DeepKSPD: Learning Kernel-matrix-based SPD Representation for Fine-grained Image Recognition

Melih Engin[1][0000−0001−5074−8533], Lei Wang[1][0000−0002−0961−0441], Luping Zhou[2,1][0000−0003−1065−6604], and Xinwang Liu[3][0000−0001−9066−1475]

[1] School of Computing and Information Technology University of Wollongong Wollongong, NSW 2500, Australia
[2] School of Electrical and Information Engineering of the University of Sydney, NSW, 2006
[3] School of Computer National University of Defense Technology Changsha, Hunan 410073, China
me648@uowmail.edu.au; leiw@uow.edu.au; luping.zhou@sydney.edu.au; xinwangliu@nudt.edu.cn

Abstract. As a second-order pooled representation, covariance matrix has attracted much attention in visual recognition, and some pioneering works have recently integrated it into deep learning. A recent study shows that kernel matrix works considerably better than covariance matrix for this kind of representation, by modeling the higher-order, nonlinear relationship among pooled visual descriptors. Nevertheless, in that study neither the descriptors nor the kernel matrix is deeply learned. Worse, they are considered separately, hindering the pursuit of an optimal representation. To improve this situation, this work designs a deep network that jointly learns local descriptors and kernel-matrix-based pooled representation in an end-to-end manner. The derivatives for the mapping from a local descriptor set to this representation are derived to carry out backpropagation. More importantly, we introduce the Daleckiĭ-Kreĭn formula from Operator theory to give a concise and unified result on differentiating general functions defined on symmetric positive-definite (SPD) matrix, which shows its better numerical stability in conducting backpropagation compared with the existing method when handling the Riemannian geometry of SPD matrix. Experiments on fine-grained image benchmark datasets not only show the superiority of kernel-matrix-based SPD representation with deep local descriptors, but also verify the advantage of the proposed deep network in pursuing better SPD representations. Also, ablation study is provided to explain why and from where these improvements are attained.

Keywords: Kernel matrix, SPD representation, Deep learning, Fine-grained image recognition.

Lei Wang is the corresponding author.

1 Introduction

To deal with image variations, modern visual recognition usually models the appearance of an image by a set of local descriptors. They evolve from early filter bank responses, through traditional local invariant features, to the activation feature maps of recent deep convolutional neural networks (CNNs). During the course, how to pool a set of local descriptors to obtain a global image representation has been a central issue, and many excellent methods have been proposed in the literature [1–4]. In the past few years, pooling a set of descriptors with covariance matrix has attracted increasing attention and shown promising results in object recognition [5], image set classification [6], and so on. It characterizes the pairwise correlation of descriptor components, and is generally called symmetric positive-definite (SPD) representation since covariance matrix is SPD. Also, this inspires the research on classification, clustering, and dimension reduction with respect to SPD representations [7–9]. In particular, several recent pioneering works integrate this covariance-matrix-based SPD representation into deep CNNs to jointly learn the covariance matrix with local visual descriptors. These works investigates multiple important issues on this deep learning framework, including the derivation of some matrix-based functions for backpropagation, the proper way to normalise covariance matrix, the help of second-order information to large-scale visual recognition, and so on. Together, they further demonstrate the great potential of this kind of representation [10–14].

The above works focus on covariance-matrix-based SPD representation. A recent progress on SPD representation is to model the nonlinear information in a set of descriptors [15–17]. Particularly, the work in [16] directly uses a kernel matrix to represent a descriptor set demonstrates its superiority. Given a set of d-dimensional descriptors, a $d \times d$ kernel matrix is computed with a predefined kernel function, where each entry is the kernel value between the realization of two descriptor components in this set. This method effectively models the nonlinear correlation among these descriptor components. The kernel function can be flexibly chosen to extract various nonlinear relationship, and the covariance matrix corresponds to the special case using a linear kernel. The resulting kernel-matrix-based SPD representation maintains the same size as its covariance-matrix-based counterpart, but produces considerable improvement on recognition performance.

Upon the existing literature, this work further improves the research on SPD representation from the following aspects. Firstly, different from its covariance counterpart, the kernel-matrix-based SPD representation in [16] has neither been developed upon deep local descriptors (instead, traditional descriptors like pixel intensities or Gabor filter responses are used only) nor been jointed learned with these descriptors via a deep learning framework. As a result, its potential has not been sufficiently explored for image recognition. The separated consideration of local descriptors and the kernel matrix in [16] prevents them from effectively negotiating with each other to obtain an optimal SPD representation for the ultimate goal of classification. Secondly, the incorporation of SPD representation, be it covariance-matrix-based or kernel-matrix-based, into deep networks

complicates the backpropagation process. Also, to make the resulting SPD representation better work with the classifier, a matrix logarithm is usually employed to map the kernel matrix from Riemannian geometry to Euclidean geometry. Sometimes, matrix square rooting has also been used for this purpose. In the literature, the seminal work in [18] develops the backpropagation algorithm for matrix logarithm from the scratch. Although instructive and informative, it has been reported in the literature that this matrix backpropagation could have numerical stability issue when used to train the deep networks and some remedy has to be developed instead [12].

To address the first issue, this work builds the kernel-matrix-based SPD representation upon deep local descriptors and benchmarks it against the state-of-the-art image recognition methods. Moreover, we develop a deep network called DeepKSPD to jointly learn the deep local descriptors and the kernel-matrix-based SPD representation in an end-to-end training manner. Particularly, for the proposed DeepKSPD network, we highlight the layers designed to be different form the existing deep networks on covariance-matrix-based SPD representation and explain the necessity of these layers.

To address the second issue, we introduce the Daleckiĭ-Kreĭn formula in Operator theory [19, 20] to the computer vision community, and utilise it to derive all the matrix derivatives involved in the mapping from a local descriptor set to the kernel-matrix-based SPD representation to fulfill the backpropagation algorithm for the proposed deep network. As shown, the Daleckiĭ-Kreĭn formula can provide us a more concise and unified result on the gradients of the functions on SPD matrices, regardless of whether matrix logarithm or matrix α-rooting are used as the normalisation method. We give theoretical proof to illustrate the relationship of this formula to the matrix backpropagation work [18], and show the discrepancy that leads to the numerical stability issue of [18].

Experimental study is conducted on multiple benchmark datasets, especially on fine-grained image recognition, to demonstrate the efficacy of the proposed DeepKSPD framework. Firstly, in contrast to the existing kernel-matrix-based representation built upon traditional local descriptors, we demonstrate the superiority of the kernel-matrix-based SPD representation using deep local descriptors. Secondly, we demonstrate the performance of the proposed DeepKSPD network in jointly learning local descriptors and the kernel-matrix-based SPD representation, with both normalisation methods of matrix logarithm and matrix α-rooting. Thirdly, ablation study is conducted to manifest the functions of the key layers in DeepKSPD, the improvement due to the kernel-matrix-based SPD representation, and the better numerical stability by using the gradients derived through the Daleckiĭ-Kreĭn formula. As will be seen, the proposed DeepKSPD network achieves the overall highest classification accuracy on the tested benchmark datasets, when compared with the related deep learning based methods.

2 Related Work

In the past decade, much work on covariance-matrix-based representation has been done in computer vision and machine learning, from a variety of perspective. Also, the recent integration of this representation with deep learning keeps producing new research results. In the following, we focus on the important existing works that are closely related to the DeepKSPD proposed in this paper.

Let $\boldsymbol{X}_{d\times n} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n]$ denote a data matrix, in which each column contains a local descriptor $\boldsymbol{x}_i$ ($\boldsymbol{x}_i \in \mathcal{R}^d$), extracted from an image. The SPD representation traditionally computes a $d \times d$ covariance matrix over $\boldsymbol{X}$ as $\boldsymbol{\Sigma} = \bar{\boldsymbol{X}}\bar{\boldsymbol{X}}^T$ (or simply $\boldsymbol{X}\boldsymbol{X}^T$), where $\bar{\boldsymbol{X}}$ denotes the centered $\boldsymbol{X}$. Originally, this covariance matrix is proposed as a region descriptor, for example, characterizing the covariance of the color intensities of pixels in an image patch. In the past several years, it has been employed as a promising second-order pooled image representation in visual recognition. One line of research on SPD representation models the nonlinear information in a set of descriptors. The work in [15] implicitly maps each descriptor $\boldsymbol{x}_i$ ($i = 1, 2, \cdots, n$) onto a kernel-induced feature space and computes a covariance matrix therein. Nevertheless, this results in a high (or even infinite) dimensional covariance matrix which is difficult to manipulate explicitly or computationally. Another work in [16] directly computes a kernel matrix $\boldsymbol{K}$ over $\boldsymbol{X}$ as follows. Let $\boldsymbol{f}_j$ denote the jth row of $\boldsymbol{X}$, consisting of the n realizations of the jth component of $\boldsymbol{x}$. The (i, j)th entry of $\boldsymbol{K}$ is calculated as $k(\boldsymbol{f}_i, \boldsymbol{f}_j)$, with a predefined kernel function k such as a Gaussian kernel. In this way, the nonlinear relationship among the d components can be extracted. The resulting kernel matrix $\boldsymbol{K}$ maintains the size of $d \times d$ and is more robust against the singularity issue caused by small sample. Covariance matrix is a special case in which k reduces to a linear kernel. As reported in [16], this kernel-matrix-based SPD representation considerably outperforms its covariance counterpart and the method in [15] on multiple visual recognition tasks.

Research on integrating the SPD representation *with* deep local descriptors or even *into* deep networks is still in its very early stage but has demonstrated both theoretical and practical values. In the recent work of Bilinear CNN [21, 13], an outer product layer is applied to combine the activation features maps from two CNNs, and this produces clear improvement in fine-grained visual recognition. This outer product essentially leads to a covariance matrix (in the form of $\boldsymbol{X}\boldsymbol{X}^T$) when the two CNNs are set as the same. The work in [18] trains a deep network for image semantic segmentation by using the covariance-matrix-based SPD representation. It carefully derives the gradients of covariance matrix functions from the scratch to carry out backpropagation. Considering that SPD matrix induces a Riemannian geometry, various normalisation operations have been used in the literature to make the matrix work with the classifiers that usually assume a Euclidean geometry. Matrix logarithm normalisation, $\log(\cdot)$, has been commonly used [22], and it is also taken in in [18]. Recently, the work in [12] shows that matrix square-rooting normalisation can even do better than the matrix logarithm counterpart, when applied to covariance-matrix-based SPD representation for fine-grained image classification. The work in [10] further shows and

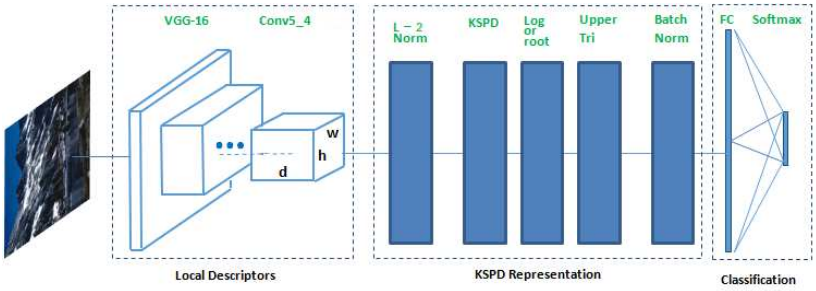

Fig. 1. The structure of the proposed DeepKSPD network

analyses the effectiveness of matrix square-rooting normalisation on large-scale image classification. Due to the verified efficacy of SPD representation on visual recognition, more works are being developed in recent literature from a variety of perspectives. For example, instead of directly computing a kernel as usual, the authors of [17] utilises Taylor series to approximate a kernel function via explicit feature maps, which allows them to generalise the Bilinear CNN framework to consider higher-order feature interaction.

3 The proposed network DeepKSPD

DeepKSPD consists of three blocks, as shown in Fig. 1. The leftmost block maps an input image into a set of deep local descriptors via a convolutional neural network. The rightmost block includes the commonly used fully connected and softmax layers for classification. In between is our design of KSPD block, which contains the layers related to the kernel-matrix-based representation and the matrix normalisation operation. For example, the input of the KSPD block is the output of the last convolutional layer (conv5_3) of the VGG-16 network (other CNN networks can certainly be used). It consists of d activation feature maps of the size of $w \times h$. They will go through the L_2 normalization layer and the KSPD layer which computes the kernel values among the d maps. Following that is the matrix normalisation layer (say, matrix logarithm or square-rooting based) to handle the Riemannian geometry of SPD matrix. Finally, since the KSPD representation is a symmetric matrix, a layer extracting its upper triangular and diagonal entries is deployed next to avoid redundancy. Particularly, an L_2 normalization and a batch normalization layer (not confuse with the above matrix normalisation layer) are added at the two ends of the KSPD block, as further explained below. We find that they help the kernel-matrix-based SPD representation to produce better classification.

***L_2* normalisation layer.** As aforementioned, its input is the output of the last convolutional layer with the dimensions of $w \times h \times d$. L_2 normalisation is done within each *feature channel*. That is, each channel with the dimensions of $w \times h$ are normalised to have a unit norm. This makes feature vectors and image

representation across a whole dataset comparable in terms of magnitude. Also, it helps to render the to-be-computed kernel values into their working range. This is essentially true when using the Gaussian RBF function which is exponential and bounded to $(0, 1]$. In this case, a poor initialisation of the Gaussian width θ (see Eq.(1)) could cause the kernel values in $\boldsymbol{K}$ too close to the boundary, making the backpropagation process inefficient. Also, it could decrease the discriminative capability of the learned SPD representation. With this L_2 normalisation layer, the proposed network becomes less sensitive to initialisation by restricting the feature vectors to a proper range and decreasing their variances.

Kernel SPD layer. The local descriptors calculated from the L_2 normalisation layer are pooled with a kernel function to obtain a global image representation. The input consists of the d normalised activation feature maps of the size of $w \times h$. These feature maps are reshaped along the depth dimension d, and this gives the data matrix $\boldsymbol{X}_{d\times n}$ with $n = w \times h$. Afterwards, the kernel matrix $\boldsymbol{K}_{d\times d}$ is computed over $\boldsymbol{X}$ to pool the n deep local descriptors, capturing the pairwise nonlinear relationship among the d feature maps. Note that in this layer the Gaussian width θ in Eq.(1) will be jointly learned via backpropagation.

Matrix function Layer. Following the KSPD layer, this framework performs matrix normalisation to handle the Riemannian geometry of SPD matrix, and this produces the matrix $\boldsymbol{H} = f(\boldsymbol{K})$. Traditionally, the normalisation function f is chosen as matrix logarithm. Recent studies [12, 10] report that matrix square rooting normalisation performs even better in majority of the cases. In our work, all the theoretical analysis assumes no specific normalisation operation and can handle any (continuously differentiable real) function f in backpropagation. In addition, Using the theoretical result provided in our work, we further generalize the existing matrix square-rooting normalisation to a matrix α-rooting normalisation, in which the power α is automatically learned via backpropagation instead of being fixed as 0.5. We also found that L_2 normalising the resulting matrix to have a unit norm allows a smoother convergence.

Batch normalisation layer. Batch normalisation layer is used as a post-processing step in our framework. During the forward propagation, each batch is normalised to have zero-mean and unit standard deviation. During the test, overall population statistics are used. In the literature, a layer alike has been used after convolutional layers to speed up convergence and reduce the sensitivity to initialisation. This batch normalisation layer in our framework functions in a similar way: it speeds up the convergence and allows a wider selection of an initial Gaussian width θ, in conjunction with the above L_2 normalisation layer, and helps to increase the overall classification accuracy subsequently. In the literature, the Bilinear CNN models use the setting of "element-wise signed square-rooting plus L_2 normalisation" after the image representation stage as the post-processing stage. Our investigation shows that the above batch normalisation setting works better with the proposed DeepKSPD framework and it is therefore taken in this paper.

4 End-to-end training of DeepKSPD

4.1 Derivatives between $\boldsymbol{X}$ and the kernel matrix $\boldsymbol{K}$

Recall that $\boldsymbol{X}_{d\times n}$ denotes a set of local descriptors. Considering that Gaussian RBF kernel is commonly used in the literature and that it is used in [16] to show the advantage of the kernel-matrix-based representation, we exemplify the proposed DeepKSPD with this kernel and derive the related gradients. Other kernels such as polynomial kernel can be dealt with in a similar way.

Let $\boldsymbol{I}_{d\times d}$ and $\mathbf{1}_{d\times d}$ denote an identity matrix and a matrix of 1s. Let $\circ$ denote the entry-wise product (Hadamard product) of two matrices, and $\exp[\cdot]$ denote an exponential function applied to a matrix in an entry-wise manner. In this way, the RBF kernel matrix $\boldsymbol{K}$ computed on $\boldsymbol{X}$ can be compactly expressed as

$$\boldsymbol{K} = \exp\left[-\theta \cdot \left((\boldsymbol{I}\circ\boldsymbol{X}\boldsymbol{X}^T)\mathbf{1} + \mathbf{1}^T(\boldsymbol{I}\circ\boldsymbol{X}\boldsymbol{X}^T)^T - 2\boldsymbol{X}\boldsymbol{X}^T\right)\right], \tag{1}$$

where θ is the Gaussian width. Let J be the objective function to be optimized by the DeepKSPD network. By temporarily assuming that the derivative $\frac{\partial J}{\partial \boldsymbol{K}}$ has been known (will be resolved in the next section), we now work out the derivatives $\frac{\partial J}{\partial \boldsymbol{X}}$ and $\frac{\partial J}{\partial \theta}$. J is a composition of functions applied to $\boldsymbol{X}$ and it can be rewritten as a function of each of the intermediate variables as follows.

$$J(\boldsymbol{X}) = J_1(\boldsymbol{A}) = J_2(\boldsymbol{E}) = J_3(\boldsymbol{K}), \tag{2}$$

where $\boldsymbol{A}$, $\boldsymbol{E}$, and $\boldsymbol{K}$ are defined, respectively, as

$$\boldsymbol{A} = \boldsymbol{X}\boldsymbol{X}^T, \qquad \boldsymbol{E} = \left((\boldsymbol{I}\circ\boldsymbol{A})\mathbf{1} + \mathbf{1}^T(\boldsymbol{I}\circ\boldsymbol{A})^T - 2\boldsymbol{A}\right), \quad \boldsymbol{K} = \exp[-\theta\cdot\boldsymbol{E}]. \tag{3}$$

Following the rules for differentiation, the following relationship can be obtained

$$\begin{aligned} &\delta\boldsymbol{A} = (\delta\boldsymbol{X})\,\boldsymbol{X}^T + \boldsymbol{X}\,(\delta\boldsymbol{X})^T, \\ &\delta\boldsymbol{E} = (\boldsymbol{I}\circ\delta\boldsymbol{A})\mathbf{1} + \mathbf{1}^T(\boldsymbol{I}\circ\delta\boldsymbol{A})^T - 2\delta\boldsymbol{A}, \quad \delta\boldsymbol{K} = (-\theta\boldsymbol{K})\circ\delta\boldsymbol{E}. \end{aligned} \tag{4}$$

It is known from the differentiation of a scalar-valued matrix function that

$$\delta J = \left\langle \operatorname{vec}\left(\frac{\partial J_3}{\partial \boldsymbol{K}}\right), \operatorname{vec}(\delta\boldsymbol{K})\right\rangle = \operatorname{trace}\left(\left(\frac{\partial J_3}{\partial \boldsymbol{K}}\right)^T \delta\boldsymbol{K}\right), \tag{5}$$

where $\operatorname{vec}(\cdot)$ denotes the vectorization of a matrix and $\langle\cdot,\cdot\rangle$ denotes the inner product. Combining this result with $\delta\boldsymbol{K} = (-\theta\boldsymbol{K})\circ\delta\boldsymbol{E}$ in Eq.(4) and using the identity that $\operatorname{trace}(\boldsymbol{A}^T(\boldsymbol{B}\circ\boldsymbol{C})) = \operatorname{trace}((\boldsymbol{B}\circ\boldsymbol{A})^T\boldsymbol{C})$, we can obtain

$$\delta J = \operatorname{trace}\left(\left(\frac{\partial J_3}{\partial \boldsymbol{K}}\right)^T \delta\boldsymbol{K}\right) = \operatorname{trace}\left(\left(-\theta\boldsymbol{K}\circ\frac{\partial J_3}{\partial \boldsymbol{K}}\right)^T \delta\boldsymbol{E}\right) = \operatorname{trace}\left(\left(\frac{\partial J_2}{\partial \boldsymbol{E}}\right)^T \delta\boldsymbol{E}\right). \tag{6}$$

The last equality holds because from Eq.(2) we know that δJ can also be written as $\operatorname{trace}\left(\left(\frac{\partial J_2}{\partial \boldsymbol{E}}\right)^T \delta\boldsymbol{E}\right)$. Noting that Eq.(6) is true for any $\delta\boldsymbol{E}$, we obtain

$$\frac{\partial J_2}{\partial \boldsymbol{E}} = (-\theta\boldsymbol{K})\circ\frac{\partial J_3}{\partial \boldsymbol{K}}. \tag{7}$$

Repeating the above process by using the relationship of $\delta \boldsymbol{E}$ and $\delta \boldsymbol{A}$ and that of $\delta \boldsymbol{A}$ and $\delta \boldsymbol{X}$ in Eq.(4), we can further have (proof is provided in the supplementary file

$$\frac{\partial J_1}{\partial \boldsymbol{A}} = \boldsymbol{I} \circ \left(\left(\frac{\partial J_2}{\partial \boldsymbol{E}} + \left(\frac{\partial J_2}{\partial \boldsymbol{E}} \right)^T \right) \mathbf{1}^T \right) - 2 \frac{\partial J_2}{\partial \boldsymbol{E}}; \quad \frac{\partial J}{\partial \boldsymbol{X}} = \left(\frac{\partial J_1}{\partial \boldsymbol{A}} + \left(\frac{\partial J_1}{\partial \boldsymbol{A}} \right)^T \right) \boldsymbol{X}. \quad (8)$$

In addition, the derivative $\frac{\partial J}{\partial \theta}$ can be obtained as

$$\frac{\partial J}{\partial \theta} = \text{trace} \left(\left(\frac{\partial J_3}{\partial \boldsymbol{K}} \right)^T (-\boldsymbol{K} \circ \boldsymbol{E}) \right). \quad (9)$$

Therefore, when $\frac{\partial J_3}{\partial \boldsymbol{K}}$ is available, we can work out $\frac{\partial J}{\partial \boldsymbol{X}}$ and $\frac{\partial J}{\partial \theta}$ accordingly.

4.2 Derivatives of the matrix function on the kernel matrix $\boldsymbol{K}$

Now, to obtain $\frac{\partial J_3}{\partial \boldsymbol{K}}$ we deal with the matrix normalisation operation between $\boldsymbol{K}$ and J, which can be written as

$$J(\boldsymbol{X}) = J_4(\boldsymbol{H}) = J_4(f(\boldsymbol{K})). \quad (10)$$

Note that $\frac{\partial J_4}{\partial \boldsymbol{H}}$ is ready to obtain because it only involves the classification layers like fully-connected layer, softmax regression and cross-entropy computation. The key issue is to obtain $\frac{\partial \boldsymbol{H}}{\partial \boldsymbol{K}}$. Now, we introduce the *Daleckiĭ-Kreĭn formula* [19] to give a concise and unified result on differentiating SPD matrix functions, of which both matrix logarithm and square-rooting normalisations are special cases.

Theorem 1 (pp.60, [20]) *Let $\mathbb{M}_d$ be the set of $d \times d$ real symmetric matrices. Let I be an open interval and $\mathbb{M}_d(I)$ is the set of all real symmetric matrices whose eigenvalues belong to I. Let $C^1(I)$ be the space of continuously differentiable real functions on I. Every function f in $C^1(I)$ induces a differentiable map from $\boldsymbol{A}$ in $\mathbb{M}_d(I)$ to $f(\boldsymbol{A})$ in $\mathbb{M}_d$. Let $Df_{\boldsymbol{A}}(\cdot)$ denote the derivative of $f(\boldsymbol{A})$ at $\boldsymbol{A}$. It is a linear map from $\mathbb{M}_d$ to itself. When applied to $\boldsymbol{B} \in \mathbb{M}_d$, $Df_{\boldsymbol{A}}(\cdot)$ is given by the Daleckiĭ-Kreĭn formula as*

$$Df_{\boldsymbol{A}}(\boldsymbol{B}) = \boldsymbol{U} \left(\boldsymbol{G} \circ \left(\boldsymbol{U}^T \boldsymbol{B} \boldsymbol{U} \right) \right) \boldsymbol{U}^T, \quad (11)$$

where $\boldsymbol{A} = \boldsymbol{U} \boldsymbol{D} \boldsymbol{U}^T$ is the eigen-decomposition of $\boldsymbol{A}$ with $\boldsymbol{D} = \text{diag}(\lambda_1, \cdots, \lambda_d)$, and $\circ$ is the entry-wise product. The entry of the matrix $\boldsymbol{G}$ is defined as

$$g_{ij} = \begin{cases} \frac{f(\lambda_i) - f(\lambda_j)}{\lambda_i - \lambda_j} & \text{if } \lambda_i \neq \lambda_j \\ f'(\lambda_i), & \text{otherwise.} \end{cases} \quad (12)$$

This theorem indicates that for a matrix function $f(\cdot)$ applied to $\boldsymbol{A}$, perturbing $\boldsymbol{A}$ by a small amount $\boldsymbol{B}$ will vary $f(\boldsymbol{A})$ by the quantity $Df_{\boldsymbol{A}}(\boldsymbol{B})$ in Eq.(11), where the variation is in the sense of the first-order approximation. Now we show how to derive the functional relationship between $\frac{\partial J_4}{\partial \boldsymbol{H}}$ and $\frac{\partial J_3}{\partial \boldsymbol{K}}$ based on Theorem 1. According to Eq.(2) and following the argument in Eq.(5),

$$\delta J = \text{trace} \left(\left(\frac{\partial J_4}{\partial \boldsymbol{H}} \right)^T \delta \boldsymbol{H} \right) = \text{trace} \left(\left(\frac{\partial J_3}{\partial \boldsymbol{K}} \right)^T \delta \boldsymbol{K} \right). \quad (13)$$

Applying the Daleckiĭ-Kreĭn formula, we can explicitly represent $\delta \boldsymbol{H}$ as

$$\delta \boldsymbol{H} = Df_{\boldsymbol{K}}(\delta \boldsymbol{K}) = \boldsymbol{U}\left(\boldsymbol{G} \circ \left(\boldsymbol{U}^T \delta \boldsymbol{K} \boldsymbol{U}\right)\right)\boldsymbol{U}^T. \tag{14}$$

Replacing $\delta \boldsymbol{H}$ in Eq.(13) with this result and applying the properties of trace($\boldsymbol{A}^T\boldsymbol{B}$), the relationship of $\frac{\partial J_4}{\partial \boldsymbol{H}}$ and $\frac{\partial J_3}{\partial \boldsymbol{K}}$ can be derived similarly as in Eqs.(6) and (7):

$$\frac{\partial J_3}{\partial \boldsymbol{K}} = \boldsymbol{U}\left(\boldsymbol{G} \circ \left(\boldsymbol{U}^T \frac{\partial J_4}{\partial \boldsymbol{H}} \boldsymbol{U}\right)\right)\boldsymbol{U}^T. \tag{15}$$

where $\boldsymbol{U}$ and $\boldsymbol{G}$ are obtained from the eigen-decomposition of $\boldsymbol{K} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^T$. For matrix logarithm (or square-rooting) normalisation, g_{ij} in Eq.(12) is computed as $\frac{\log \lambda_i - \log \lambda_j}{\lambda_i - \lambda_j}$ (or $\frac{\sqrt{\lambda_i} - \sqrt{\lambda_j}}{\lambda_i - \lambda_j}$) when $i \neq j$, and λ_i^{-1} (or $\frac{1}{2\sqrt{\lambda_i}}$) otherwise.

The work in [18] derives the derivative of the matrix logarithm from the scratch with the basic facts of matrix differentiation, which is detailed and instructive. As previously mentioned, that work does not connect with the well-established Daleckiĭ-Kreĭn formula. Moreover, it is reported that the derivation in [18] could lead to numerical instability during training a deep model [12]. To solve this problem and clarify the link with the existing result in [18], we prove the following proposition (proof is provided in the supplementary file).

Proposition 1 *The functional relationship obtained in [18] shown in Eq.(16) (with the notation in this work for consistency) is equivalent to that in Eq.(15) obtained by this work.*

$$\frac{\partial J_3}{\partial \boldsymbol{K}} = \boldsymbol{U}\left\{\left(\tilde{\boldsymbol{G}} \circ \left(2\boldsymbol{U}^T\left(\frac{\partial J_4}{\partial \boldsymbol{H}}\right)_{sym}\boldsymbol{U}\log(\boldsymbol{D})\right)\right) + \left(\boldsymbol{D}^{-1}\left(\boldsymbol{U}^T\frac{\partial J_4}{\partial \boldsymbol{H}}\boldsymbol{U}\right)\right)_{diag}\right\}\boldsymbol{U}^T, \tag{16}$$

where $\boldsymbol{K} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^T$; $\tilde{g}_{ij} = (\lambda_i - \lambda_j)^{-1}$ *when* $i \neq j$ *and zero otherwise;* $\boldsymbol{A}_{diag}$ *means the off-diagonal entries of* $\boldsymbol{A}$ *are all set to zeros; and* $\boldsymbol{A}_{sym}$ *is defined to represent* $(\boldsymbol{A} + \boldsymbol{A}^T)/2$.

Connecting with the results in operator theory not only facilitates the access to the derivatives of general SPD matrix functions, but also provides us more insight on these functions that could be useful for future research.

4.3 Numerical Stability Issue of the Matrix Gradients

The numerical stability issue associated with the derivation in [18] is explained as follows. Recall that Eq.(16) is used in [18] to calculate the gradients of the matrix function $\partial J_3/\partial \boldsymbol{K}$. In the matrix $\tilde{\boldsymbol{G}}$, the elements are $\tilde{g}_{ij} = (\lambda_i - \lambda_j)^{-1}$ when $i \neq j$ and 0 when $i = j$ in Eq.(16). When two of the eigenvalues are too close to each other, due to the finite precision arithmetic, the λ_i will cancel λ_j and the $\tilde{g}_{ij}$ will become *infinity*. This causes problems in the backpropagation process, as reported by [12]. Using double precision is not enough to alleviate the problem. A few possible workarounds are: excluding the batches causing this problem or appending a small number ϵ to the term $\tilde{g}_{ij}$. However, both of these approaches cause considerable performance drop.

In our derivation with the Daleckiĭ-Kreĭn formula, this issue is resolved in Eq.(12), where g_{ij} is defined as $\frac{f(\lambda_i)-f(\lambda_j)}{\lambda_i-\lambda_j}$ if $\lambda_i \neq \lambda_j$ when $\lambda_i \neq \lambda_j$ and $f'(\lambda_i)$ otherwise. In this case, when λ_i is too close to λ_j, we can formulate the problem as $g_{ij} = \lim_{\lambda_i \to \lambda_j} \frac{f(\lambda_i)-f(\lambda_j)}{\lambda_i-\lambda_j}$, where λ_j is viewed as constant. Since this is a $\frac{0}{0}$ uncertainty, by applying L'Hôpitals rule we obtain that $g_{ij} = \lim_{\lambda_i \to \lambda_j} \frac{f(\lambda_i)-f(\lambda_j)}{\lambda_i-\lambda_j} = \lim_{\lambda_i \to \lambda_j} \frac{f'(\lambda_i)}{1} = f'(\lambda_j)$. In this way, the numerical stability in [18] is avoided. This theoretical analysis will be further supported by an experiment conducted later in this paper.

4.4 Generalise to Matrix α-rooting normalisation

We are aware of the recent success of matrix square-rooting [12, 10] used in deep learning structures to handle the Riemannian geometry of SPD matrices. In addition to highlighting that matrix square-rooting and matrix logarithm are two special cases of our derivation in Eq.(12), we further generalise the existing matrix square-rooting normalisation to a case that we call "matrix α-rooting" normalisation. It is defined as $f(\lambda) = \lambda^\alpha$ where α is a parameter to be jointly learned by our DeepKSPD framework, instead of being fixed as 0.5 in the matrix square-rooting normalisation. $\frac{\partial J_3}{\partial \boldsymbol{K}}$ will still maintain as in Eq.(15) and the derivative with respect to the parameter α, $\frac{\partial J}{\partial \alpha}$, can be derived as:

$$\frac{\partial J}{\partial \alpha} = \operatorname{trace}\left(\left(\frac{\partial \boldsymbol{J}_4}{\partial \boldsymbol{H}}\right)^T \left[\boldsymbol{U}(\log(\boldsymbol{D}) \circ \boldsymbol{D}^\alpha)\boldsymbol{U}^T\right]\right), \tag{17}$$

where $\boldsymbol{U}$ and $\boldsymbol{D}$ are the eigen-decomposition of $\boldsymbol{K}$ as previous. Note that this matrix α-rooting is still guaranteed to be numerically stable in backpropagation, as shown in Section 4.3. Its performance will also be experimentally verified later.

5 Experimental Result

We have two tasks: i) test the performance of KSPD built upon deep local descriptors and ii) test the performance of the proposed DeepKSPD network, on fine-grained image and scene recognition. Bounding boxes are not used in all datasets. Example images are in the supplementary file.

Datasets. Four benchmark data sets are tested. For scene recognition, the MIT Indoor data set has 67 classes with predefined 5600 training and 1340 test images. For fine-grained image recognition, three data sets of Cars [23], Birds [24], and Aircrafts [25] are tested. The Cars dataset has 16185 images from 196 classes; the Aircrafts contains 10200 images of 100 classes (variants). The Birds has 11788 samples of 200 bird species. *Note that in order to have a fair comparison with [13] and [12] on the Aircraft dataset, images are first resized* 512×512 *then a central* 448×448 *patch is cropped. This increases the classification accuracy by* $2\% \sim 3\%$.

Setting of Proposed Methods. For the first task, we propose a method called KSPD-VGG, which builds kernel-matrix-based SPD representation upon the

deep local descriptors from VGG-19 pretrained on ImageNet. Specifically, the 512 feature maps (of size 27×27) of the last convolutional layer are reshaped to form 512 vectors with the dimensions of 729. These vectors are further used to compute the 512×512 kernel matrix $\boldsymbol{K}$. Then, the matrix logarithm is applied and the resulting KSPD representations of all images are further processed by PCA dimension reduction (to 4096 dimensions), standardization (to zero mean and unit standard deviation), and ℓ_2 normalization. A linear SVM classifier is employed to perform the classification.

For the second task, the proposed DeepKSPD network is trained and tested. DeepKSPD has three blocks (Fig. 1). In the local descriptor block, the network hyperparameters are set by following VGG-16. In the proposed KSPD block, no hyperparameter needs to be tuned, and θ and α will be automatically learned with their initial values set as 0.1 and 0.5, respectively, for all the experiments. We test both matrix logarithm (denoted as DeepKSPD-logm) and matrix α-rooting (denoted as DeepKSPD-rootm) normalisations. In the classification block, the size of FC layer is set as the number of classes for each data set. DeepKSPD is trained by Adaptive Moment Estimation (Adam) in mini-batch mode (with the batch-size of 20). A two-step training procedure [26] is applied as good performance is observed [26, 21]. Specifically, we first train the last layer using softmax regression for 100 epochs, and fine-tune the whole system. The total training epochs are $70 \sim 100$. We only use flipping in training time as data augmentation.

Methods in Comparison. We compare our KSPD-VGG and DeepKSPD with methods that are either comparable or competitive in the literature, as listed in the first column in Table 1, and are roughly grouped into three categories.

The first category can be deemed as feature extraction methods, to which KSPD-VGG belongs. This category also includes FV-SIFT [27], FC-VGG [18], FV-VGG [28], and COV-VGG (standing for covariance-matrix-based SPD representation). COV-VGG's setting is same as that of KSPD-VGG, except that a covariance matrix is constructed instead of a kernel matrix. Note that, we directly quote the results of FV-SIFT and FC-VGG from the literature, and provide our own implementation of FV-VGG, COV-VGG, and KSPD-VGG to ensure the same setting for fair comparison.

The second category includes six end-to-end learning methods. DeepKSPD-logm and DeepKSPD-rootm are the proposed methods. B-CNN denotes the bilinear CNN method [21] and Improved BCNN [12] is an extension of B-CNN where matrix square-rooting is applied. CBP [14] and LRBP [11] are both COV-based methods and KP [17] estimates Gaussian RBF features using Taylor series.

In the third category, additional methods previously reported on these data sets are quoted to extend the comparison and provide a whole picture.

Results and Discussion. From Table 1 we have the following observations. First, the proposed KSPD-VGG, DeepKSPD-logm, DeepKSPD-rootm demonstrate their effectiveness for visual recognition. On every dataset, the best performance is achieved by the proposed DeepKSPD. Moreover, DeepKSPD is superior to KSPD-VGG (up to 9.7 percentage points on Cars) and other competitive meth-

Table 1. Comparison of Methods

ACC (%)	MIT indoor	Cars	Aircraft	Birds	Average
Symbiotic Model [29]	–	78.0	72.5	–	–
FV-revisit [30]	–	82.7	80.7	–	–
FV-SIFT [27]	–	59.2	61.0	18.8	–
FC-VGG [21]	67.6	36.5	45.0	61.0	52.5
FV-VGG [28]	73.7	75.2	72.7	71.3	73.1
FV-VGG-ft [21]	–	85.7	78.7	74.7	73.1
COV-VGG	74.2	80.3	81.4	76	78.0
KSPD-VGG (**proposed**)	77.2	83.5	83.8	78.5	80.1
BCNN [13]	77.6	91.3	86.6	84.1	84.5
Improved BCNN [12]	–	92.0	88.5	85.8	–
CBP [14]	76.17	–	–	84.0	–
LRBP [11]	–	90.9	87.3	84.2	–
KP [17]	–	92.4	86.9	86.2	–
DeepKSPD-logm (**proposed**)	79.6	90.5	**91.5**	84.8	86.6
DeepKSPD-rootm (**proposed**)	**81.0**	**93.2**	91.0	**86.5**	**87.9**

ods, demonstrating the essentials of the end-to-end learning of kernel-matrix-based representation. Among the two DeepKSPD methods, DeepKSPD-rootm performs better on MIT indoor, Cars, and Birds, while DeepKSPD-logm performs better on Aircraft. Overall, DeepKSPD-rootm wins over DeepKSPD-logm, which is consistent with the observation in [13] that matrix α-rooting seems to have some advantages in scaling the eigenvalues over matrix logarithm.

Second, KSPD-based methods consistently win COV-based ones **(or bilinear)** on all data sets, either based on feature extraction (KSPD-VGG vs COV-VGG) or using end-to-end training (Deep KSPD vs other COV-based methods including B-CNN, improved B-CNN, CBP and LRBP). It is interesting to see that KP also shows promising performance by approximating kernel representation, which supports our arguments of employing kernel representation for visual recognition. However, this method neither directly learns kernel representation nor explicitly handles the Riemannian geometry of SPD matrix as our method. Instead, it approximates the kernel representation by Taylor expansion.

Third, the SPD representation (being it based on an outer product, covariance, or kernel matrix) outperforms Fisher vector representation in the given tasks. DeepKSPD also outperforms FV-VGG-ft obtained from fine-tuned VGG-16. The latter attained 78.7% on Aircraft, 74.7% on Birds, and 85.7% on Cars [21], which is worse than 81.0%, 86.5% and 93.2% achieved by DeepKSPD.

Fourth, in the literature, matrix logarithm normalisation has not been very successfully incorporated into deep CNNs up to our work due to numerical instability issue. Furthermore, it was dismissed due to poor results compared to matrix square-rooting. Our numerically stable gradients render the embedding of matrix logarithm into deep architecture possible. More importantly, we show that matrix logarithm is still relevant as it yields the best results on Aircraft dataset. Thus, matrix functions to handle the Riemannian geometry could be regarded as hyper-parameters and properly chosen via validation mechanism.

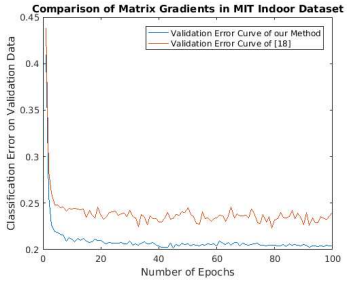

Fig. 2. On numerical stability.

ACC (%)	MIT indoor	Cars	Aircraft	Birds
Improved BCNN [12]	–	92.0	88.5	85.8
DeepCOV-rootm	79.2	91.7	88.7	85.4
DeepKSPD-rootm	**81.0**	**93.2**	**91.0**	**86.5**

Table 2. DeepKSPD vs DeepCOV.

Numerical stability. Before ending the experiment part, we also conduct a test on the numerical stability of our formulation of matrix derivative. We investigate the performance of our DeepKSPD-logm on MIT indoor dataset with the derivative of matrix logarithm computed by [18] and our formulation, respectively. The result is shown in Fig. (2). As can be seen, our method achieves lower classification error consistently in all epochs. After 100 epochs of training, the classification error is 22.4% using our method, and 24% using the formulation in [18], well demonstrating the advantage of our derived unified solution.

6 Ablation Study

Different from the literature, our framework utilizes an L_2 normalisation layer before the pooling layer and a batch normalization layer after. Traditionally, these layers are not used in bilinear models. However, RBF kernel matrix has a very different nature than covariance matrix. As previously explained, its kernel values are bounded between 0 and 1. If one feature channel dominates the others in terms of the magnitude, it will cause numerical problems. A common way in machine learning to tackle this problem is to normalise feature channels o make them comparable in magnitude. We also adopt this approach and integrate L_2 normalisation as a layer into our framework before kernel pooling stage.

Post-processing is very important in SPD representations. In the literature, bilinear models [14, 12, 21, 17, 11] use element-wise signed rooting followed by L_2 normalisation layers, which contributes around **5.7%** [21] of the classification accuracy. In our framework, KSPD layer has a parameter θ which must be initialized properly. We found that batch normalisation layer, that is used to cope with poor initialization of convolutional layers, can be used for this task. Therefore, we replace element-wise signed square root with batch normalisation layer. In the table below, the experiments are conducted with DeepKSPD-rootm structure built on VGG-16 network.

According to Table 3 on average, batch normalization layer contributes to 2.56% of the performance; whereas, element-wise signed square root + L_2 normalisation processing increases performance about 1.45%. Furthermore, the convergence is around 3 times faster. Most importantly, our design choice allows a

Table 3. Comparison of Post Processing

ACC (%)	MIT indoor	Cars	Aircraft	Birds
DeepKSPD-sqrt-L_2	80.6	90.1	86.1	84.7
DeepKSPD-w/o BN	77.6	89.6	84.3	81.0
DeepKSPD-w BN	**81.0**	**93.2**	**91.0**	**86.5**

Table 4. Final Parameter Values

ACC (%)	MIT indoor	Cars	Aircraft	Birds
Initial θ	0.1	0.1	0.1	0.1
Initial α	0.5	0.5	0.5	0.5
Final θ	0.63	1.4	0.67	0.93
Final α	0.49	0.52	0.53	0.52

universal initial value (we choose 0.1) for the parameter θ for all the datasets. Note that we conduct a grid search for θ and report the best result in Table 3 for DeepKSPD-sqrt-L_2 and DeepKSPD-w/o BN.

In the Table 4 the initial and final values of α and θ are given. In the literature, [12, 10] do a similar experiment with matrix rooting; however, the authors only do a grid search to find the best root. We provide derivatives for matrix rooting and update rooting power α with each iteration in our work.

As shown, α values do not deviate much from their initial values. However θ values end up in much different values from their starting point. Even when the initial θ is much lower than its final values, DeepKSPD performs excellent in each case; supporting our design choice to tackle the initialization problem.

6.1 Kernel Representation versus Covariance Representation

Since DeepKSPD does not adopt the same network as bilinear methods, to show the benefits purely from kernelising, we test the covariance and the kernel representations when they share an identical network. For this purpose, we introduce another model called DeepCOV that is identical to DeepKSPD except that DeepCOV adopts covariance-based-matrix representation. We compare DeepCOV to DeepKSPD in Table 2. As shown, on all the datasets DeepKSPD outperforms DeepCOV. This is a clear demonstration of the superiority of kernelising local descriptors over the second-order pooling of them. Furthermore, DeepCOV performs almost identically to [12]. This indicates that the layers and strategies designed in our DeepKSPD cater well for the special characteristics of KSPD representation that are not necessarily presented in the bilinear models.

7 Conclusion

Motivated by the recent progress on SPD representation, we develop a deep neural network that jointly learns local descriptors and kernel-matrix-based SPD representation for fine-grained image recognition. The matrix derivatives required by the backpropagation process are derived and linked to the established literature on the theory of positive definite matrix. Experimental result on benchmark datasets demonstrates the improved performance of kernel-matrix-based SPD representation when built upon deep local descriptors and the superiority of the proposed DeepKSPD network.

References

1. Sivic, J., Zisserman, A.: Video google: A text retrieval approach to object matching in videos. In: 9th IEEE International Conference on Computer Vision (ICCV 2003). (2003) 1470–1477
2. Wang, J., Yang, J., Yu, K., Lv, F., Huang, T.S., Gong, Y.: Locality-constrained linear coding for image classification. In: The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010. (2010) 3360–3367
3. Jegou, H., Douze, M., Schmid, C., Pérez, P.: Aggregating local descriptors into a compact image representation. In: The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010. (2010) 3304–3311
4. Sánchez, J., Perronnin, F., Mensink, T., Verbeek, J.J.: Image classification with the fisher vector: Theory and practice. International Journal of Computer Vision **105**(3) (2013) 222–245
5. Jayasumana, S., Hartley, R.I., Salzmann, M., Li, H., Harandi, M.T.: Kernel methods on the riemannian manifold of symmetric positive definite matrices. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition. (2013) 73–80
6. Wang, R., Guo, H., Davis, L.S., Dai, Q.: Covariance discriminative learning: A natural and efficient approach to image set classification. [7] 2496–2503
7. 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012, IEEE Computer Society (2012)
8. Fleet, D.J., Pajdla, T., Schiele, B., Tuytelaars, T., eds.: Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part II. Volume 8690 of Lecture Notes in Computer Science., Springer (2014)
9. Bach, F.R., Blei, D.M., eds.: Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015. Volume 37 of JMLR Workshop and Conference Proceedings., JMLR.org (2015)
10. Li, P., Xie, J., Wang, Q., Zuo, W.: Is second-order information helpful for large-scale visual recognition? In: IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017. (2017) 2089–2097
11. Kong, S., Fowlkes, C.C.: Low-rank bilinear pooling for fine-grained classification. CoRR **abs/1611.05109** (2016)
12. Lin, T.Y., Maji, S.: Improved Bilinear Pooling with CNNs. In: British Machine Vision Conference (BMVC). (2017)
13. Lin, T.Y., RoyChowdhury, A., Maji, S.: Bilinear cnns for fine-grained visual recognition. In: Transactions of Pattern Analysis and Machine Intelligence (PAMI). (2017)
14. Gao, Y., Beijbom, O., Zhang, N., Darrell, T.: Compact bilinear pooling. CoRR **abs/1511.06062** (2015)
15. Harandi, M.T., Salzmann, M., Porikli, F.M.: Bregman divergences for infinite dimensional covariance matrices. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014. (2014) 1003–1010
16. Wang, L., Zhang, J., Zhou, L., Tang, C., Li, W.: Beyond covariance: Feature representation with nonlinear kernel matrices. In: 2015 IEEE International Conference on Computer Vision, ICCV 2015. (2015) 4570–4578
17. Cui, Y., Zhou, F., Wang, J., Liu, X., Lin, Y., Belongie, S.: Kernel pooling for convolutional neural networks. In: Computer Vision and Pattern Recognition (CVPR), Honolulu, HI (2017)

18. Ionescu, C., Vantzos, O., Sminchisescu, C.: Matrix backpropagation for deep networks with structured layers. In: 2015 IEEE International Conference on Computer Vision, ICCV 2015. (2015) 2965–2973
19. Daleckiĭ, Y.L., Kreĭn, S.G.: Integration and differentiation of functions of hermitian operators and applications to the theory of perturbations. (Russian) Vorone. Gos. Univ. Trudy Sem. Funkcional. Anal. 1 (1) (1956) 81–105 English translation is in book *Thirteen Papers on Functional Analysis and Partial Differential Equations*, American Mathematical Society Translations: Series 2, vol.47, 1965.
20. Bhatia, R.: Positive Definite Matrices. Princeton University Press (2015)
21. Lin, T., Roy Chowdhury, A., Maji, S.: Bilinear CNN models for fine-grained visual recognition. In: 2015 IEEE International Conference on Computer Vision, ICCV 2015. (2015) 1449–1457
22. Arsigny, V., Fillard, P., Pennec, X., Ayache, N.: Log-euclidean metrics for fast and simple calculus on diffusion tensors. Magnetic Resonance in Medicine **56**(2) (2006) 411–421
23. Krause, J., Stark, M., Deng, J., Fei-Fei, L.: 3d object representations for fine-grained categorization. In: 4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13), Sydney, Australia (2013)
24. Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S., Perona, P.: Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology (2010)
25. Maji, S., Rahtu, E., Kannala, J., Blaschko, M.B., Vedaldi, A.: Fine-grained visual classification of aircraft. CoRR **abs/1306.5151** (2013)
26. Branson, S., Horn, G.V., Belongie, S., Perona, P.: Bird species categorization using pose normalized deep convolutional nets. In: British Machine Vision Conference (BMVC), Nottingham (2014)
27. Perronnin, F., Sánchez, J., Mensink, T.: Improving the fisher kernel for large-scale image classification. In: Proceedings of the 11th European Conference on Computer Vision: Part IV. ECCV'10, Springer-Verlag (2010) 143–156
28. Cimpoi, M., Maji, S., Vedaldi, A.: Deep filter banks for texture recognition and segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015. (2015) 3828–3836
29. Chai, Y., Lempitsky, V., Zisserman, A.: Symbiotic segmentation and part localization for fine-grained categorization. In: IEEE International Conference on Computer Vision. (2013)
30. Gosselin, P.H., Murray, N., Jégou, H., Perronnin, F.: Revisiting the Fisher vector for fine-grained classification. Pattern Recognition Letters **49** (November 2014) 92–98