we compare with previous methods from hand-crafted texture descriptor to global FC feature to the convolutional feature. In Table 2, the comparison with the previous state-of-the-art methods is illustrated.

LM [9] and MR8 [10] treat the clustered filter bank response as the texton, and then learn an image representation by distributions on a texton dictionary. Later, Varma et al. [4,11] used the raw image patch around a point instead of the filter bank response. We evaluate 3×3 and 7×7 patch size in our experiment. As for LBPs feature, two variants are compared here, including LBP_{riu2} [13] and CLBP [14]. LBP_{riu2} considers circular symmetric and preserves only those frequent patterns, and CLBP combines multiple LBP type features via joint histogramming. From Table 2, we can see the CNN-based approaches outperform all head-crafted approaches.

The Deep Convolutional Activation Feature (DeCAF) [42] is the top output of AlexNet [18], and the FC-CNN [64] is the top output of VGG-16 [21]. Thus they are both the off-the-shelf feature from the last FC layer with 4096 dimension. Clearly, the DeCAF and FC-CNN, as discussed in introduction, are both unsuitable for texture classification.

The FV-CNN [28] uses Fisher Vector to aggregate the convolutional feature extracted from VGG-16. 65600 dimension FV feature is produced by 64 Gaussian components and 512 dimension convolutional feature, then compressed to 4096 by PCA. The convolutional feature is suitable for texture classification because it can capture the orderless representation. This feature aggregation scheme makes the FV-CNN perform significantly better than the DeCAF and FC-CNN in all three datasets. The

Table 2
Comparisons with state-of-the-art methods in terms of Mean Accuracy (%). Hand-crafted features include texton-based methods (LM, MR8) and patch-based methods. FV-CNN, B-CNN and Deep-TEN using the aggregated convolutional feature outperform DeCAF and FC-CNN which use the FC feature. Our method encodes the convolutional feature to exploit the sparsity and locality with a dictionary in which atoms $K = 1024$, $K = 2048$ and neighbors $L = 5$.

method	DTD	FMD	KTH-T2b
Hand-crafted features			
LM [9]	18.8	18.2	53.5
MR8 [10]	15.9	22.1	53.0
$\text{Patch}_{3\times3}$ [11]	14.6	20.9	57.8
$\text{Patch}_{7\times7}$ [11]	18.0	21.2	58.4
LBP_{riu2} [13]	37.1	–	62.7
CLBP [14]	42.6	43.6	64.2
FC features			
DeCAF [42]	54.8	60.7	70.7
FC-CNN [64]	59.5	69.3	71.1
Convolutional features			
FV-CNN [28]	**67.3**	**73.5**	**73.3**
FV-FC-CNN [28]	70.1	76.4	73.8
B-CNN [27]	70.2	78.5	75.8
$\text{Deep-TEN}_{ResNet50}$ [62]	–	80.2	**82.0**
$\text{VGG-16}_{baseline}$ [21]	66.0	78.2	75.5
$\text{Ours}_{1024,5}$	70.6	80.6	77.2
$\text{Ours}_{2048,5}$	**71.1**	**82.4**	76.9

FV-FC-CNN [28] combining the FC feature's and the convolutional feature's Fisher Vector is roughly equivalent to all above methods. The B-CNN [27] feeds the convolutional feature to bilinear pooling which takes the location-wise outer product of the convolutional feature and sum pooling across all locations. Bilinear pooling feature slightly improves the performance in all cases. The Deep-TEN [62] employs a VLAD-like pooling layer to aggregate convolutional feature. Since Deep-TEN uses ResNet [23] with 50 layers while the other methods use VGG with 16 layers and Deep-TEN ignores the standard train-test protocol in KTH-T2b and FMD, it is not competitive compared with the other methods. Our method outperforms Deep-TEN in the FMD dataset. The methods mentioned above improve the benchmark and verify the convolutional feature's robustness and efficiency, but they just

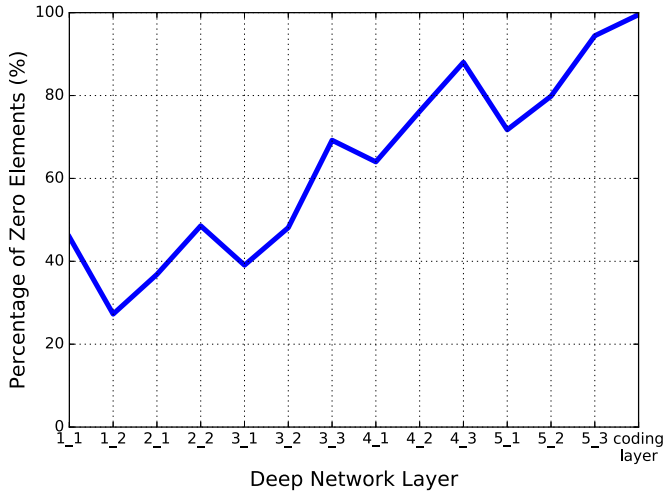

Fig. 5. The percentage of zero elements increases when an image passes through our deep network. The deep convolutional neural network tends to be sparse along with the depth. The adventive decrease in conv3_1, conv4_1 and conv5_1 is the result of maxpooling.

use the off-the-shelf convolutional feature by aggregating, without exploiting the character of CNNs.

The VGG-16 is the original network in [21] and trained in the texture datasets with standard train-test protocol. Interestingly, the VGG-16 as a common network for classification is roughly equivalent to the FV-CNN-like special-domain methods. To be more specific, we consider the key of the learned good representation is the sparsity and locality as discussed and showed in introduction and the next section. Our method not only utilizes the convolutional feature like FV-CNN and B-CNN, but also encodes the convolutional feature to exploit the sparsity and locality like pure VGG-16 in a trainable way, thus our method achieves the best performance with the dictionary where atoms $K = 1024$, $K = 2048$ and neighbors $L = 5$.

4.4. Diagnostic analysis

In this section, we analyze the effectiveness of sparsity and locality, the number of atoms and neighbors, and the effectiveness of global representation.

4.4.1. The effectiveness of sparsity and locality

The sparsity and locality are the key to the good performance of CNNs. In this paper, we use locality-aware coding to explore the sparsity and locality to build a robust network architecture and discriminative representation. In this section, qualitative experiments will be illustrated to give an intuitionistic and visual result.

As shown in Fig. 5, the deep convolutional neural network tends to be sparse along with the depth. At the beginning, there are no zero elements in the inputs of the conv1_1, but about half of the outputs are negative and will be switched off by ReLU (conv1_1: 45.86%). After the initialization in the conv1_2, the percentage of zero elements increasingly rises layer by layer (conv1_2: 27.26%, conv2_2: 48.55%, conv3_3: 69.21%, conv4_3: 88.02%, conv5_3: 94.43%). The adventive decrease in the conv3_1, conv4_1 and conv5_1 layer is the result of maxpooling between two stages of VGG-16. Maxpooling keeps the max values and drops the smaller one, causing many zero elements dropped, even though ReLU tends to produce a sparse deep architecture.

The sparsity is a valuable property brought by ReLU for deep networks. The sparsity makes deep network be a more robust architecture with co-adaptation disentangling and model combination. The co-adaptation disentangling ensures invariance for small changes of the input. The model combination shapes the deep neural model as an ensemble classifier of exponential linear models.

Our locality-aware coding enhances these advantages by reconstructing the input feature using a dictionary. Only closest atoms will be selected as the neighbors dictionary so that the input features are disentangling from the other neighbors dictionary. As shown in Fig. 5, locality-aware coding products more sparse output (L/K) for model combination and build an ensemble linear classifier. However, the network only with the sparsity is still hard to obtain a discriminative feature. The locality is more essential for discriminative than sparsity due to the semantic clustering.

Locality-aware coding using a class-specific dictionary further implements the locality. A feature will be represented by the atoms near to itself, as the dictionary is generated by concatenating the result of a class-specific K-means algorithm. Locality-aware coding enforces a tightness reconstruction based on the semantic clustering. Reasonably, the activation distribution in an encoded feature is highly peaked in one class. This leads to discriminative representation over learned CNNs.

As depicted in Fig. 6, we visualize the histogram generated by the fabric and wood category from the FMD dataset. The fabric and wood corresponding atoms locate in the first and the last one tenth of our dictionary, respectively. The average activation values of the conv5_3 and locality-aware feature are illustrated. It can be seen that the elements of locality-aware feature are very peaked at the corresponding category atoms and approaching zero elsewhere, as shown in Fig. 6(b) and (d), which forms a good approximation to the ideal discriminative representation. However, the conv5_3 feature can't present such class-specific representation like locality-aware feature, as shown in Fig. 6(a) and (c).

4.4.2. The number of atoms and neighbors

In this section, we investigate the number of the atoms K and the neighbors L on the FMD dataset, to give a discussion of the performance of the number selection. As illustrated in Fig. 7, it is noticed that the number of neighbors L is important. Some trade-off should be considered between reconstruction and locality: small L is hard to reconstruct the original convolutional feature, but large L disturbs locality-aware coding by more irrelevant atoms. In practice, we found that 5 nearest neighbors are the promising configuration when $K = 1024$ or $K = 2048$.

We also experiment with three dictionaries which contain 1024, 2048 and 4096 atoms, respectively. It can be seen that more atoms won't always lead better performance, because the outliers might be captured as atoms, blurring the bounds between different categories. In our experiment, 2048 atoms are most suitable for the FMD dataset, more atoms might be better in a larger dataset.

4.4.3. The effectiveness of global representation

The global maxpooling not only aggregates the encoded feature, also prevents locality-aware coding from overfitting due to dimensionality reduction. To investigate the effect of global maxpooling, we evaluate the models with and without global maxpooling for both locality-aware feature and VGG-16 feature on the FMD dataset.

Table 3 compares four models: VGG-16, VGG-16$_{WithGloMax}$, Ours$_{WithoutGloMax}$ and Ours$_{WithGloMax}$. VGG-16$_{WithGloMax}$ is the standard VGG-16 model but using global maxpooling to replace the traditional vectorization. Ours$_{WithGloMax}$ is our default configuration used in our experiments where the encoded features are aggregated by global maxpooling, while Ours$_{WithoutGloMax}$ does not employ the global maxpooling. Four models are pre-trained on the FMD dataset.

In our method, only closest atoms will be chosen to represent the input. Thus the activated atoms have a large probability to locate in the corresponding category region of the class-specific

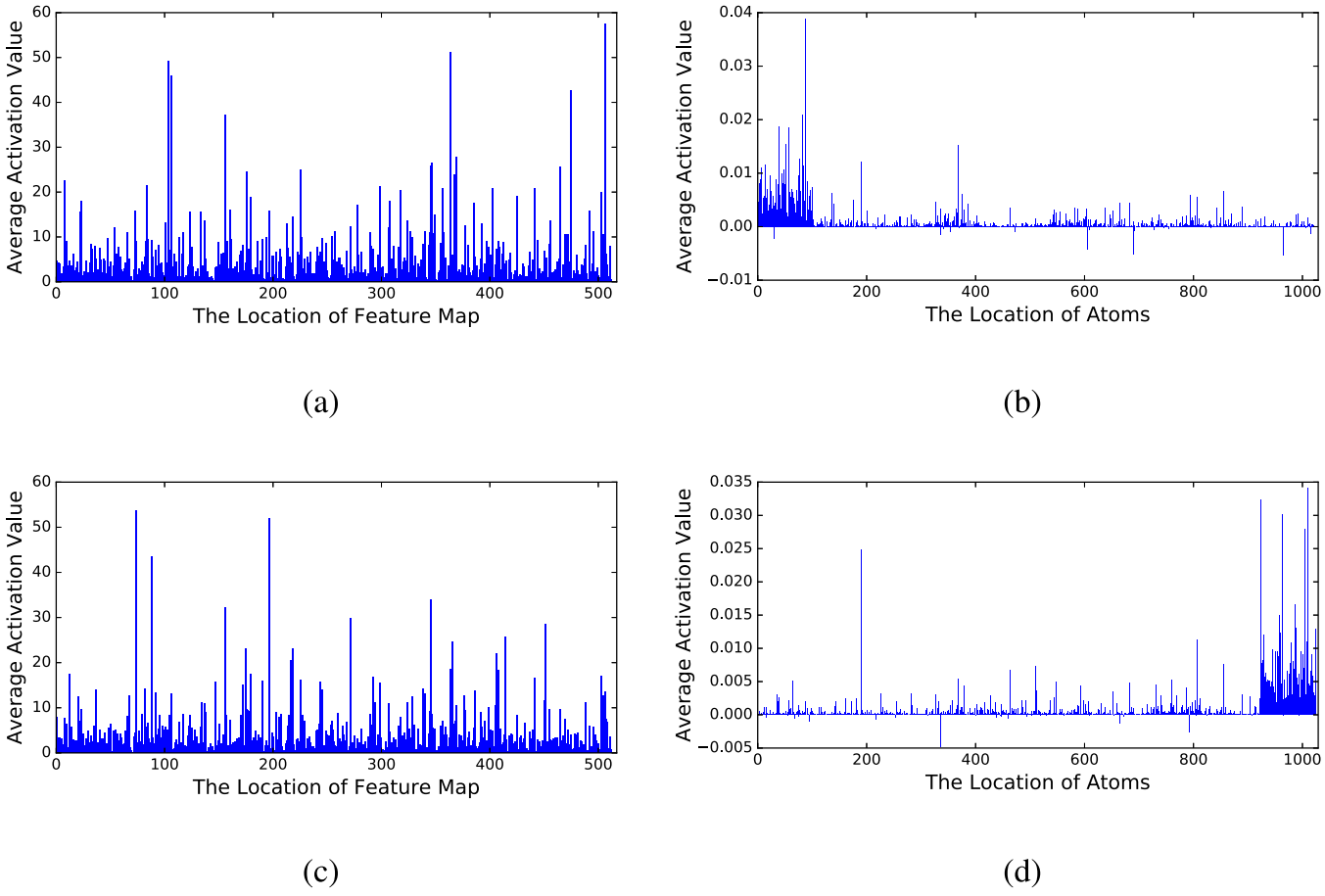

Fig. 6. Average activation values in FMD dataset. (a) Average of conv5_3 for fabric category, (b) Average of locality-aware feature for fabric category, (c) Average of conv5_3 for wood category, (d) Average of locality-aware feature for wood category. (Note that since each layer is sparse, there is bias between real magnitude of activation values and that can be witnessed here.)

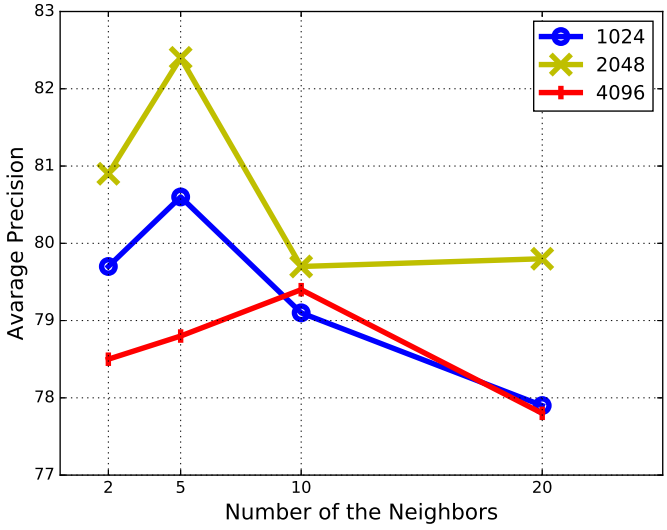

Fig. 7. The performance of the different number of atoms and neighbors.

dictionary. Although locality-aware feature (75.7%) is more class-specific than VGG-16 (78.2%), it still suffers from overfitting due to the high dimensionality in $\text{Ours}_{WithoutGloMax}$.

Global maxpooling aggregates all activations within a channel to reduce the parameters in the followed FC layer, hence global maxpooling improves our locality-aware coding generalization ability (75.7% → 80.6%) by reducing the overfitting and building a connection between atoms and classifier. Moreover, a single locality-aware feature's activations mainly distribute in the corresponding semantic region, thus taking the max values of the locality-aware feature in a channel won't harm this semantic region, but make a more conclusive distribution. Fig. 6(b) and (d) intuitively explain this point that global aggregation improves the semantic clustering of the category region.

On the contrary, global maxpooling doesn't help in VGG-16, even harms the performance (78.2% → 77.1%). We conclude that, global maxpooling computes the statistics within a channel, but the convolutional features don't take the locality into account, leading a confused distribution in all region of channels. Fig. 6(a) and (c) explain this point to some extent.

The effect of global maxpooling for semantic locality is illustrated in Fig. 8. We pick up locality-aware feature (Fig. 8(a)) from the model $\text{Ours}_{WithoutGloMax}$ and VGG-16 feature (Fig. 8(c)) from the

Table 3
Comparison for the effect of global maxpooling. $\text{VGG-16}_{WithGloMax}$ is the standard VGG-16 model but replaces the vectorization with global maxpooling. $\text{Ours}_{WithGloMax}$ is our default model using global maxpooling to aggregate the local feature, while $\text{Ours}_{WithoutGloMax}$ does not utilize the global maxpooling.

Network model	Mean accuracy (%)
VGG-16	78.2
$\text{VGG-16}_{WithGloMax}$	77.1
$\text{Ours}_{WithoutGloMax}$	75.7
$\text{Ours}_{WithGloMax}$	80.6

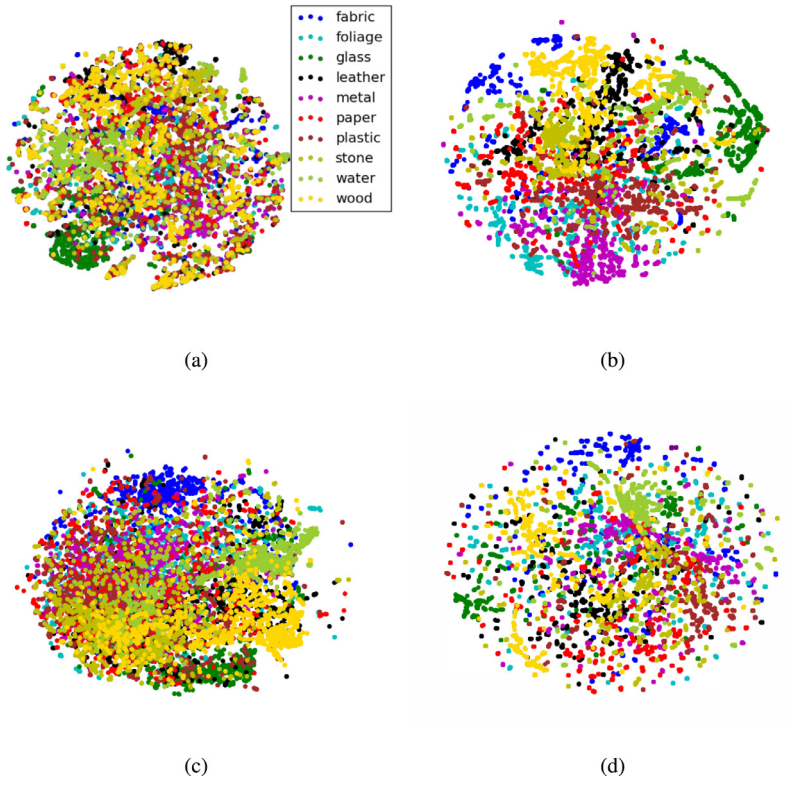

Fig. 8. Feature visualization on the FMD dataset. We visualize four kinds of features: (a) The locality-aware feature before vectorizing from the model Ours$_{WithoutGloMax}$. (b) The locality-aware feature after global maxpooling from the model Ours$_{WithGloMax}$. (c) The VGG-16 feature before vectorizing from the model VGG-16. (d) The VGG-16 feature after global maxpooling from the model VGG-16$_{WithGloMax}$. By comparing (a) with (b), we can see that the global maxpooling increases the semantic clustering for locality-aware feature. While there is no improvement in the semantic performance of VGG-16 in (c) and (d). For all four subfigures, we pick 10,000 samples and use the t-SNE algorithm to visualize them (best viewed in color).

model VGG-16 before vectorized, and pick up locality-aware feature (Fig. 8(b)) from the model Ours$_{WithGloMax}$ and VGG-16 feature (Fig. 8(d)) from the model VGG-16$_{WithGloMax}$ after global maxpooling. Four kinds of features are picked on the FMD dataset, each of them contains 10,000 samples. Then we visualize these four kinds of features by running t-SNE algorithm [43] to find a 2-dimensional embedding of the high-dimensional feature, and color them with regard to category. Four subfigures in Fig. 8 show the semantic clustering. By comparing Fig. 8(a) with (b), we can see that the global maxpooling increases the semantic clustering for locality-aware feature. The distribution is more tightness and discriminative. On the contrary, as shown in Fig. 8(c) and (d), there is no improvement for global maxpooling in the semantic performance of VGG-16.

4.4.4. The impact of backbones

The proposed locality-aware coding layer can be built on any CNN base model, e.g., ResNet18, ResNet50 and VGG-16. The performance of these three models on the FMD dataset are shown in Table 4. ResNet uses the residual connection between convolutional module, which makes it possible to build very deep network. We employ the standard ResNet18 and ResNet50 in our experiment. We can see the ResNet50 with 49 convolutional layers significantly outperforms the ResNet18 and VGG-16 networks. However, ResNet18 is worse than VGG-16, because the ResNet18 may be trapped into a local minima as it is unnecessary to use residual architecture when its depth is similar with the VGG-16 network. We further apply our locality-aware coding technique to three backbone models. For the VGG-16 and ResNet18 networks, locality-aware coding leads to 2.4% and 1.5% increase, respectively. For the ResNet50 network, in order to keep the dictionary over-complete, we add an extra convolutional layer with 1×1 kernel to reduce the dimensions of final feature from 2048 to 512. The performance of the reduced ResNet50 is also reported in Table 4 referred to as ResNet50$_{reduce}$. It is little worse than the original ResNet50, and our method can improve the ResNet50$_{reduce}$ with 1.2%.

Table 4
Comparisons for three different CNN backbones on the FMD dataset. Our method is applied to three different CNN backbones including VGG-16, ResNet18, and ResNet50.

Network model	Mean accuracy (%)
VGG-16	78.2
Ours$_{VGG-16}$	80.6
ResNet18	74.8
Ours$_{ResNet18}$	76.3
ResNet50	80.9
ResNet50$_{reduce}$	80.0
Ours$_{ResNet50}$	81.2

4.4.5. Visualization of the atoms

We provide a visualization of atoms chosen by a given feature, as shown in Fig. 9. Rather than maximize post-probability [67], we simply represent an atom by its nearest convolutional feature and corresponding receptive field patch. To this end, we first extract all convolutional feature as feature space. Then each atom finds its nearest neighbor in the feature space and binds with the neighbor's receptive field patch. Given an input convolutional feature, our method encodes it using the dictionary. As $L = 5$, 5 atoms are activated and their binded receptive field patches are shown in Fig. 9. The weighted average image that generated by atoms activation values is also shown. In this way, we can understand how our method encodes feature with regard to nearest neighbor, i.e., locality.

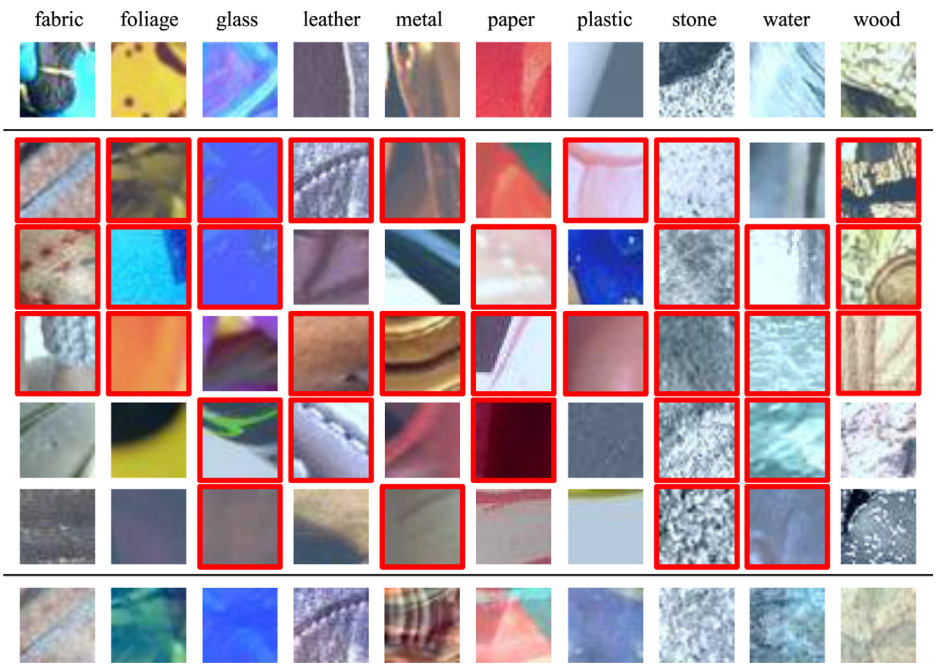

Fig. 9. Visualization of the atoms. We randomly pick 10 patches for each FMD dataset categories. Then we visualize the corresponding receptive field of atoms chosen by input patches. The first line is the picked patches, the middle five lines are the chosen atoms ordered by their values, and the bottom line is the weighted average image. The red box indicates that the atom and the input patch belong to the same category.

We experiment on the FMD dataset using randomly picked patches for 10 categories. In Fig. 9, the first line is the picked patches, the middle five lines are the chosen atoms ordered by their values, the bottom line is the weighted average image. The red highlighted indicates that the chosen atom belongs to the same category with the input patch. It can be seen that the atoms with the same category tend to have high weight and rank in the top, and the majority of the chosen atoms are the same category with the input patch. The visualization demonstrates the effectiveness of locality-aware coding. The locality-aware coding force representation is restrained by semantic locality. Thus the activation distribution for each encoded feature is highly peaked in one category region.

4.5. Discussion

Our model outperforms traditional off-the-shelf methods and achieves state-of-the-art results on DTD, FMD and KTH-T2b datasets, which is attributed to the following reasons. First, our method integrates the standard convolutional layers and the proposed locality-aware coding layer which is readily pluggable into any CNN architecture and amenable to training via backpropagation. This makes our method outperform the non-deep methods, e.g., hand-crafted methods [9–11] and FV-CNN [28]. Second, we build an orderless feature using a global representation layer. The orderless property is suitable for texture classification, and thus we can get better accuracy than FC-CNN [28] and VGG-16 [21]. Third, the locality-aware coding tends to enforce a tightness reconstruction based on the semantic clustering such that the activation distribution in an encoded feature is highly peaked in one class. This leads to a discriminative representation over the CNN features as discussed in Section 3.1 and Section 4.4.

As shown in Fig. 9, an obvious category correlation can be seen between hard samples like the paper and plastic category, which may result in misclassification. Because the paper and plastic are visually similar and will confuse with each other's feature atoms. We can increase the number of atoms to construct a larger feature space to address this problem, but with the increase of computational complexity brought by more atoms. We can also see that the stone category has very strong distribution in the feature space, so the chosen atoms are stone category uniformly. However, stone's strong performance may harm the encoding of the wood category, due to the mutual porous appearance character.

5. Conclusion and future work

In this paper, we have presented a new trainable layer named locality-aware coding by investigating the effectiveness of sparsity and locality in the CNNs for texture classification. Our method integrated the standard convolutional layers and a class-specific dictionary learning which is readily pluggable into any CNN architecture and amenable to training via backpropagation. The locality-aware coding layer built on top of CNN activations leads to better accuracy and improved the discriminative power of the CNNs feature while allowing arbitrary input image sizes. Our model outperforms traditional off-the-shelf methods and achieves state-of-the-art results on DTD, FMD and KTH-T2b datasets.

There are two interesting directions for future work:

- In this work, we exploited the global maxpooling scheme to aggregate the CNN features. Several works [68] demonstated that the second-order statistic representation e.g., the Symmetric Positive Definite (SPD) matrix, is more applicable for texture classification. Hence, it is interesting to aggregate the CNN features under an end-to-end deep network on SPD manifolds.
- Our method adopted a locality-aware coding layer conducted with the locality constraint to capture class-specific information of samples. It is interesting to incorporate more complex networks such as ResNet [23] to further improve performance.

Acknowledgements

This work was supported in part by the Natural Science Foundation of China (NSFC) under Grants No. 61702037 and No. 61773062, Beijing Municipal Natural Science Foundation under Grant No. L172027, and Beijing Institute of Technology Research Fund Program for Young Scholars.

References

[1] N. Werghi, C. Tortorici, S. Berretti, A. Del Bimbo, Boosting 3D LBP-based face recognition by fusing shape and texture descriptors on the mesh, IEEE Trans. Inf. Forensics Secur. 11 (5) (2016) 964–979.

[2] C. Li, Y. Huang, L. Zhu, Color texture image retrieval based on gaussian copula models of Gabor wavelets, Pattern Recognit. 64 (2017) 118–129.

[3] Y. Song, W. Cai, Y. Zhou, D.D. Feng, Feature-based image patch approximation for lung tissue classification, IEEE Trans. Med. Imaging 32 (4) (2013) 797–808.

[4] M. Varma, A. Zisserman, A statistical approach to material classification using image patch exemplars, IEEE Trans. Pattern Anal. Mach. Intell. 31 (11) (2009) 2032–2047.

[5] C. Kampouris, S. Zafeiriou, A. Ghosh, S. Malassiotis, Fine-grained material classification using micro-geometry and reflectance, in: Proceedings of the European Conference on Computer Vision (ECCV), Springer, 2016, pp. 778–792.

[6] S. Ahmad, L.-F. Cheong, Facilitating and exploring planar homogeneous texture for indoor scene understanding, in: Proceedings of the European Conference on Computer Vision (ECCV), Springer, 2016, pp. 35–51.

[7] L. Liu, P. Fieguth, X. Wang, M. Pietikäinen, D. Hu, Evaluation of LBP and deep texture descriptors with a new robustness benchmark, in: Proceedings of the European Conference on Computer Vision (ECCV), Springer, 2016, pp. 69–86.

[8] M. Cimpoi, S. Maji, I. Kokkinos, A. Vedaldi, Deep filter banks for texture recognition, description, and segmentation, Int. J. Comput. Vis. 118 (1) (2016) 65–94.

[9] T. Leung, J. Malik, Representing and recognizing the visual appearance of materials using three-dimensional textons, Int. J. Comput. Vis. 43 (1) (2001) 29–44.

[10] M. Varma, A. Zisserman, A statistical approach to texture classification from single images, Int. J. Comput. Vis. 62 (1) (2005) 61–81.

[11] M. Varma, A. Zisserman, Texture classification: are filter banks necessary? in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 2, IEEE, 2003, pp. II–691.

[12] S.P. Awate, T. Tasdizen, R.T. Whitaker, Unsupervised texture segmentation with nonparametric neighborhood statistics, in: Proceedings of the European Conference on Computer Vision (ECCV), Springer, 2006, pp. 494–507.

[13] T. Ojala, M. Pietikainen, T. Maenpaa, Multiresolution gray-scale and rotation invariant texture classification with local binary patterns, IEEE Trans. Pattern Anal. Mach. Intell. 24 (7) (2002) 971–987.

[14] Z. Guo, L. Zhang, D. Zhang, A completed modeling of local binary pattern operator for texture classification, IEEE Trans. Image Process. 19 (6) (2010) 1657–1663.

[15] Z. Zhu, X. You, C.P. Chen, D. Tao, W. Ou, X. Jiang, J. Zou, An adaptive hybrid pattern for noise-robust texture analysis, Pattern Recognit. 48 (8) (2015) 2592–2608.

[16] K. Wang, C.-E. Bichot, Y. Li, B. Li, Local binary circumferential and radial derivative pattern for texture classification, Pattern Recognit. 67 (2017) 213–229.

[17] L. Liu, S. Lao, P.W. Fieguth, Y. Guo, X. Wang, M. Pietikäinen, Median robust extended local binary pattern for texture classification, IEEE Trans. Image Process. 25 (3) (2016) 1368–1381.

[18] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems (NIPS), 2012, pp. 1097–1105.

[19] A. Babenko, A. Slesarev, A. Chigorin, V. Lempitsky, Neural codes for image retrieval, in: Proceedings of the European Conference on Computer Vision (ECCV), Springer, 2014, pp. 584–599.

[20] T. Kong, A. Yao, Y. Chen, F. Sun, HyperNet: towards accurate region proposal generation and joint object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 845–853.

[21] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556, 2014.

[22] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 1–9.

[23] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770–778.

[24] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: Towards real-time object detection with region proposal networks, in: Advances in Neural Information Processing Systems (NIPS), 2015, pp. 91–99.

[25] F. Yang, W. Choi, Y. Lin, Exploit all the layers: fast and accurate CNN object detector with scale dependent pooling and cascaded rejection classifiers, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2129–2137.

[26] A. Sharif Razavian, H. Azizpour, J. Sullivan, S. Carlsson, CNN features off-the-shelf: an astounding baseline for recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2014, pp. 806–813.

[27] T.-Y. Lin, S. Maji, Visualizing and understanding deep texture representations, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2791–2799.

[28] M. Cimpoi, S. Maji, A. Vedaldi, Deep filter banks for texture recognition and segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 3828–3836.

[29] A. Babenko, V. Lempitsky, Aggregating local deep features for image retrieval, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1269–1277.

[30] L. Liu, P. Wang, C. Shen, L. Wang, A. van den Hengel, C. Wang, H.T. Shen, Compositional model based fisher vector coding for image classification, IEEE Trans. Pattern Anal. Mach. Intell. PP (99) (2017). 1–1, doi: 10.1109/TPAMI.2017.2651061.

[31] J. Yue-Hei Ng, F. Yang, L.S. Davis, Exploiting local features from deep networks for image retrieval, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2015, pp. 53–61.

[32] L. Liu, C. Shen, A. van den Hengel, Cross-convolutional-layer pooling for image recognition, IEEE Trans. Pattern Anal. Mach. Intell. PP (99) (2016). 1–1, doi: 10.1109/TPAMI.2016.2637921.

[33] L. Liu, C. Shen, A. van den Hengel, The treasure beneath convolutional layers: cross-convolutional-layer pooling for image classification, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 4749–4757.

[34] J.L. Long, N. Zhang, T. Darrell, Do convnets learn correspondence? in: Advances in Neural Information Processing Systems (NIPS), 2014, pp. 1601–1609.

[35] Y. Ioannou, D. Robertson, D. Zikic, P. Kontschieder, J. Shotton, M. Brown, A. Criminisi, Decision forests, convolutional networks and the models in-between, (2016), arXiv:1603.01250.

[36] L. Wang, W. Ouyang, X. Wang, H. Lu, Visual tracking with fully convolutional networks, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 3119–3127.

[37] N. Srivastava, G.E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Lear. Res. 15 (1) (2014) 1929–1958.

[38] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks., in: Aistats, vol. 15, 2011, p. 275.

[39] J. Li, T. Zhang, W. Luo, J. Yang, X.-T. Yuan, J. Zhang, Sparseness analysis in the pretraining of deep neural networks, IEEE Trans. Neural Netw. Learn. Syst. PP (99) (2016) 1–14, doi:10.1109/TNNLS.2016.2541681.

[40] H. Lee, C. Ekanadham, A.Y. Ng, Sparse deep belief net model for visual area v2, in: Advances in Neural Information Processing Systems (NIPS), 2008, pp. 873–880.

[41] V. Nair, G.E. Hinton, Rectified linear units improve restricted Boltzmann machines, in: Proceedings of the International Conference on Machine Learning (ICML), 2010, pp. 807–814.

[42] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, T. Darrell, DeCAF: a deep convolutional activation feature for generic visual recognition, in: Proceedings of the International Conference on Machine Learning (ICML), vol. 32, 2014, pp. 647–655.

[43] L.v.d. Maaten, G. Hinton, Visualizing data using t-SNE, J. Mach. Lear. Res. 9 (Nov) (2008) 2579–2605.

[44] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, Y. Gong, Locality-constrained linear coding for image classification, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2010, pp. 3360–3367.

[45] C.-P. Wei, Y.-W. Chao, Y.-R. Yeh, Y.-C.F. Wang, Locality-sensitive dictionary learning for sparse representation based classification, Pattern Recognit. 46 (5) (2013) 1277–1287.

[46] J. Sivic, A. Zisserman, Video Google: a text retrieval approach to object matching in videos, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), IEEE, 2003, pp. 1470–1477.

[47] H. Jégou, F. Perronnin, M. Douze, J. Sanchez, P. Perez, C. Schmid, Aggregating local image descriptors into compact codes, IEEE Trans. Pattern Anal. Mach. Intell. 34 (9) (2012) 1704–1716.

[48] F. Perronnin, Y. Liu, J. Sánchez, H. Poirier, Large-scale image retrieval with compressed fisher vectors, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2010, pp. 3384–3391.

[49] Y. Sun, X. Wang, X. Tang, Deep convolutional network cascade for facial point detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013, pp. 3476–3483.

[50] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, J. Sivic, Netvlad: CNN architecture for weakly supervised place recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 5297–5307.

[51] Y. Wen, K. Zhang, Z. Li, Y. Qiao, A discriminative feature learning approach for deep face recognition, in: Proceedings of the European Conference on Computer Vision (ECCV), Springer, 2016, pp. 499–515.

[52] Y. Gong, L. Wang, R. Guo, S. Lazebnik, Multi-scale orderless pooling of deep convolutional activation features, in: Proceedings of the European Conference on Computer Vision (ECCV), Springer, 2014, pp. 392–407.

[53] L. Liu, C. Shen, L. Wang, A. Van Den Hengel, C. Wang, Encoding high dimensional local features by sparse coding based fisher vectors, in: Advances in Neural Information Processing Systems (NIPS), 2014, pp. 1143–1151.

[54] M.D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: Proceedings of the European Conference on Computer Vision (ECCV), Springer, 2014, pp. 818–833.

[55] J. Ng, F. Yang, L. Davis, Exploiting local features from deep networks for image retrieval, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2015, pp. 53–61.

[56] G. Tolias, R. Sicre, H. Jégou, Particular object retrieval with integral max-pooling of CNN activations, in: Proceedings of the International Conference on Learning Representations (ICLR), 2016.

[57] K. He, X. Zhang, S. Ren, J. Sun, Spatial pyramid pooling in deep convolutional networks for visual recognition, IEEE Trans. Pattern Anal. Mach. Intell. 37 (9) (2015) 1904–1916.

[58] P. Koniusz, A. Cherian, Sparse coding for third-order super-symmetric tensor descriptors with application to texture recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 5395–5403.
[59] Y. Quan, C. Bao, H. Ji, Equiangular kernel dictionary learning with applications to dynamic texture analysis, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 308–316.
[60] L.A. Gatys, A.S. Ecker, M. Bethge, Texture synthesis and the controlled generation of natural stimuli using convolutional neural networks, in: Bernstein Conference 2015, 2015, p. 219.
[61] T.-Y. Lin, A. RoyChowdhury, S. Maji, Bilinear CNN models for fine-grained visual recognition, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1449–1457.
[62] H. Zhang, J. Xue, K. Dana, Deep ten: texture encoding network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 708–717.
[63] M. Lin, Q. Chen, S. Yan, Network in network, (2013), arXiv:1312.4400.
[64] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, A. Vedaldi, Describing textures in the wild, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 3606–3613.
[65] L. Sharan, C. Liu, R. Rosenholtz, E.H. Adelson, Recognizing materials using perceptually inspired features, Int. J. Comput. Vis. 103 (3) (2013) 348–371.
[66] B. Caputo, E. Hayman, P. Mallikarjuna, Class-specific material categorisation, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), vol. 2, IEEE, 2005, pp. 1597–1604.
[67] L.A. Gatys, A.S. Ecker, M. Bethge, Image style transfer using convolutional neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 2414–2423.
[68] P. Li, J. Xie, Q. Wang, W. Zuo, Is second-order information helpful for large-scale visual recognition? in: 2017 IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2089–2097.

Xingyuan Bu received the B.S., and M.S. degrees in computer science from Beijing Institute of Technology (BIT), Beijing, China, in 2015, and 2018. Currently he is a researcher in the Department of Computer Vision Technology, Baidu, China. His research interests include deep learning, computer vision and machine learning, specifically in the areas of visual classification and detection.

Yuwei Wu received the Ph.D. degree in computer science from Beijing Institute of Technology (BIT), Beijing, China, in 2014. He is now an Assistant Professor at School of Computer Science, BIT. From August 2014 to August 2016, he was a post-doctoral research fellow at School of Electrical & Electronic Engineering, Nanyang Technological University (NTU), Singapore. He has strong research interests in computer vision and information retrieval. He received outstanding Ph.D. Thesis award from BIT, and Distinguished Dissertation Award Nominee from China Association for Artificial Intelligence (CAAI).

Zhi Gao received the B.S. degree in computer science from Beijing Institute of Technology (BIT), Beijing, China, in 2017. Now, he is a Ph.D. candidate in the Beijing Laboratory of Intelligent Information Technology, BIT, Beijing, China. His research interests include pattern recognition, machine learning, and computer vision on the Riemannian manifold.

Yunde Jia (M'11) is professor of computer science at BIT, and serves as the Director of the Beijing Laboratory of Intelligent Information Technology. He received the B.S., M.S., and Ph.D. degrees in mechatronics from the Beijing Institute of Technology (BIT) in 1983, 1986, and 2000, respectively. He has previously served as the Executive Dean of the School of Computer Science at BIT from 2005 to 2008. He was a Visiting Scientist at Carnegie Mellon University from 1995 to 1997, and a Visiting Fellow at the Australian National University in 2011. His current research interests include computer vision, media computing, and intelligent systems.