Pattern Recognition 92 (2019) 1–12

Contents lists available at ScienceDirect

Pattern Recognition

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

Learning a robust representation via a deep network on symmetric positive definite manifolds

Zhi Gao[a], Yuwei Wu[a,*], Xingyuan Bu[a], Tan Yu[b], Junsong Yuan[c], Yunde Jia[a]

[a] Beijing Laboratory of Intelligent Information Technology, School of Computer Science, Beijing Institute of Technology (BIT), Beijing, 100081, China
[b] School of EEE, Nanyang Technological University (NTU), 639798, Singapore
[c] Department of Computer Science and Engineering, State University of New York at Buffalo, USA

A R T I C L E I N F O

Article history:
Received 5 April 2018
Revised 2 January 2019
Accepted 13 March 2019
Available online 14 March 2019

Keywords:
Feature aggregation
SPD Matrix
Riemannian manifold
Deep convolutional network

A B S T R A C T

Recent studies have shown that aggregating convolutional features of a Convolutional Neural Network (CNN) can obtain impressive performance for a variety of computer vision tasks. The Symmetric Positive Definite (SPD) matrix becomes a powerful tool due to its remarkable ability to learn an appropriate statistic representation to characterize the underlying structure of visual features. In this paper, we propose a method of aggregating deep convolutional features into a robust representation through the SPD generation and the SPD transformation under an end-to-end deep network. To this end, several new layers are introduced in our method, including a nonlinear kernel generation layer, a matrix transformation layer, and a vector transformation layer. The nonlinear kernel generation layer is employed to aggregate convolutional features into a kernel matrix which is guaranteed to be an SPD matrix. The matrix transformation layer is designed to project the original SPD representation to a more compact and discriminative SPD manifold. The vectorization and normalization operations are performed in the vector transformation layer to take the upper triangle elements of the SPD representation and carry out the power normalization and l_2 normalization to reduce the redundancy and accelerate the convergence. The SPD matrix in our network can be considered as a mid-level representation bridging convolutional features and high-level semantic features. Results of extensive experiments show that our method notably outperforms state-of-the-art methods.

1. Introduction

Deep Convolutional Neural Networks (CNNs) have shown great success in many vision tasks. There are several successful networks, *e.g.*, VGG [1] and ResNet [2]. Driven by the emergence of large-scale data sets and fast development of computation power, features based on CNNs have proven to perform remarkably well on a wide range of visual recognition tasks. Liu et al. [3], and Babenko and Lempitsky [4] demonstrated that convolutional features can be seen as a set of local features which can capture the visual representation related to objects. To make better use of deep convolutional features, many efforts have been devoted to aggregating them, such as max pooling [5], cross-dimensional pooling [6], sum pooling [4], and bilinear pooling [3,7]. However, modeling these convolutional features to boost the feature learning ability of a CNN is still a challenging task. This work investigates a more effective scheme to aggregate convolutional features into a robust representation via a Symmetric Positive Definite (SPD) manifold network in an end-to-end framework.

The SPD matrix has shown the powerful representation ability and been widely used in the computer vision community, such as the face video recognition [8], 3D face recognition [9], medical image processing [10,11], and metric learning [12]. Through the theory of the non-Euclidean Riemannian geometry, the SPD matrix often turns out to be better suited in capturing desirable data distribution properties.

The second-order statistic information of convolutional features, *e.g.*, the covariance matrix or Gaussian distribution, is the commonly used SPD matrix representation endowed with CNNs [13–15]. The dimensionality of convolutional features extracted from CNNs may be much larger than that of hand-crafted features. As a result, the covariance matrix or Gaussian distribution is inferior to precisely model the real convolutional feature distribution. When the dimensionality of features is larger than the number of features, the covariance matrix and Gaussian distribution are Symmetric Positive SemiDefinite (PSD) matrices. The PSD matrix representation has an unreasonable manifold structure and may lose

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

https://doi.org/10.1016/j.patcog.2019.03.007
0031-3203/© 2019 Elsevier Ltd. All rights reserved.

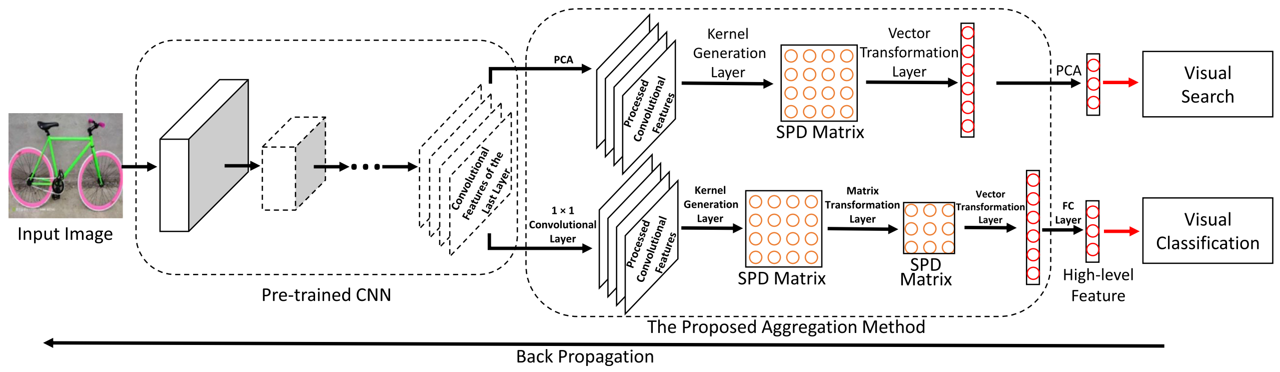

Fig. 1. The flowchart of our SPD aggregation network. We focus on obtaining an robust SPD matrix representation from convolutional features via the SPD generation and transformation. The PCA and 1×1 convolutional layer are applied to preprocess convolutional features from a pre-trained CNN. Then, the kernel generation layer, matrix transformation layer, and vector transformation layer are employed successively. Several visual search tasks are utilized to demonstrate the power of our method only using a pre-trained network, and visual classification tasks are under an end-to-end framework by finetuning the network.

some high-level variation information among convolutional features and be less discriminative than the real SPD representation. Moreover, most covariance matrices embedded into deep networks only contain the linear correlation between features. Owning the ability of capturing the nonlinear relationships between features is indispensable for a generic representation.

It is thus desirable that a more discriminative and suitable SPD representation aggregated from deep convolutional features should be established in an end-to-end framework for visual analysis. To this end, we design a series of new layers to overcome the existing issues aforementioned based on the following two observations.

- Kernel functions possess an ability of modeling nonlinear relationships of data, and they are easy and flexible to be computed. Wang et al. [16] have witnessed significant advances of positive definite kernel functions whose kernel matrices are real SPD matrices, no matter what the dimensionality and the number of features are. Since many kernel functions are differentiable, such as Radial Basis Function (RBF) kernel function, Polynomial kernel function, and Laplacian kernel function, they can be readily not only utilized to aggregated features but also embedded into a network to carry out end-to-end training. The kernel function is well aligned with the design requirement of a deep network.
- Many works [17–19] showed that a learnable mapping from the SPD matrix to another makes the representation become more discriminative and task-specific. The mapped matrix is still an SPD matrix which not only has characteristics of a general SPD matrix that captures desirable properties of visual features but also is more suitable and discriminative for the specific visual task.

Motivated by empirical observations mentioned above, we introduce a convolutional feature aggregation scheme which consists of the SPD generation and the SPD transformation. Three new layers, *i.e.,* a kernel generation layer, a matrix transformation layer, and a vector transformation layer are designed to replace the traditional pooling layers and fully connected (FC) layers. In particular, we deem each feature map as a sample and present a kernel generation layer using a nonlinear kernel function to generate an SPD matrix. The proposed kernel matrix models nonlinear relationships between feature maps and ensures that the SPD matrix is nonsingular. The proposed matrix transformation layer can be employed to transform the SPD matrix to a more compact and discriminative one by learnable parameters. It can not only capture the real spatial information but also encode high-level variation information. Thanks to the symmetry property of the SPD matrix, the vector transformation layer carries out the upper triangle vectorization and normalization operations on the SPD matrix. Through these layers, a robust vector representation is got under an end-to-end framework. The proposed SPD aggregation scheme can also be utilized under a non-learnable process only with a pre-trained network, except the matrix transformation layer, since the matrix transformation layer needs to be optimized. The SPD representation actually acts as a mid-level representation bridging convolutional features and high-level semantics features. In this paper, we apply it to visual search and classification tasks to show the effectiveness of our method in both non-learnable and end-to-end frameworks, and the applied network architecture is shown in Fig. 1.

In summary, our contributions are three-fold.

(1) We formulate the convolutional feature aggregation as an SPD matrix non-linear generation and transformation problem on the Riemannian manifold to obtain a robust representation. Our middle-level SPD representation can well characterize the underlying structure of convolutional features.
(2) We carry out the nonlinear aggregation of convolutional features under an end-to-end Riemannian deep network architecture, where three novel layers are introduced. The state-of-the-art performance of our SPD aggregation network is consistently achieved over visual search and visual classification tasks.
(3) We exploit the faster matrix operation to speed up the computation in the kernel generation layer. In addition, we present the component decomposition and retraction of the orthogonal Stiefel manifold to carry out the backpropagation in the matrix transformation layer.

The remaining sections are organized as follows. We review recent works about feature aggregation methods in both the Euclidean Space and Riemannian Space in Section 2. Section 3 presents details of our SPD aggregation method. We report and discuss experimental results in Section 4, and conclude the paper in Section 5.

2. Related work

Feature aggregation is an important component for computer vision tasks. Although recent works have witnessed significant advances of CNNs, it is still a challenging work to find a suitable way to aggregate convolutional features. Several state-of-the-art feature aggregation methods are embedded into the deep network, such as Vector of Locally Aggregated Descriptor (VLAD) [20,21], and Fisher Vector (FV) [22]. Recent researches show that exploiting the manifold structure is more effective than the hypothetical Euclidean distribution in several visual tasks. The difference between our method and the traditional aggregation methods [20–22] in the Euclidean space is that we use the powerful SPD manifold to capture desirable feature distributions. In the following, we review typical techniques of feature aggregation in the SPD manifold.

Recently, the second-order statistic information has been demonstrated to have better performance than the first-order statistic information [13]. The covariance matrix is the commonly used second-order statistic information representation, which is on the SPD manifold. Huang et al. [8] and Faraki et al. [23] aggregated local features into a covariance matrix to represent data. Zhang et al. [10] employed sparse inverse covariance estimation (SICE) to model brain connectivity networks. These methods [8,10,23] model linear relationships of local features on the SPD manifold, but they do not utilize the powerful deep features or are not applied in the end-to-end framework. By contrast, we build the SPD aggregation in an end-to-end deep network, capturing complex nonlinear variation information of features.

To overcome this issue, Lin et al. [3] presented a general orderless bilinear pooling model computing the outer product of local features. Ionescu et al. [14] proposed a DeepO2P network that uses a covariance matrix as the image representation and maps points on the manifold to the logarithm tangent space, and derived a new chain rule for derivatives of these operations. These works [3,8,10,14,23] mainly focus on generating a second-order statistics representation.

In order to improve the discriminative power of the second-order statistics representation, many works devoted to transforming the second-order statistics representation. Li et al. [13] presented a matrix power normalization (MPN) method on the second-order statistics representation. This work can tackle the PSD issue of the covariance matrix. Gao et al. [7] proposed two versions of compact bilinear pooling (CMP) via the Random Maclaurin (RM) and Tensor Sketch (TS). Kong and Fowlkes [24] introduced a learnable low-rank bilinear pooling (LRBP) method to reduce the dimensionality of the bilinear representation. The two works [7,24] aim to relieve the high cost of the computation and memory of the second-order statistics representation. Yu and Salzmann [25] studied the statistical distribution of the second-order statistics representation and proposed a statistically-motivated second-order (SMSO) pooling. They compressed the second-order representation to a Chi-square distribution vector and normalized it to a Gaussian distribution vector. They also introduced a covariance descriptor unit [15] based on the SMSO pooling and embedded it into deep networks. These methods [3,7,13–15,24,25] are confined to the drawbacks of the covariance matrix or Gaussian distribution. In contrast, our method exploits the kernel matrix to improve the PSD representation and simple linear relationships between convolutional features. Moreover, our SPD transformation consists of both the compression and normalization, and our learnable transformation can implement the mapping from a matrix to another matrix, not to a vector. Recently, Engin et al. [26] designed a deep kernel matrix based SPD representation. Compared with [26], the transformation layers in our method lead to a better discriminative power.

Other SPD Riemannian networks mainly map an SPD matrix to a more discriminative manifold space. Dong et al. [17], and Huang and Gool [18] proposed Riemannian networks contemporaneously, in which inputs of their networks are SPD matrices. The networks map a high dimensional SPD matrix to a low dimensional discriminative SPD manifold by a nonlinear mapping. However, the two works [17,18] only focused on how to transform the SPD matrix without utilizing the powerful convolutional features. The generation of the input SPD matrix cannot be guided by the loss function. In contrast, our method concentrates on both the SPD transformation and the SPD generation from convolutional features. They are jointly training in our network.

Our work is closely related with [13,15,26]. We make it clear that the proposed convolutional feature aggregation method is composed of the SPD generation and the SPD transformation. Compared with [13], our method utilizes the kernel matrix as the representation instead of the second-order statistic covariance matrix, characterizing complex nonlinear variation information of features. Moreover, our aggregation method contains a learnable transformation process compared with [13], making the SPD representation more compact and robust in an end-to-end framework. The generated SPD matrix in our method is more powerful than the covariance matrix in [15], avoiding drawbacks of the PSD matrix. In addition, instead of a mapping from a matrix to a vector, the vectorization operation in our work is taking the upper triangle of a matrix since there are already mapping operations between SPD matrices. Compared with [26], our method contains the compression, normalization, and generation simultaneously. Our transformation layers bring a more task-driven representation.

3. SPD aggregation method

Our model aims to aggregate convolutional features into a powerful representation via the SPD manifold in an end-to-end fashion. To this end, we design three novel layers, *i.e.*, a kernel generation layer, a matrix transformation layer, and a vector transformation layer. We will elaborate on our method in this section.

3.1. Preprocessing of convolutional features

A CNN model trained on a large dataset such as ImageNet has a better general representation ability. We expect to fuse convolutional features of the last convolutional layer and adjust the dimensionality of convolutional features for different computer vision tasks. We exploit the Principal Component Analysis (PCA) between the last convolutional layer and the kernel generation operation to reduce the dimensionality of each local convolutional feature in non-learnable tasks, so that it is beneficial to alleviate information redundancy. Each dimensionality has the largest variance and is independent of other dimensionalities. For the end-to-end tasks, we introduce a convolutional layer whose filter's size is 1×1 between the last convolutional layer of the pre-trained network and the kernel generation layer to make the processed convolutional features more adaptive to the SPD matrix. A Relu layer follows the 1×1 convolutional layer to enhance the nonlinear ability.

3.2. Kernel generation layer

We present the kernel generation layer to aggregate convolutional features into an SPD matrix. Let $X \in \mathbb{R}^{C \times H \times W}$ be 3-dimensional convolutional features. C is the number of channels. H and W are the height and width of each feature map, respectively. Let $x_i \in \mathbb{R}^{C}$ denote the ith local feature, and there are N local features in total, where $N = H \times W$. $f_i \in \mathbb{R}^{H \times W}$ is the ith feature map.

Although several approaches have applied a covariance matrix Cov to be a generic feature representation and obtained promising results, two issues remain to be addressed. First, the rank of the covariance matrix extracted from an image region should hold $rank(Cov) \le \min(C, N-1)$. Otherwise, the covariance matrix is prone to be singular when the dimensionality C of local features is larger than the number of local features N. Second, for a generic representation, the capability of modeling nonlinear feature relationships is essential. However, the covariance matrix only evaluates the linear correlation between features.

To address these issues, we adopt the nonlinear kernel matrix as a generic feature representation to aggregate deep convolutional features. In particular, we take advantage of the Riemannian structure of SPD matrices to describe the high-order statistic and nonlinear correlations between deep convolutional features. The nonlinear kernel matrix is capable of modeling nonlinear feature relationships and guaranteed to be nonsingular. Different from the traditional kernel-based methods whose entries evaluate the similarity between a pair of samples, we apply the kernel mapping to each feature map $f_1, f_2, \cdots, f_C$ rather than each local feature $x_1, x_2, \cdots, x_N$ [16]. Mercer kernels are usually employed to carry out the mapping implicitly. The Mercer kernel is a function $\mathcal{K}(\cdot,\cdot)$ which can generate a kernel matrix $K \in \mathbb{R}^{C\times C}$ using pairwise inner products between mapped feature maps for all input convolutional features. The K_{ij} in our nonlinear kernel matrix K can be defined as

$$K_{ij} = \mathcal{K}\big(f_i, f_j\big) = \big\langle \phi(f_i), \phi(f_j)\big\rangle, \tag{1}$$

where $\phi(\cdot)$ is an implicit mapping. In this paper, we exploit the Radial Basis Function (RBF) kernel function expressed as

$$\mathcal{K}\big(f_i, f_j\big) = exp\big(-\|f_i - f_j\|^2/2\sigma^2\big), \tag{2}$$

where σ is a positive constant and set to the mean Euclidean distance of all feature maps in our experiments. What Eq. (2) reveals is the nonlinear relationship between convolutional features.

We show an important theorem for the kernel generation operation. Based on the Theorem 1, the kernel matrix K of the RBF kernel function is guaranteed to be positive definite no matter what C and N are.

Theorem 1. *Let $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^M$ denotes a set of different points and $\mathbf{x}_i \in \mathbb{R}^n$, and any $\mathbf{x}_i$ are not equal. Then the kernel matrix $K \in \mathbb{R}^{M\times M}$ of the RBF kernel function $\mathcal{K}$ on $\mathbf{X}$ is guaranteed to be a positive definite matrix, whose (j, k)th element is $K_{jk} = \mathcal{K}\big(\mathbf{x}_j, \mathbf{x}_k\big) = \exp\big(-\alpha\|\mathbf{x}_j - \mathbf{x}_k\|^2\big)$ and $\alpha > 0$.*

Proof. The Fourier transform convention $\hat{\mathcal{K}}(\xi)$ of the RBF kernel function $\mathcal{K}\big(\mathbf{x}_j, \mathbf{x}_k\big) = \exp\big(-\alpha\|\mathbf{x}_j - \mathbf{x}_k\|^2\big)$ is

$$\hat{\mathcal{K}}(\xi) = (2\pi/\alpha)^{n/2}\int_{R^n} e^{i\xi\mathbf{x}_j}e^{-i\xi\mathbf{x}_k}e^{-\|\xi\|^2/2\alpha}d\xi. \tag{3}$$

Then we calculate the quadratic form of the kernel matrix K. Let $\mathbf{c} = (c_1, \ldots, c_M) \in \mathbb{R}^{M\times 1}$ denote an arbitrary nonzero vector. The quadratic form Q is

$$\begin{aligned} Q &= \mathbf{c}^\top K\mathbf{c} = \sum_{j=1}^{M}\sum_{k=1}^{M} c_j c_k exp\big(-\alpha\|\mathbf{x}_j - \mathbf{x}_k\|^2\big) \\ &= \sum_{j=1}^{M}\sum_{k=1}^{M} c_j c_k (2\pi/\alpha)^{n/2}\int_{R^n} e^{i\xi\mathbf{x}_j}e^{-i\xi\mathbf{x}_k}e^{-\|\xi\|^2/(2\alpha)}d\xi \\ &= (2\pi/\alpha)^{n/2}\int_{R^n} e^{-\|\xi\|^2/(2\alpha)}\left\|\sum_{k=1}^{M} c_k e^{-i\xi\mathbf{x}_k}\right\|^2 d\xi, \end{aligned} \tag{4}$$

where $\top$ is the transpose operation. Because $e^{-\|\xi\|^2/(2\alpha)}$ is a positive and continuous function, the quadratic form $Q = 0$ on the condition that

$$\sum_{k=1}^{M} c_k e^{-i\xi\mathbf{x}_k} = 0. \tag{5}$$

However, since $\mathbf{x}_i$ are not euqal, the complex exponentials $e^{-i\xi\mathbf{x}_1}, \cdots, e^{-i\xi\mathbf{x}_M}$ are linearly independence. Accordingly, $Q > 0$ and the kernel matrix K is an SPD matrix. □

In this work, K is the generated SPD matrix as the mid-level image representation. Any SPD manifold optimization can be applied directly, without the worry of being applied to the PSD matrix. As we know, the kernel generation layer should be differentiable to meet the requirement of an end-to-end deep learning framework. Clearly, Eq. (2) is differentiable with respect to the input X. Denoting by $\mathcal{L}$ the loss function, the gradient with respect to the kernel matrix is $\frac{\partial\mathcal{L}}{\partial K}$. $\frac{\partial\mathcal{L}}{\partial K_{ij}}$ is an element in $\frac{\partial\mathcal{L}}{\partial K}$. We can compute the partial derivatives of $\mathcal{L}$ with respect to f_i and f_j, which are

$$\begin{aligned} \frac{\partial\mathcal{L}}{\partial f_i} &= \sum_{j=1}^{C}\frac{\partial\mathcal{L}}{\partial K_{ij}}\frac{K_{ij}}{-\sigma^2}\big(f_i - f_j\big) \\ \frac{\partial\mathcal{L}}{\partial f_j} &= \sum_{i=1}^{C}\frac{\partial\mathcal{L}}{\partial K_{ij}}\frac{K_{ij}}{-\sigma^2}\big(f_j - f_i\big). \end{aligned} \tag{6}$$

In this process, the gradient of the SPD matrix can flow back to convolutional features.

During forward propagation Eq. (2) and backward propagation Eq. (6), we have C^2 cycles to compute the kernel matrix K and $2C^2$ cycles to gain the gradient $\frac{\partial\mathcal{L}}{\partial X}$ with respect to convolutional features. Obviously, both the forward and backward propagations are computationally demanding. It is well known that the computation using matrix operations is preferable due to the parallel computing in computers. Accordingly, our kernel generation layer is able to be calculated in a faster way via matrix operations. Let's reshape the convolutional features $X \in \mathbb{R}^{C\times H\times W}$ to a matrix $M \in \mathbb{R}^{C\times N}$. Each row of M is a reshaped feature map $\mathbf{f}_i \in \mathbb{R}^{1\times N}$ obtained from f_i and each column of M is a convolutional local feature x_i. Note that, $\|f_i - f_j\|^2$ in Eq. (2) can be expanded to $\|f_i - f_j\|^2 = \mathbf{f}_i\mathbf{f}_i^\top - 2\mathbf{f}_i\mathbf{f}_j^\top + \mathbf{f}_j\mathbf{f}_j^\top$. For each of inner products $\mathbf{f}_i\mathbf{f}_i^\top$, $2\mathbf{f}_i\mathbf{f}_j^\top$, and $\mathbf{f}_j\mathbf{f}_j^\top$, it needs to be calculated C^2 times in cycles of Eq. (2). Now, we can convert C^2 times inner products operation to a once matrix multiplication operation,

$$\begin{aligned} K1 &= (M\circ M)\mathbf{1}^\top \\ K2 &= \mathbf{1}(M\circ M)^\top \\ K3 &= MM^\top, \end{aligned} \tag{7}$$

where $\circ$ is the Hadamard product and $\mathbf{1} \in \mathbb{R}^{C\times N}$ is a matrix whose elements are all "1"s. $K1$, $K2$ and $K3$ are all $C\times C$ real matirces. The element $K1(i, j)$ is the square of 2-norm of ith row vector of M, and is equal to the calculation output of $\mathbf{f}_i\mathbf{f}_i^\top$. The element $K2(i, j)$ is the square of 2-norm of jth column vector of M, and is equal to the calculation output of $\mathbf{f}_j\mathbf{f}_j^\top$. The element $K3(i, j)$ is equal to $\mathbf{f}_i\mathbf{f}_j^\top$. $K1$, $K2$ and $K3$ can be calculated in advance.

Therefore, we compute $-\|f_i - f_j\|^2/2\sigma^2$ in Eq. (2) by the matrix addition and multiplication, and implement the $exp(\cdot)$ in a parallel computing way instead of calculating each element in the cycle. To summarize, the kernel matrix K can be calculated by matrix operations as follows.

$$K = exp\big(-(K1 + K2 - 2K3)/2\sigma^2\big), \tag{8}$$

where $exp(A)$ means the exponential operation to each element in the matrix A. Although calculating the $exp(\cdot)$ function directly is time-consuming, it can be computed efficiently in a matrix form through Eq. (8), which is faster than through Eq. (2). Similarly, back propagation process in Eq. (6) can also be carried out in the matrix operation which is given by

$$\frac{\partial\mathcal{L}}{\partial M} = 4\left(\mathbf{1}^\top\left(\frac{\frac{\partial\mathcal{L}}{\partial K}\circ K}{-2\sigma^2}\right)\circ M^\top - M^\top\left(\frac{\frac{\partial\mathcal{L}}{\partial K}\circ K}{-2\sigma^2}\right)\right)^\top. \tag{9}$$

Remark: The covariance matrix representation, as a special case of SPD matrices, captures feature correlations compactly in an object region, and therefore has been proven to be effective for many applications. Given the local features $x_1, x_2, \cdots, x_N$, the traditional covariance representation Cov is defined as

$$Cov = \frac{1}{N-1}\sum_{i=1}^{N}(x_i-\mu)(x_i-\mu)^\top, \tag{10}$$

where $\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$ is the mean vector. The covariance matrix can also be seen as a kernel matrix where the (i, j)th element of Cov is expressed as

$$Cov_{ij} = \left\langle \frac{\overline{f_i}}{\sqrt{N-1}}, \frac{\overline{f_j}}{\sqrt{N-1}} \right\rangle, \tag{11}$$

where $\langle\cdot,\cdot\rangle$ denotes the inner product, $\overline{f_i} = f_i - \mu_i\mathbf{1}$ and μ_i is the mean value of f_i. Therefore, the covariance matrix corresponds to a special case of the nonlinear kernel matrix defined in Eq. (1), where $\phi(f_i) = (\overline{f_i} - \mu_i\mathbf{1})/\sqrt{N-1}$. Through this way, we can find that the covariance matrix only contains the simple linear correlation between features. Moreover, whether the covariance matrix is a positive definite matrix depends on C and N, *i.e., rank*$(Cov) \le \min(C, N-1)$.

3.3. Matrix transformation layer

Inspired by Yu and Salzmann [15], we add a learnable layer to make the network more flexible and adaptive to the specific end-to-end task. Based on the SPD matrix generated by the kernel generation layer, we expect to transform the existing SPD representation to be a more discriminative, suitable and desirable matrix. To preserve the powerful ability of the SPD matrix, the transformed matrix should also be an SPD matrix. Moreover, we attempt to adjust the dimensionality of the SPD matrix to make it more flexible and compact. Here, we design a matrix transformation layer in our network.

Let's define the Riemannian manifold of the $n\times n$ SPD matrix as Sym_n^+. The output SPD matrix K of the kernel generation layer lies on the manifold Sym_C^+. We use a matrix mapping to complete the transformation operation. We map the input SPD matrix which lies on the original manifold Sym_C^+ to a new discriminative and compact SPD manifold $Sym_{C'}^+$, where C' is the dimensionality of the matrix transformation layer. In this way, the desired transformation operation can be obtained by learning a mapping between manifolds. Given a $C\times C$ SPD matrix K as an input, the output SPD matrix can be calculated as

$$Y = W^\top K W, \tag{12}$$

where $Y \in \mathbb{R}^{C'\times C'}$ is the output of the transformation layer, and $W \in \mathbb{R}^{C\times C'}$ are the learnable parameters which are randomly initialized during training. C' controls the size of Y. Based on the Theorem 2, the learnable parameters W should be a column full rank matrix to make Y be an SPD matrix as well.

Theorem 2. *Let $A \in \mathbb{R}^{C\times C}$ denote an SPD matrix, $W \in \mathbb{R}^{C\times C'}$ and $B = W^\top AW$, where $C \ge C'$. B is an SPD matrix if and only if W is a column full rank matrix, i.e., rank$(W) = C'$.*

Proof. If A is an SPD matrix, W is a column full rank matrix and $rank(W) = C'$. For homogeneous equations $W\mathbf{x} = \mathbf{0}$ and $\mathbf{x} \in \mathbb{R}^{C'\times 1}$, $W\mathbf{x} = \mathbf{0}$ only has a zero solution, where $\mathbf{0}$ is the zero vector. For arbitrary nonzero vector $\mathbf{x}$, $W\mathbf{x} \neq \mathbf{0}$. We calculate the quadric form $\mathbf{x}^\top B\mathbf{x}$,

$$\mathbf{x}^\top B\mathbf{x} = \mathbf{x}^\top W^\top AW\mathbf{x} = (W\mathbf{x})^\top A(W\mathbf{x}). \tag{13}$$

Because $W\mathbf{x} \neq \mathbf{0}$ and A is an SPD matrix, $\mathbf{x}^\top B\mathbf{x} > \mathbf{0}$. This proves that B is an SPD matrix.

On the other hand, if B is an SPD matrix, for arbitrary nonzero vector $\mathbf{x} \in \mathbb{R}^{C'\times 1}$, $\mathbf{x}^\top B\mathbf{x} = (W\mathbf{x})^\top A(W\mathbf{x}) > \mathbf{0}$. As A is an SPD matrix, $W\mathbf{x} \neq \mathbf{0}$. Only if $\mathbf{x} = \mathbf{0}$ can lead to $W\mathbf{x} = \mathbf{0}$. Accordingly, $rank(W) = C'$ and W is a column full rank matrix. □

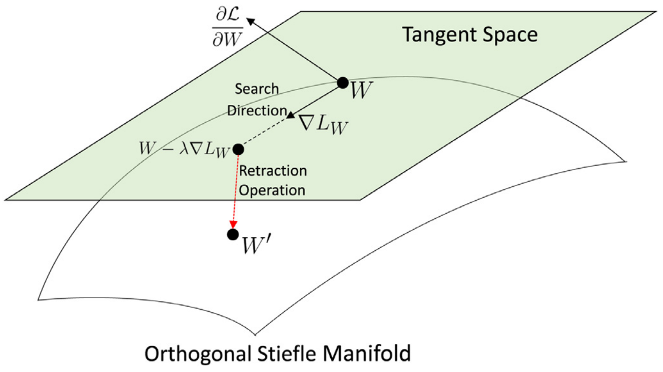

Fig. 2. The illustration of the optimization process of W. W is an original point on the orthogonal Stiefel manifold. W' is a new point after an iterative update. $\frac{\partial \mathcal{L}}{\partial W}$ is the partial derivative of the loss function with respect to W. ∇L_W is the manifold gradient lying on the tangent space.

Since there are learnable parameters in the matrix transformation layer, we should compute the gradient of the loss function $\mathcal{L}$ with respect to the input K and the parameters W. The gradient with respect to the input K is

$$\frac{\partial \mathcal{L}}{\partial K} = W\frac{\partial \mathcal{L}}{\partial Y}W^\top, \tag{14}$$

where $\frac{\partial \mathcal{L}}{\partial Y}$ is the gradient with respect to the output Y.

Since W is a column full rank matrix, it is on a non-compact Stiefel manifold [27]. However, directly optimizing W on the non-compact Stiefel manifold is infeasible. To overcome this issue, we relax W to be semi-orthogonal, *i.e.,* $W^\top W = I_{C'}$. In this case, W is on the orthogonal Stiefel manifold $St(C', C)$ [27]. The optimization space of parameters W is changed from the non-compact Stiefel manifold to the orthogonal Stiefel manifold $St(C', C)$. Considering the manifold structure of W, the optimization process is quite different from the gradient descent method in the Euclidean space. After we compute the partial derivative with respect to W, the partial derivative is a Euclidean gradient and may be towards any directions. As shown in Fig. 2, if we utilize the Euclidean gradient to update W, the new point can not be guaranteed to on the Stiefel manifold. Considering the tangent space is a Euclidean space, and points on the manifold or the tangent space can be transformed into the other via definite projections, we apply the Riemannian optimization methods to update W to guarantee the manifold constraint, which resorts to the tangent space. The optimization method contains three steps as illustrated in Fig. 2. Firstly, we convert the partial derivative to the manifold gradient that lies on the tangent space. In the second step, along with the manifold gradient, we find a new point on the tangent space. Finally, the retraction operation is applied to map the new point on the tangent space back to the orthogonal Stiefel manifold. In this way, an iteration of the optimization process on the manifold is completed. Next, we will elaborate on each step.

The partial derivative $\frac{\partial \mathcal{L}}{\partial W}$ with respect to W is computed by

$$\frac{\partial \mathcal{L}}{\partial W} = K^\top W\frac{\partial \mathcal{L}}{\partial Y} + KW\left(\frac{\partial \mathcal{L}}{\partial Y}\right)^\top. \tag{15}$$

The partial derivative $\frac{\partial \mathcal{L}}{\partial W}$ doesn't contain any manifold constraints. Considering W is a point on the orthogonal Stiefel manifold, the partial derivative needs to be converted to the manifold gradient, which is on the tangent space. On the orthogonal Stiefel manifold, the partial derivative $\frac{\partial \mathcal{L}}{\partial W}$ is a Euclidean gradient at the point W, not tangent to the manifold. The tangential component of $\frac{\partial \mathcal{L}}{\partial W}$ is what we need for optimization. The normal component is perpendicular to the tangent space. We decompose $\frac{\partial \mathcal{L}}{\partial W}$ into two vectors that are perpendicular to each other, *i.e.*, one is tangent to the manifold, and the other is the normal component based on the Theorem 3.

Theorem 3. *Let M denote an orthogonal Stiefel manifold and $\mathbf{X}$ is a point on M. $F(\mathbf{X})$ denotes a function defined on the orthogonal Stiefel manifold. If the partial derivative of F with respect to $\mathbf{X}$ is $F_{\mathbf{X}}$, the manifold gradient ∇F at $\mathbf{X}$ which is tangent to M is $\nabla F = F_{\mathbf{X}} - \mathbf{X} sym(\mathbf{X}^\top F_{\mathbf{X}})$, where $sym(A) = \frac{1}{2}(A + A^\top)$.*

Proof. Because $\mathbf{X}$ is a point on the orthogonal Stiefel manifold, $\mathbf{X}^\top \mathbf{X} = I$, where I is an identity matrix. Differentiating $\mathbf{X}^\top \mathbf{X} = I$ yields

$$\mathbf{X}^\top \Delta_1 + {\Delta_1}^\top \mathbf{X} = 0, \tag{16}$$

where Δ_1 is a tangent vector. Thus, $\mathbf{X}^\top \Delta_1$ is a skew-symmetric matrix. We define the space orthogonal to the tangent space at the manifold point is the normal space. Because a small region of the manifold can be regarded as a approximate Euclidean space, the inner product of the tangent space vector Δ_1 and normal space vector Δ_2 is

$$\langle \Delta_1, \Delta_2 \rangle = tr(\Delta_1, \Delta_2) = 0. \tag{17}$$

Note that, if $\Delta_2 = \mathbf{X}S$, where S is a $p \times p$ symmetric matrix, then Δ_2 is in the normal space. So the normal space of $\mathbf{X}$ can be represented by a set $\{\mathbf{X}S\}$, where S is any $p \times p$ symmetric matrix. Thus, the partial derivative $F_{\mathbf{X}}$ can be projected to the normal space,

$$\pi_N(\mathbf{X}) = \mathbf{X} sym(\mathbf{X}^\top F_{\mathbf{X}}), \tag{18}$$

and $\pi_N(\mathbf{X})$ is the normal component of $F_{\mathbf{X}}$. Thus the manifold gradient which is equal to the tangential component $\pi_T(\mathbf{X})$ at $\mathbf{X}$ can be computed,

$$\nabla F = \pi_T(\mathbf{X}) = F_{\mathbf{X}} - \pi_N(\mathbf{X}) = F_{\mathbf{X}} - \mathbf{X} sym(\mathbf{X}^\top F_{\mathbf{X}}). \tag{19}$$

□

Then the tangential component ∇L_W at W can be expressed by the partial derivative $\frac{\partial \mathcal{L}}{\partial W}$,

$$\nabla L_W = \frac{\partial \mathcal{L}}{\partial W} - W sym\left(W^\top \frac{\partial \mathcal{L}}{\partial W} \right). \tag{20}$$

∇L_W is the manifold gradient of the orthogonal Stiefel manifold. Searching along the gradient ∇L_W on the tangent space gets a new point. Finally, we use the retracting operation to map the point on the tangent space back to the Stiefel manifold space,

$$W := q(W - \lambda \nabla L_W), \tag{21}$$

where $q(\cdot)$ is the retraction operation that mapps the data back to the manifold. $q(A)$ denotes the Q matrix of QR decomposition to A. $A \in \mathbb{R}^{n \times p}$, $A = QR$, where $Q \in \mathbb{R}^{n \times p}$ is a semi-orthogonal matrix and $R \in \mathbb{R}^{p \times p}$ is a upper triangular matrix. λ is the learning rate, respectively.

Note that, we can make a Relu activation function layer follow the matrix transformation layer. The output of the Relu layer is still an SPD matrix based on Dong et al. [17].

3.4. Vector transformation layer

Since the inputs of common tasks are all vectors, we should vectorize the SPD matrix. Because of the symmetry of the SPD matrix Y, Y is determined by $\frac{C' \times (C'+1)}{2}$ elements, *i.e.*, the upper triangular matrix or the lower triangular matrix of Y. Here, we take the upper triangular matrix of Y, multiply the non-diagonal elements of Y by $\sqrt{2}$ and reshape the upper triangular matrix into a vector V for visual tasks,

$$\begin{aligned} V &= \left[Y_{11}, \sqrt{2}Y_{12}, \ldots, \sqrt{2}Y_{1C'}, Y_{22}, \sqrt{2}Y_{23}, \ldots, \sqrt{2}Y_{C'(C'-1)}, Y_{C'C'} \right] \\ &= \left[V_1, V_2, \ldots, V_{\frac{C' \times (C'+1)}{2}} \right]. \end{aligned} \tag{22}$$

Due to the symmetry of the matrix Y, the gradient $\frac{\partial \mathcal{L}}{\partial Y}$ is also a symmetric matrix. For the diagonal elements of Y, their gradients of the loss function are equal to the gradients of their corresponding elements in the vector V. The gradient of non-diagonal elements of Y is $\sqrt{2}$ times to the corresponding elements in the vector V. The gradient with respect to Y is given by

$$\frac{\partial \mathcal{L}}{\partial Y} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial V_1} & \frac{\sqrt{2}\partial \mathcal{L}}{\partial V_2} & \cdots & \frac{\sqrt{2}\partial \mathcal{L}}{\partial V_{C'-1}} & \frac{\sqrt{2}\partial \mathcal{L}}{\partial V_{C'}} \\ \frac{\sqrt{2}\partial \mathcal{L}}{\partial V_2} & \frac{\partial \mathcal{L}}{\partial V_{C'+1}} & \frac{\sqrt{2}\partial \mathcal{L}}{\partial V_{C'+2}} & \cdots & \frac{\sqrt{2}\partial \mathcal{L}}{\partial V_{2\times C'-1}} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \frac{\sqrt{2}\partial \mathcal{L}}{\partial V_{C'}} & \frac{\sqrt{2}\partial \mathcal{L}}{\partial V_{2\times C'-1}} & \frac{\sqrt{2}\partial \mathcal{L}}{\partial V_{3\times C'-3}} & \cdots & \frac{\partial \mathcal{L}}{\partial V_{\frac{C' \times (C'+1)}{2}}} \end{bmatrix}. \tag{23}$$

The normalization operation is important as well. We apply the power normalization ($V_i := sign(V_i)\sqrt{|V_i|}$) and l_2 normalization ($V := V/\|V\|_2$) operations on the vector V in the vector transformation layer. The gradient formulations Eqs. (9), (14) and (21) calculate the gradient with respect to the input of the corresponding layer, respectively. Once these gradients are obtained, the standard Stochastic Gradient Descent (SGD) optimization can be easily employed to update parameters directly with the learning rate. Note that we can use more than one matrix transformation layer in the network, where each one can be followed by a Relu layer as the activation layer.

4. Experiments

To demonstrate the benefits of our method, we conduct extensive experiments on both visual search and classification tasks. Search tasks with CNN can be divided into two categories: using pre-trained CNN or fine-tuned CNN models [28]. Although many visual search tasks are under an end-to-end framework via fine-tuning CNNs and achieve better performance, we implement the search tasks of only post-processing extracted features from a pre-trained network to demonstrate that the SPD representation can achieve good performance in a non-learnable process as well. We present visual search tasks on six datasets and turn to visual classification tasks on seven datasets.

4.1. Visual search

We validate the effectiveness of the proposed SPD aggregation method through the image retrieval task, the object instance search task, and the person re-identification task. Given a specific object as the query, the object instance search aims to not only retrieve images that contain the query but also locate all its occurrences.

4.1.1. Datasets and evaluation protocols

We conduct comprehensive experiments on seven public datasets which are popularly used in the visual search tasks, *i.e.*, Oxford5K [29], Paris6K [30], Sculptures6K [31], Holidays [32], Holidays + 100K [29], UKbench [33], and the Market-1501 [34]. Note that, Holidays + 100K contains the Holidays dataset and additionally 100K distractor images from Flickr. Experiments conducted on the Holidays + 100K dataset is to demonstrate the search ability of our SPD representation for the large-scale scenario. The Holidays, Holidays + 100K, and UKbench datasets are used for image retrieval, Oxford5K, Paris6K, and Sculptures6K are used for the object instance search task. For the Oxford5K, Paris6K, Sculptures6K, and Holidays datasets, the performance is evaluated by mean average precision (mAP). For the UKbench dataset, the performance is reported as the average number of same-object images within top four results, *i.e.*, $4 \times recall@4$. The Market-1501 is a widely used person re-identification dataset. We utilize the first 751 person for training and the rest 750 person for test.

4.1.2. Implementation details

For all dataset images, retrieval images, and query objects images, we resize them to 448×448 and extract convolutional features from the *conv5-3* layer of the VGG-16 model which is pre-trained on the ImageNet dataset. Then the PCA is applied to reduce the dimensionality of each convolutional local feature from 512 to 64. The PCA also makes each feature map more independent. We aggregate these 64 feature maps into an SPD matrix representation by the nonlinear kernel generation layer, and the SPD matrices are mapped to the tangent space by the matrix logarithm $logm(\cdot)$ function, since we exploit the ℓ_2 distance to measure the similarity between representations. The vector transformation layer follows it, and the dimensionality of the vector representation is 2080 $(\frac{64\times(64+1)}{2})$. Considering the speed is important for search tasks, and the 2080-dimensional representation is still large, we perform the PCA once more to the vector representation, in which the dimensionality is reduced to 256. Image retrieval is carried out by calculating the ℓ_2 distance between the obtained 256-dimensional representations of the query image and database images.

For the object instance search, a set of N object proposals $\mathcal{I} = \{p_j\}_{j=1}^{N}$ of each reference image I are extracted by Edge Boxes [35] and serve as potential regions for the query object. Given a query q and a reference image I, we first find the best-matched proposal $\tilde{p}$ of I, and the relevance between q and I is determined by the ℓ_2 distance between q and $\tilde{p}$. The bounding box of $\tilde{p}$ corresponds to the detecting location of the object in the reference image. We then sort the reference images according to $R(q, I)$ and get search results. Besides, following Tolias et al. [5], we utilize the average query expansion (AQE) to further enhance the object instance search performance.

4.1.3. Comparisons with other image retrieval methods

The performance of different image retrieval methods is illustrated in Table 1. Several well-known methods [3,4,6,20,36–45] are evaluated in our experiments. Among these methods, MOP-CNN [20], Neural Codes [36], Razavian [37], VLAD [38], SPoC [4], CroW [6], and Bilinear [3] are feature aggregation schemes, which are non-learnable and post-process extracted features from a pre-trained network. As can be seen, our scheme provides considerable improvement over state-of-the-art feature aggregation schemes on compact representations. The proposed method performs especially well on the Holidays dataset. Even CroW [6] using a tricky pooling method achieves 0.849 mAP on the Holidays dataset with a 512-dimensional representation, a 3.3% improvement in mAP is achieved by our 256-dimensional representation. The dimensionality of VLAD [38] representation is $12,800$, much larger than our method, but the performance is still 4.6% lower than ours. Results show the discriminative power of the proposed SPD aggregation scheme. On the UKbench dataset, the average number of same-object images within top four results of our method is 3.78 compared with 3.65 achieved by the SPoC [4]. CKN [39], NetVLAD [21], R-MAC [40], and Gordo et al. [41,42] are end-to-end frameworks. They finetune the network to be adaptive to the retrieval dataset. Even so, with the same VGG-16 network structure and experimental configuration, our method achieves better performance than NetVLAD, R-MAC and Gordo et al. Our method gets 88.2 on the Holidays dataset, 5.1%, 6.8%, and 1.5% higher than the three methods, respectively. BoW+SN [43] mines similar neighbours to replace the original point in the retrieval process considering more context-sensitive information, while our method only uses the original point. RED [44] and RDP [45] utilize the diffusion machine to compute similarities. RED [44] and RDP [45] are based on the ResNet that is more powerful. Besides, the three works and our method focus on different aspects of the retrieval task. They focus on the similarity computing, while we focus a more robust representation. Moreover, the dimensionality of our representation is only 256, lower than most methods, which can reduce the consume of retrieval time and storage.

4.1.4. Comparisons with other object instance search methods

For the object instance search task, Table 2 shows the performance compared with state-of-the-art methods [4–6,21,36–38,40–42,45–48]. Without the AQE scheme, the proposed method obtains 82.2% mAP on the Oxford5Kd dataset, 90.0% on the Paris6K dataset and 57.0% on the Sculptures6K dataset. Compared with existing feature aggregation methods [4–6,36,38,46] that only post-process extracted features from a pre-trained network, our method brings a great improvement. Since methods in [4,36,38] ignore the object localization process which is advantageous to the visual search performance, they are not competitive compared with other methods. The mAP of Razavian et al. [37] is 84.4% on the Oxford5Kd dataset, 2.2% higher than ours. On the Paris6K dataset, the mAP of [37] is 85.3%, while our method achieves 90.0%. Besides, the method in [37] enlarges the query object bounding box which is not compliant with the standard evaluation protocol. With the

Table 1
Image retrieval performance comparisons with state-of-the-art methods on the Holidays, Holidays+100K and UKbench datasets. The results are reported by mAP (%) and $4 \times recall@4$.

Method	Network	Dimensionality	Holidays	Holidays+100K	UKbench
MOP-CNN [20]	CaffeNet	512	78.3	–	–
Neural Codes [36]	–	256	78.9	–	3.56
Razavian [37]	AlexNet	256	74.2	–	3.54
VLAD [38]	GoogLeNet	12800	83.6	–	–
SPoC [4]	VGG-19	256	80.2	–	3.65
CroW [6]	VGG-16	512	84.9	–	–
CroW [6]	VGG-16	256	83.1	–	–
Bilinear [3]	VGG-16	256	79.1	71.3	3.48
CKN[39]	AlexNet	4096	82.9	–	–
NetVLAD [21]	VGG-16	4096	83.1	–	–
R-MAC [40]	VGG-16	256	81.4	69.4	–
Gordo et al. [41]	VGG-16	512	86.7	–	–
Gordo et al. [42]	ResNet101	2048	90.3	–	–
BoW+SN [43]	Alexnet	–	–	–	3.98
RED [44]	ResNet50	2048	93.3	–	3.94
RDP [45]	ResNet50	2048	95.7	–	3.93
Ours	VGG-16	256	**88.2**	**79.6**	**3.78**

Table 2
Object instance search performance comparisons with state-of-the-art methods on the Oxford5K, Paris6K and Sculptures6K datasets in terms of mAP (%).

Method	Network	Dimensionality	Oxford5K	Paris6K	Sculptures6K
Neural Codes [36]	–	256	54.5	–	–
SPoC [4]	VGG-19	256	65.7	64.4	–
VLAD [38]	VGG-16	12800	64.9	69.4	–
CNNaug-ss [46]	AlexNet	$4-15k$	68.0	79.5	42.3
Razavian et al. [37]	AlexNet	1024×32	84.4	85.3	67.4
CroW + AQE [6]	VGG-16	256	69.2	78.5	–
Tolias et al. [5]	VGG-16	512	77.3	86.5	–
NetVLAD [21]	VGG-16	4096	71.6	79.7	–
MAC + R + AQE [40]	VGG-16	512	85.4	87.0	–
Gordo et al. + AQE [41]	VGG-16	512	0.891	0.912	–
Gordo et al. + AQE [42]	ResNet101	2048	90.6	96.0	–
Gordo et al. + AQE + DBA [42]	ResNet101	2048	94.7	96.6	–
Noh et al. + AQE [47]	ResNet50	2048	90.0	95.7	–
R-Match + AQE [48]	ResNet101	2048×21	91.0	95.5	–
RDP + AQE [45]	ResNet50	2048	95.3	–	–
Ours	VGG-16	256	**82.2**	**90.0**	**57.0**
Ours + AQE	VGG-16	256	**88.9**	**94.0**	**75.3**

Table 3
Person re-identification performance on the Market-1501 dataset.

Method	Network	Rank-1	Rank-5	Rank-10	mAP
Cheng et al. [49]	ResNet50	72.3	86.4	90.6	46.8
CRAFT-MFA [50]	AlexNet	73.8	–	–	47.1
Lin et al. [51]	DGD	73.8	–	–	47.1
LSRO [52]	ResNet50	78.6	–	–	56.2
SVDNet [53]	ResNet50	82.3	92.3	95.2	62.1
PSE [54]	ResNet50	87.7	94.5	96.8	69.0
SPReID [55]	Inception-V3	93.7	97.6	98.4	83.4
Wang et al. [56]	–	89.5	–	–	74.1
Suh et al. [57]	Inception-V1	90.2	96.1	97.4	76.0
Ours	VGG-16	97.5	99.3	99.5	58.7

AQE, the proposed method obtains 88.9% mAP on the Oxford5Kd dataset, 94.0% on the Paris6K dataset and 75.3% on the Sculptures6K dataset. [21,40–42,47] are end-to-end methods with the same VGG-16 network, while the proposed method is comparable with them, even has better performance than [21,40]. Gordo et al. + AQE [41] and Noh et al. + AQE [47] are under the end-to-end framework and finetune the network, which learn to select local features and make local features more robust instead of a simple Edge box. R-Match + AQE [48] and RDP + AQE [45] focus on the diffusion machine. We focus on a discriminative representation. The two works and our method focus on different aspects of the search tasks. Besides, Bai and Co-authors [45,47,48] use the ResNet as the backbone, and their representation dimensionality is much higher than ours, which are more powerful yet cause the burden of computation and storage.

4.1.5. Comparisons with other person re-identification methods

We conduct experiments with state-of-the-art person re-identification works on the Market-1501 dataset, and the results are shown in Table 3. The main difference between competitive methods [49–57] and our method is that the methods in [49–57] model features in the Euclidean space, while our method models features on the SPD manifold. We can see that our method achieves the best performance in the top-rank evaluation, showing that our representation has robust power to search similar images. We achieve 97.5% on the Rank-1 evaluation, and nearly 100% on the rank-5 and rank-10 evaluations. SPReID [55], Wang et al. [56], and Suh et al. [57] achieve state-of-the-art performance on the top-rank evaluation, 93.7%, 89.5% and 90.2% on rank-1, respectively. The representations in [55–57] are guided by body part weights to overcome the background clutter and pose variation issues. While our method achieves 97.5%, a significant improvement, even our method aims to learn a general representation not focusing on person re-identification challenging issues. For this reason, our mAP evaluation is not very good, as several probe images have large differences from the query image in postures and clothing which are the specific person re-identification challenges.

4.2. Visual classification

In order to demonstrate benefits of the proposed SPD aggregation method in the end-to-end framework, we conduct experiments on the visual classification tasks.

4.2.1. Datasets and evaluation protocols

We choose seven datasets in experiments. Describable Textures Dataset (DTD) [58], Flickr Material Database (FMD) [59], KTH-TIPS-2b (KTH-2b) [60], and Material In Context (MINC-2500) [61] are texture classification datasets. MIT-Indoor (MIT) [62] is the indoor scene understanding dataset. CUB-200-2011 [63] and FGVC-aircraft [64] are the fine-grained classification datasets. In our experiment, the standard train-test protocols [3,25,65] are used.

4.2.2. Implementation details

The basic convolutional layers and pooling layers before our SPD aggregation are from the VGG-16 model which is pre-trained on the ImageNet dataset. We remove layers after the *conv5-3* layer of the VGG-16 model. Then we insert our SPD aggregation method into the network following the *conv5-3* layer. Finally, an FC layer and a softmax layer follow the vector transformation layer where the output dimensionality of the FC layer is equal to the number of categories. We use the SGD optimization with a mini-batch size of 32. The training process is divided into two stages. At the first training stage, we fix the parameters before the SPD aggregation and train the rest new layers. The learning rate starts at 0.1 and reduced by 10 when error plateaus. At the second training stage, we train the whole network. The learning rate starts at 0.001 and divided by 10 when error plateaus.

4.2.3. Comparisons with feature aggregation methods in the Euclidean space

In this section, we compare the proposed SPD aggregation framework with several Euclidean Space convolutional feature aggregation methods in the texture and fine-grained image classification tasks, where the image size is resized to 224×224. First

Table 4
Comparisons for CNNs based methods on the FMD, KTH-T2b, CUB-200-2011, and FGVC-aircraft datasets in terms of Average Precision (%). Our method is bold in the last line.

Method	FMD	KTH-2b	CUB-200-2011	FGVC-aircraft
FV-CNN [22]	73.5	73.3	49.9	–
FV-FC-CNN [22]	76.4	73.8	54.9	–
B-CNN [3]	77.8	79.7	74.0	74.3
Deep-TEN$_{ResNet50}$ [66]	80.2	82.0	–	–
VGG-16 [1]	77.8	78.3	68.0	75.0
Ours	**79.2**	**81.1**	**72.4**	**77.8**

a 1×1 convolutional layer whose number of channels is 512 follows the *conv5-3* convolutional layer. Then our SPD aggregation method including a kernel generation layer, a matrix transformation layer, and a vector transformation layer is inserted after the 1×1 convolutional layer. The output size of the matrix transformation layer is 512×512, and the output size of the vector transformation layer is 131328 ($\frac{512\times(512+1)}{2}$). Considering these datasets are not big enough, we only use one matrix transformation layer to avoid overfitting. Table 4 shows the comparisons on texture datasets and fine-grained datasets.

The following methods are evaluated in our experiments, *i.e.,* FV-CNN [22], FV-FC-CNN [22], B-CNN [3], and Deep-TEN [66]. The pure VGG-16 model [1] is used as the baseline. FV-CNN aggregates convolutional features from the VGG-16 *conv5-3* layer. The dimensionality of FV-CNN is 65600 and is compressed to 4096 by the PCA for classification. FV-FC-CNN incorporates the FC features and FV vector. B-CNN uses the Bilinear pooling method on the *conv5-3* layer of the VGG-16 model. The Deep-TEN uses 50-layers ResNet, larger number of training samples, and image size, while the other methods use VGG-16 model. Thus it is not scientific to compare Deep-TEN with the other feature aggregation methods.

Compared with the traditional convolutional feature aggregation method FV and the pure VGG-16 model, our method achieves large improvements, showing the powerful representation ability. In addition, we can see that, our method gets better performance compared with B-CNN, especially on the KTH-2b and FGVC-aircraft datasets. The average precision of our method on the KTH-2b and FGVC-aircraft datasets are 81.1% and 77.8%, respectively. In contrast, B-CNN achieves 79.7% and 74.3%. On the CUB-200-2011 dataset, we also get the comparable result.

4.2.4. Comparisons with feature aggregation methods in the Riemannian space

In this subsection, we conduct experiments to compare our method with other SPD matrix representation methods. Three datasets are used: the DTD, MIT, and MINC-2500 datasets, whose images are resized to 448×448. Following the experiments in [25], we add a Batch Normalization layer before the softmax layer to all methods.

The following SPD representation methods are evaluated in this experiment, B-CNN [3], MPN [13], DeepO2P [14], CBP [7], LRBP [24], and SMSO [25]. We utilize the pure VGG-16 [1] as the baseline method. The accuracies are shown in Table 5. We can find that, our method achieves the state-of-the-art performance on the three datasets, 70.9% on the DTD dataset, 79.9% on the MIT dataset, and 83.2% on the MINC-2500 dataset. All SPD matrix representation methods lead to better performance than the baseline VGG-16 model. The proposed method can reduce the dimensionality of the representation like CBP and LRBP, but our representation is more discriminative with a significant improvement. SMSO and our method improve a significant margin than other methods, showing the effectiveness of a learnable mapping operation in the network. Although SMSO is the best method on the three datasets, our method consistently has better performance than it, especially on the MINC-2500 dataset, 5.2% higher than it. The experiments show the power of the proposed kernel matrix representation and transformation process.

Table 5
Comparisons for Riemannian manifold representation methods on the DTD, MIT, and MINC-2500 datasets in terms of Average Precision (%). Our method is bold in the last line.

Method	DTD	MIT	MINC-2500
VGG-16 [1]	60.1	64.5	73.0
B-CNN [3]	67.5	77.6	74.5
MPN [13]	68.0	76.5	76.2
DeepO2P [14]	66.1	72.4	69.3
CBP-TS [7]	67.7	76.8	73.3
CBP-RM [7]	63.2	73.9	73.5
LRBP [24]	65.8	73.6	69.0
SMSO [25]	69.3	79.5	78.0
Ours	**70.9**	**79.9**	**83.2**

4.3. Dimension experiments

Here, we conduct dimension experiments on the image search and image classification tasks. For the image search tasks, we evaluate different dimensions of the representation after PCA, and the results are shown in Table 6. For the image classification tasks, as it is an end-to-end framework, we evaluate different dimensions of the output channel of the 1×1 preprocessing convolutional layer and the output matrix size of the matrix transformation layer. The results are shown in Table 7.

Through the dimension experiments on the image search tasks, we find that dimensionalities have different effects on different datasets. In the large-scale dataset, *e.g.,* the Holidays+100K dataset, a high-dimensional representation achieves better performance, since a sample needs to be distinguished from a large number of other samples. In other datasets which contain moderate numbers of samples, an appropriate dimensionality is needed. Our representation is a high-order statistic and preserves very detailed information. For these moderate datasets, an SPD matrix is redundant. Thus, for the not very large image search tasks, an appropriate dimensionality reduction and a compact representation is feasible.

In the classification experiments, we find the high-dimensionality representation (the outputs of the two evaluated layers are both 512) achieves the best performance on both two datasets, and a low-dimensional representation (*e.g.,* 256 and 128) is not enough for the two classification tasks. We conclude that our method is not over-fitting on the two datasets, and a high-dimensionality representation is helpful to memory information of multiple categories. Maintaining the same dimension as the backbone can exploit the pre-trained network efficiently and avoid information loss. Besides, the dimensionality reduction on the matrix transformation layer is more likely to lead to worse results, especially on the MINC-2500 dataset.

4.4. Ablation experiments

In this subsection, we investigate different contributions of the proposed components. We evaluate the 1×1 preprocessing convolutional layer and the three proposed layers consecutively using the end-to-end framework on the classification task. The experiments are conducted on two datasets: the DTD and MINC-2500 datasets. The analyses are presented as follows, and the results are shown in Table 8. In Table 8, “Conv”, “Gener”, “Mtrans”, “VTri”, and “VNorm” mean that the network has the 1×1 preprocessing convolutional layer, the kernel generation layer, the matrix transformation layer, the upper triangle operation of the vector transfor-

Table 6
Dimension experiments on the search tasks.

Dimensionality	Holidays	Holidays+100K	UKbench	Oxford5K	Paris6K	Sculptures6K
1024	73.3	85.2	3.80	85.7	83.9	55.9
512	86.9	83.4	3.81	85.4	87.7	57.4
256	88.2	79.6	3.78	82.2	90.0	57.0
128	85.9	73.3	3.71	78.4	87.1	48.4

Table 7
Dimension experiments on the image classification tasks.

Method	DTD	MINC-2500
Conv512+Trans512	70.9	83.2
Conv512+Trans256	69.6	82.1
Conv512+Trans128	68.6	78.7
Conv256+Trans256	69.1	82.2
Conv256+Trans128	68.4	81.2
Conv128+Trans128	67.9	79.7

Table 8
Ablation experiments on the image classification tasks.

Method	DTD	MINC-2500
Conv+Gener	68.1	81.0
Conv+Gener+MTrans	69.0	81.5
Conv+Gener+MTrans+VTri	69.6	81.7
Conv+Gener+VTri+VNorm	70.4	81.4
Gener+MTrans+VTri+VNorm	71.0	82.9
Conv+Gener+MTrans+VTri+VNorm	70.9	83.2

mation layer, and the normalization operation of the vector transformation layer, respectively.

The ablation experiments show that all components have contributions to the proposed aggregation method. Using a 1×1 convolutional layer and a kernel generation layer, our method is better than the linear aggregation method B-CNN, showing the discriminative power of the kernel matrix with non-linear statistic information and the nonsingularity. When we compare “Conv+Gener+MTrans+VTri+VNorm” and “Gener+MTrans+VTri+VNorm”, the preprocessing of convolutional features makes features more adaptive on the MINC-2500 dataset. Without the matrix transformation layer, *i.e.,* “Conv+Gener+VTri+VNorm”, the performance drops 0.5% and 1.8% on the two datasets, respectively. The vector transformation layer which includes the upper triangle operation and the normalization operation also has the significance, achieving 1.9% and 1.7% improvement on the two datasets. Through the ablation experiment, we find that these components play different roles on our method, and using all components achieves the best performance.

5. Conclusion and discussion

In this paper, we have proposed a new powerful feature aggregation method in an end-to-end framework which captures discriminative feature distribution via modeling convolutional features on an SPD manifold. Through theoretical analyses, our method can measure nonlinear complex relationships of convolutional features and guarantee the robust SPD manifold structure, where each component of our method has its own contribution. Our feature aggregation method has a wide range of applications, since most pattern recognition and computer vision tasks need robust representations. Via the image retrieval, object instance search, person re-identification, and fine-grained and texture image classification experiments, the discriminative power of our representation is demonstrated. However, there still exists two weaknesses. Our method still uses the softmax classifier, which is a general probability distribution classifier not a manifold-specific classifier. Besides, due to the QR decomposition, the optimization of the manifold is time-consuming.

Based on this work, we think there are three interesting directions for future works: (1) How to design a special classifier to be adaptive to SPD manifold properties. Maybe a similarity-based classifier is feasible, as there exist several true metrics which can consider the manifold geometry. (2) How to avoid the complex manifold optimization. Inspired by Huang et al. [67], learning a discriminative tangent space instead of a manifold may be a good idea. (3) Besides the SPD manifold, there exist several other commonly used manifold representations, such as the subspace on the Grassmannian manifold. How to build end-to-end frameworks on these manifolds is also worth doing.

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] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv:1409.1556 (2014).
[2] 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.
[3] 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.
[4] A.B. Yandex, V. Lempitsky, Aggregating local deep features for image retrieval, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2016, pp. 1269–1277.
[5] 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.
[6] Y. Kalantidis, C. Mellina, S. Osindero, Cross-dimensional weighting for aggregated deep convolutional features, in: Proceedings of the European Conference on Computer Vision (ECCV), 2016, pp. 685–701.
[7] Y. Gao, O. Beijbom, N. Zhang, T. Darrell, Compact bilinear pooling, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 317–326.
[8] Z. Huang, R. Wang, S. Shan, X. Chen, Face recognition on large-scale video in the wild with hybrid Euclidean-and-Riemannian metric learning, Pattern Recognit. 48 (2015) 3113–3124.
[9] W. Hariri, H. Tabia, N. Farah, A. Benouareth, D. Declercq, 3d face recognition using covariance based descriptors, Pattern Recognit. Lett. 78 (2016) 1–7.
[10] J. Zhang, L. Zhou, L. Wang, Subject-adaptive integration of multiple sice brain networks with different sparsity, Pattern Recognit. 63 (2016) 642–652.
[11] H. Laanaya, F. Abdallah, H. Snoussi, C. Richard, Learning general gaussian kernel hyperparameters of SVMs using optimization on symmetric positive-definite matrices manifold, Pattern Recognit. Lett. 32 (2011) 1511–1515.
[12] M. Faraki, M.T. Harandi, F. Porikli, No fuss metric learning, a hilbert space scenario, Pattern Recognit. Lett. 98 (2017) 83–89.
[13] P. Li, J. Xie, Q. Wang, W. Zuo, Is second-order information helpful for large-scale visual recognition? in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017, pp. 2070–2078.

[14] C. Ionescu, O. Vantzos, C. Sminchisescu, Matrix backpropagation for deep networks with structured layers, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 2965–2973.
[15] K. Yu, M. Salzmann, Second-order convolutional neural networks, Clin. Immunol. Immunopathol. 66 (2017) 230–238.
[16] L. Wang, J. Zhang, L. Zhou, C. Tang, Beyond covariance: feature representation with nonlinear kernel matrices, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 4570–4578.
[17] Z. Dong, S. Jia, C. Zhang, M. Pei, Y. Wu, Deep manifold learning of symmetric positive definite matrices with application to face recognition, in: Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI), 2017, pp. 4009–4015.
[18] Z. Huang, L.J. Van Gool, A Riemannian network for SPD matrix learning, in: Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI), 2017, pp. 2036–2042.
[19] M. Harandi, M. Salzmann, R. Hartley, Dimensionality reduction on SPD manifolds: the emergence of geometry-aware methods, IEEE Trans. Pattern Anal. Mach. Intell. 40 (2018) 48–62.
[20] 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), 2014, pp. 392–407.
[21] 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.
[22] 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.
[23] M. Faraki, M.T. Harandi, A. Wiliem, B.C. Lovell, Fisher tensors for classifying human epithelial cells, Pattern Recognit. 47 (2014) 2348–2359.
[24] S. Kong, C. Fowlkes, Low-rank bilinear pooling for fine-grained classification, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 7025–7034.
[25] K. Yu, M. Salzmann, Statistically-motivated second-order pooling, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 600–616.
[26] M. Engin, L. Wang, L. Zhou, X. Liu, DeepKSPD: learning kernel-matrix-based SPD representation for fine-grained image recognition, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 612–627.
[27] P.A. Absil, R. Mahony, R. Sepulchre, Optimization Algorithms on Matrix Manifolds, 2009.
[28] L. Zheng, Y. Yang, Q. Tian, Sift meets CNN: a decade survey of instance retrieval, IEEE Trans. Pattern Anal. Mach. Intell. 40 (2018) 1224–1244.
[29] J. Philbin, O. Chum, M. Isard, J. Sivic, A. Zisserman, Object retrieval with large vocabularies and fast spatial matching, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2007, pp. 1–8.
[30] J. Philbin, O. Chum, M. Isard, J. Sivic, Lost in quantization: improving particular object retrieval in large scale image databases, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2008, pp. 1–8.
[31] R. Arandjelovic, A. Zisserman, Smooth object retrieval using a bag of boundaries, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2011, pp. 375–382.
[32] H. Jegou, M. Douze, C. Schmid, Hamming embedding and weak geometric consistency for large scale image search, in: Proceedings of the European Conference on Computer Vision (ECCV), 2008, pp. 304–317.
[33] D. Nister, H. Stewenius, Scalable recognition with a vocabulary tree, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2006, pp. 2161–2168.
[34] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, Q. Tian, Scalable person re-identification: a benchmark, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2015, pp. 1116–1124.
[35] C.L. Zitnick, P. Dollár, Edge boxes: locating object proposals from edges, in: Proceedings of the European Conference on Computer Vision (ECCV), 2014, pp. 391–405.
[36] A. Babenko, A. Slesarev, A. Chigorin, V. Lempitsky, Neural codes for image retrieval, in: Proceedings of the European Conference on Computer Vision (ECCV), 2014, pp. 584–599.
[37] A.S. Razavian, J. Sullivan, S. Carlsson, A. Maki, Visual instance retrieval with deep convolutional networks, IEEE Trans. Media Technol.Appl. 4 (2016) 251–258.
[38] Y.H. Ng, F. Yang, L.S. 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.
[39] M. Paulin, J. Mairal, M. Douze, Z. Harchaoui, F. Perronnin, C. Schmid, Convolutional patch representations for image retrieval: an unsupervised approach, Int. J. Comput. Vis. 121 (2017) 149–168.
[40] F. Radenovi, G. Tolias, O. Chum, CNN image retrieval learns from bow: Unsupervised fine-tuning with hard examples, in: Proceedings of the European Conference on Computer Vision (ECCV), 2016, pp. 3–20.
[41] A. Gordo, J. Almazn, J. Revaud, D. Larlus, Deep image retrieval: learning global representations for image search, in: Proceedings of the European Conference on Computer Vision (ECCV), 2016, pp. 241–257.
[42] A. Gordo, J. Almazan, J. Revaud, D. Larlus, End-to-end learning of deep visual representations for image retrieval, Int. J. Comput. Vis. 124 (2017) 237–254.
[43] S. Bai, S. Sun, X. Bai, Z. Zhang, Q. Tian, Improving context-sensitive similarity via smooth neighborhood for object retrieval, Pattern Recognit. 83 (2018) 353–364.
[44] S. Bai, Z. Zhou, J. Wang, X. Bai, L.J. Latecki, Q. Tian, Ensemble diffusion for retrieval, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 774–783.
[45] S. Bai, X. Bai, Q. Tian, L.J. Latecki, Regularized diffusion process on bidirectional context for object retrieval, IEEE Trans. Pattern Anal. Mach. Intell. (99) (2018) 1.
[46] A.S. Razavian, H. Azizpour, J. Sullivan, S. Carlsson, Cnn features off-the-shelf: an astounding baseline for recognition, in: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2014, pp. 512–519.
[47] H. Noh, A. Araujo, J. Sim, T. Weyand, B. Han, Largescale image retrieval with attentive deep local features, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017, pp. 3456–3465.
[48] A. Iscen, G. Tolias, Y. Avrithis, T. Furon, O. Chum, Efficient diffusion on region manifolds: recovering small objects with compact CNN representations, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2077–2086.
[49] D. Cheng, Y. Gong, X. Chang, W. Shi, A. Hauptmann, N. Zheng, Deep feature learning via structured graph Laplacian embedding for person re-identification, Pattern Recognit. 82 (2017) 94–104.
[50] Y.C. Chen, X. Zhu, W.S. Zheng, J.H. Lai, Person re-identification by camera correlation aware feature augmentation, IEEE Trans. Pattern Anal. Mach. Intell. 40 (2017) 392–408.
[51] J. Lin, L. Ren, J. Lu, J. Feng, J. Zhou, Consistent-aware deep learning for person re-identification in a camera network, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 3396–3405.
[52] Z. Zheng, L. Zheng, Y. Yang, Unlabeled samples generated by gan improve the person re-identification baseline in vitro, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 3774–3782.
[53] Y. Sun, L. Zheng, W. Deng, S. Wang, Svdnet for pedestrian retrieval, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 3820–3828.
[54] M. Saquib Sarfraz, A. Schumann, A. Eberle, R. Stiefelhagen, A pose-sensitive embedding for person re-identification with expanded cross neighborhood re-ranking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 420–429.
[55] M.M. Kalayeh, E. Basaran, M. Gokmen, M.E. Kamasak, M. Shah, Human semantic parsing for person re-identification, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 1062–1071.
[56] J. Wang, S. Zhou, J. Wang, Q. Hou, Deep ranking model by large adaptive margin learning for person re-identification, Pattern Recognit. 74 (2017) 241–252.
[57] Y. Suh, J. Wang, S. Tang, T. Mei, K. Mu Lee, Part-aligned bilinear representations for person re-identification, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018.
[58] 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.
[59] L. Sharan, C. Liu, R. Rosenholtz, E.H. Adelson, Recognizing materials using perceptually inspired features, Int. J. Comput. Vis. 103 (2013) 348–371.
[60] B. Caputo, E. Hayman, P. Mallikarjuna, Class-specific material categorisation, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2005, pp. 1597–1604.
[61] S. Bell, P. Upchurch, N. Snavely, K. Bala, Material recognition in the wild with the materials in context database (supplemental material), in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 413–420.
[62] A. Quattoni, A. Torralba, Recognizing indoor scenes, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009, pp. 413–420.
[63] L. Xie, Q. Tian, R. Hong, S. Yan, B. Zhang, Hierarchical part matching for fine-grained visual categorization, in: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2013, pp. 1641–1648.
[64] S. Maji, E. Rahtu, J. Kannala, M. Blaschko, A. Vedaldi, Fine-grained visual classification of aircraft, arXiv:1306.5151 (2013).
[65] 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.
[66] 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.
[67] Z. Huang, R. Wang, S. Shan, X. Li, X. Chen, Log-euclidean metric learning on symmetric positive definite manifold with application to image set classification, in: Proceedings of the International Conference on Machine Learning (ICML), 33, 2015, pp. 720–729.

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, computer vision, and Riemannian computation.

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 Rapid-Rich Object Search (ROSE) Lab, School of Electrical & Electronic Engineering (EEE), 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).

Xingyuan Bu received the B.S. degree in computer science from Beijing Institute of Technology (BIT), Beijing, China, in 2015. Currently he is a M.S. in the Beijing Laboratory of Intelligent Information Technology, BIT, Beijing, China. His research interests include deep learning, pattern recognition and machine learning, specifically in the areas of image classification and image understanding.

Tan Yu is currently pursuing his Ph.D. at Nanyang Technological University, Singapore. He received his master degree from University of California, Riverside and his bachelor degree from University of Electronic Science and Technology of China. His research interests focus on video indexing and retrieval.

Junsong Yuan (M'08-SM'14) received Ph.D. from Northwestern University in 2009 and M.Eng. from National University of Singapore in 2005. Before that, he graduated from the Special Class for the Gifted Young of Huazhong University of Science and Technology (HUST), Wuhan, China, in 2002. He is currently an associate professor at Department of Computer Science and Engineering (CSE), the State University of New York, Buffalo, USA. Before that he was an associate professor at School of Electrical and Electronics Engineering (EEE), Nanyang Technological University (NTU), Singapore. He received 2016 Best Paper Award from IEEE Trans. on Multimedia, Doctoral Spotlight Award from IEEE Conf. on Computer Vision and Pattern Recognition (CVPR'09), Nanyang Assistant Professorship from NTU, and Outstanding EECS Ph.D. Thesis award from Northwestern University. He is currently Senior Area Editor of Journal of Visual Communication and Image Representation (JVCI), Associate Editor of IEEE Trans. on Image Processing (T-IP), IEEE Trans. on Circuits and Systems for Video Technology (T-CSVT), and served as Guest Editor of International Journal of Computer Vision (IJCV). He is Program Co-chair of ICME'18 and VCIP'15, and Area Chair of ACM MM'18, ACCV'18'14, ICPR'18'16, CVPR'17, ICIP'18'17 etc.

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.