A Robust Distance Measure for Similarity-Based Classification on the SPD Manifold

Zhi Gao, Yuwei Wu, *Member, IEEE*, Mehrtash Harandi, *Member, IEEE*, and Yunde Jia, *Member, IEEE*

***Abstract*— The symmetric positive definite (SPD) matrices, forming a Riemannian manifold, are commonly used as visual representations. The non-Euclidean geometry of the manifold often makes developing learning algorithms (e.g., classifiers) difficult and complicated. The concept of similarity-based learning has been shown to be effective to address various problems on SPD manifolds. This is mainly because the similarity-based algorithms are agnostic to the geometry and purely work based on the notion of similarities/distances. However, existing similarity-based models on SPD manifolds opt for holistic representations, ignoring characteristics of information captured by SPD matrices. To circumvent this limitation, we propose a novel SPD distance measure for the similarity-based algorithm. Specifically, we introduce the concept of point-to-set transformation, which enables us to learn multiple lower dimensional and discriminative SPD manifolds from a higher dimensional one. For lower dimensional SPD manifolds obtained by the point-to-set transformation, we propose a tailored set-to-set distance measure by making use of the family of alpha–beta divergences. We further propose to learn the point-to-set transformation and the set-to-set distance measure jointly, yielding a powerful similarity-based algorithm on SPD manifolds. Our thorough evaluations on several visual recognition tasks (e.g., action classification and face recognition) suggest that our algorithm comfortably outperforms various state-of-the-art algorithms.**

***Index Terms*— Metric learning, similarity-based classification, symmetric positive definite (SPD) manifold, visual information.**

I. Introduction

MANY visual representations lie on Riemannian manifolds, which are non-Euclidean spaces. Developing a true Riemannian manifold geometry has received significant attention in the computer vision community. The sphere, symmetric positive definite (SPD), Stiefel, and Grassmannian manifolds are frequently encountered Riemannian manifolds [1]. In this article, we focus on the SPD manifold formed by SPD matrices, e.g., the covariance matrix, the kernel matrix, and the diffusion tensor image (DTI). The SPD matrix utilizes the second-order or high-order statistical information to capture the desirable feature distribution and has been successfully applied to many visual tasks, such as face recognition [2], action recognition [3], transfer learning [4], object detection [5], and medical image processing [6].

Classification is one of the basic tasks in computer vision. However, the non-Euclidean geometry often makes developing classifiers on SPD manifolds difficult and complicated. The input of a standard Euclidean-space-learning classifier (e.g., the support vector machine) is a feature vector that lies in the Euclidean space, while the SPD manifold is clearly not a Euclidean space. Most existing SPD manifold classification methods convert an SPD representation to a requisite vector by the tangent approximation [7], [8], the kernel method [9]–[12], or the coding technique [13]–[15]. These methods are not superior because the vector operation on an SPD representation inevitably distorts the matrix structure. To better build a Riemannian-space-learning classifier, the similarity-based classification scheme [16] is a good choice and has shown promising performance. The similarity-based classifier does not consider the input-lied space, and it needs only similarities of any pairs of inputs. In this case, a robust distance measure that computes the similarity between two SPD representations is critical for classifiers to develop SPD manifolds.

To measure a true SPD manifold geometry, many efforts have been devoted to the distance measure on SPD manifolds, such as the affine-invariant metric (AIM) [17], log-Euclidean metric (LEM) [18], Stein divergence [19], Burg matrix divergence [20], and alpha–beta divergence [21], [22]. Because underlying data distributions of concrete tasks are customarily different, metric learning is widely employed to provide a proper distance measure from the given data. Most existing methods learn the SPD distance measure in a more discriminative space, e.g., the Euclidean tangent space [23]–[25], the reproducing kernel Hilbert space (RKHS) [26], and other representative SPD manifolds [27], through which similar pairs are close and dissimilar pairs are far apart.

As the SPD representation is aggregated from local features that contain multiple types of visual information [25], the SPD representation can be seen as a combination of visual information statistics. For example, elements of a covariance matrix are relations of dimensions of local features. We argue that visual information may have different characteristics, i.e., different distributions and contributions for both constructing the SPD representation and measuring the distance between two SPD representations. It is necessary to separately consider different visual information to build a more robust distance measure. However, many existing metric learning

Manuscript received September 24, 2018; revised March 24, 2019 and August 3, 2019; accepted August 28, 2019. Date of publication September 27, 2019; date of current version September 1, 2020. This work was supported in part by the Natural Science Foundation of China (NSFC) under Grant 61702037 and Grant 61773062 and in part by Beijing Municipal Natural Science Foundation under Grant L172027. *(Zhi Gao and Yuwei Wu are co-first authors.) (Corresponding author: Yuwei Wu.)*

Z. Gao, Y. Wu, and Y. Jia are with the Beijing Laboratory of Intelligent Information Technology, School of Computer Science, Beijing Institute of Technology (BIT), Beijing 100081, China (e-mail: gaozhi_2017@bit.edu.cn; wuyuwei@bit.edu.cn; jiayunde@bit.edu.cn).

M. Harandi is with the Department of Electrical and Computer Systems Engineering, Monash University, Melbourne, VIC 3800, Australia (e-mail: mehrtash.harandi@monash.edu).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2019.2939177

methods [23]–[27] only treat an SPD representation as a global representation and ignore diversities of different visual information, resulting in an inferior method for exploring the SPD representation.

In this article, we propose a novel SPD distance measure for similarity-based classification on SPD manifolds. Unlike existing methods that apply a single SPD manifold distance measure, we aim to compute multiple subdistances from discovered discriminative visual information. Specifically, we apply a point-to-set transformation containing multiple low-dimensional manifold projections on each SPD representation to discover discriminative visual information, and the obtained low-dimensional SPD matrices constitute an SPD set. Then, the distance between two original SPD representations is converted to a set-to-set distance that integrates multiple individual subdistances of low-dimensional matrix pairs from each original SPD representation. In this work, the learnable alpha–beta divergence is utilized to measure the subdistances, which can preserve the desirable SPD manifold structure. In fact, learning the set-to-set distance measure can be seen as a multi-metric learning problem on the SPD set [28], [29]. Parameters of the proposed distance measure are modeled on Riemannian manifolds, and we develop a Riemannian optimization algorithm to learn the parameters. Evaluated by experiments, our distance measure is extremely helpful in capturing meaningful nearest neighbors for a similarity-based classifier.

In summary, our contributions are threefold as follows.

1) We present an effective SPD distance measure for the similarity-based classification scheme on SPD manifolds. Considering different characteristics of visual information in the SPD representation, the proposed distance measure can more effectively explore the discriminative visual information and enhance the robustness of the similarity-based classifier, even when we only use a simple nearest neighbor classifier.
2) We formulate the SPD distance as an SPD point-to-set transformation and a set-to-set distance measure problem. The point-to-set transformation contains multiple low-dimensional manifold projections on original SPD representations, and the set-to-set distance is designed as integrating individual subdistances on low-dimensional manifolds.
3) We learn multiple individual alpha–beta divergences with different weights on low-dimensional manifolds as subdistances to specifically consider different distributions and contributions of visual information. In addition, we introduce a Riemannian optimization algorithm to jointly learn the point-to-set transformation and the set-to-set distance to find a solution effectively and efficiently.

The rest of this article is organized as follows. We review the related SPD manifold classification and metric learning methods in Section II. The preliminaries of our method are stated in Section III. Section IV presents the details of our distance measure, including the point-to-set transformation and the set-to-set distance. The corresponding optimization is described in Section V. Section VI provides the computational complexities of our distance measure and optimization. We show the experimental results on six data sets to demonstrate the effectiveness of our method in Section VII, and conclude this article in Section VIII.

II. Related Work

In this section, we review recent SPD manifold classification methods and metric learning methods.

A. SPD Manifold Classification

In the SPD manifold classification task, most works convert the SPD representation to a Euclidean vector, among which the tangent approximation is a widely used approach. The tangent space of a manifold point is a Euclidean space, and thus Euclidean-space-learning classifiers can be applied. Dong *et al.* [7] and Huang and Van Gool [8] flatted the SPD representation to a vector via the tangent approximation and applied a multilayer perceptron for the classification task. Except for the tangent approximation, an alternative SPD manifold analysis approach is to embed SPD matrices into RKHS by a Mercer kernel function, where RKHS is also a Euclidean space. Wang *et al.* [10] and Vemulapalli *et al.* [11] utilized the LEM-based kernel function to embed the SPD representation into RKHS and applied Euclidean classifiers. Zhang *et al.* [12] proposed a Stein kernel function for the SPD representation, in which a learnable Stein divergence was exploited. In addition, the coding technique is a notable approach for various image processing and computer vision tasks. Harandi *et al.* [13] and Harandi and Salzmann [14] made use of kernel functions to embed the SPD representation into RKHS and performed coding and dictionary learning. These methods [7], [8], [10]–[14] all convert the SPD representation into a vector. Considering that the Riemannian space has a heterogeneous gap with the Euclidean vector space, the vector operation inevitably distorts the structure of the SPD matrix. In contrast, our similarity-based classifier does not address the input-lied space, and it needs a distance measure to compute similarities of pairs of samples. Our true geometry distance measure makes the classifier preserve the SPD manifold structure.

B. SPD Manifold Metric Learning

In general, there exist three categories of SPD manifold metric learning methods: learning a distance measure in the Euclidean tangent space, in the kernel space, and on other SPD manifolds.

In order to utilize the useful Euclidean metric, several works seek a more discriminative tangent space. Vemulapalli and Jacobs [23] proposed a metric learning method on the tangent space and learned a Mahalanobis metric. The drawback of this work is that the matrix vectorization distorts the matrix structure. Huang *et al.* [24] introduced the LEM learning (LEML) approach to transform the matrix on the tangent space to other tangent spaces. LEML not only avoids distorting the matrix structure but also brings high efficiency. Recently, Zhou *et al.* [25]

presented a sample-specific version of LEML named α-based covariance-like metric learning (α-CML) that learns to adjust eigenvalues of the SPD matrix for more discriminative power. In addition, some methods embed the SPD matrix into RKHS by a well-established kernel method. Quang *et al.* [26] generalized the LEM between two finite-dimensional SPD matrices to infinite-dimensional covariance matrices in RKHS by Hilbert–Schmidt operators. Several metric learning methods [2], [30] combine the discriminative powers of multiple types of manifold representations into RKHS. To overcome the inaccurate approximation of the Euclidean space and preserve the SPD manifold structure, Harandi *et al.* [27] proposed projecting the high-dimensional SPD matrix into a low-dimensional manifold and learning a metric in the new manifold.

The SPD manifold metric learning methods mentioned above treat the SPD matrix as a global representation. In contrast, we treat the SPD matrix as a combination of different visual information that has different characteristics for the distance measure. Our method discovers multiple visual information from the SPD representation via a point-to-set transformation and computes a set-to-set distance. Our set-to-set distance can be seen as a multi-metric learning problem, which has been used in various fields of the machine learning community, such as face and kinship verification [29], facial expression recognition [31], and multi-sensor fusion [32]. However, different from these methods, we are the first to learn multiple distance measures on Riemannian manifolds.

III. Preliminaries

This section provides a brief review of the SPD manifold. Note that throughout this article, vectors are represented by bold lower case letters, e.g., $\boldsymbol{v}$, and matrices are denoted by bold upper case letters, e.g., $\boldsymbol{W}$.

A. SPD Manifold

Given an SPD matrix $\boldsymbol{X} \in \mathbb{R}^{n\times n}$, the matrix satisfies $\boldsymbol{v}^\top \boldsymbol{X}\boldsymbol{v} > 0$, where $\boldsymbol{v} \in \mathbb{R}^n$ is any nonzero vector. Let us define the SPD manifold Sym_n^+ that consists of all $n \times n$ SPD matrices

$$\mathrm{Sym}_n^+ = \{\boldsymbol{X} \in \mathbb{R}^{n\times n} : \boldsymbol{v}^\top \boldsymbol{X}\boldsymbol{v} > 0 \quad \forall \boldsymbol{v} \in \mathbb{R}^n - \boldsymbol{0_n}\} \tag{1}$$

where $\mathbb{R}^n - \boldsymbol{0_n}$ is the $\mathbb{R}^n$ space without the zero vector.

Then, we introduce the definitions of the matrix logarithm function $\mathrm{logm}(\cdot)$ and the matrix exponential function $\mathrm{expm}(\cdot)$ [17], which is used later.

B. Matrix Logarithm Function $\mathrm{logm}(\boldsymbol{X})$

Given an SPD matrix $\boldsymbol{X} \in \mathbb{R}^{n\times n}$, the matrix logarithm function $\mathrm{logm}(\boldsymbol{X})$ is

$$\mathrm{logm}(\boldsymbol{X}) = \sum_{u=1}^{\infty} \frac{(-1)^{u-1}}{u}(\boldsymbol{X} - \boldsymbol{I}_n)^u = \boldsymbol{U}\mathrm{diag}(\log(\lambda_u))\boldsymbol{U}^\top \tag{2}$$

where $\top$ is the transpose operation, and $\boldsymbol{I}_n$ is an identity matrix with a size of $n \times n$. $\boldsymbol{X} = \boldsymbol{U}\mathrm{diag}(\lambda_u)\boldsymbol{U}^\top$ is the eigenvalue decomposition of $\boldsymbol{X}$, λ_u is the uth eigenvalue, and $\mathrm{diag}(\lambda_u)$ is a diagonal matrix whose elements are $\{\lambda_u\}_{i=1}^n$.

C. Matrix Exponential Function $\mathrm{expm}(\boldsymbol{X})$

Given an SPD matrix $\boldsymbol{X} \in \mathbb{R}^{n\times n}$, the matrix exponential function $\mathrm{expm}(\boldsymbol{X})$ is

$$\mathrm{expm}(\boldsymbol{X}) = \sum_{u=1}^{\infty} \frac{(-1)^{u-1}}{u}(\boldsymbol{X} - \boldsymbol{I}_n)^u = \boldsymbol{U}\mathrm{diag}(\exp(\lambda_u))\boldsymbol{U}^\top \tag{3}$$

D. Geometry of SPD Manifolds

The geometry of an SPD manifold is induced by the AIM [17], which is defined as

$$D^{\mathrm{AIM}}(\boldsymbol{X}, \boldsymbol{Y}) = \|\mathrm{logm}(\boldsymbol{X}^{-\frac{1}{2}}\boldsymbol{Y}\boldsymbol{X}^{-\frac{1}{2}})\|_F \tag{4}$$

where $\boldsymbol{X}, \boldsymbol{Y} \in \mathrm{Sym}_n^+$ are two SPD matrices, and $\|\cdot\|_F$ is the Frobenius norm of a matrix. Although AIM measures the true geometry of the SPD manifold, it has high computational complexity. Arsigny *et al.* [18] exploited the tangent space and introduced a lower computational cost SPD matrix metric named the LEM

$$D^{\mathrm{LEM}}(\boldsymbol{X}, \boldsymbol{Y}) = \|\mathrm{logm}(\boldsymbol{X}) - \mathrm{logm}(\boldsymbol{Y})\|_F \tag{5}$$

where the SPD matrix is projected into the tangent space by $\mathrm{logm}(\cdot)$. The tangent space is a Euclidean space; thus, the Euclidean metric can be applied directly. LEM is an approximate geodesic distance. In addition to AIM and LEM, there are several other representative SPD matrix distance measures, including the Stein divergence [19], the Jeffrey Kullback-Leibler divergence (Jeffrey KL divergence) [33], and the Burg matrix divergence [20].

For a specific task, due to the different data distributions, these SPD distance measures may have different performance [15]. It is impractical to choose an optimal SPD matrix distance measure after trial-and-error procedures. To solve this problem, Cichocki and Amari [34] proposed a learnable alpha–beta divergence, which can be adaptive to the underlying data distribution. For two SPD matrices $\boldsymbol{X}, \boldsymbol{Y} \in \mathrm{Sym}_n^+$, the alpha–beta divergence is defined as

$$\begin{aligned} D^{(\alpha,\beta)}(\boldsymbol{X}\|\boldsymbol{Y}) &= \frac{1}{\alpha\beta}\log\left(\det\left(\frac{\alpha(\boldsymbol{X}\boldsymbol{Y}^{-1})^{\beta} + \beta(\boldsymbol{X}\boldsymbol{Y}^{-1})^{-\alpha}}{\alpha+\beta}\right)\right) \\ &= \frac{1}{\alpha\beta}\sum_{u=1}^{n}\log\left(\frac{\alpha\lambda_u^{\beta} + \beta\lambda_u^{-\alpha}}{\alpha+\beta}\right) \\ &\qquad \alpha \neq 0,\ \beta \neq 0,\ \text{and}\ \alpha+\beta \neq 0 \end{aligned} \tag{6}$$

where (α, β) is the parameter, $\det(\cdot)$ denotes the determinant of a matrix, and λ_u is the uth eigenvalue of matrix $\boldsymbol{X}\boldsymbol{Y}^{-1}$. The alpha–beta divergence is a smooth function with respect to the parameter (α, β) [21]. With different α and β values, the alpha–beta divergence can be equivalent to other well-known SPD matrix distance measures, such as AIM and Stein divergence. Detailed proof can be found in [21]. In Table I, we show four special cases of alpha–beta divergence with corresponding values of α and β. $\boldsymbol{X}$ and $\boldsymbol{Y}$ are the input SPD matrices, λ_u is the uth eigenvalue of $\boldsymbol{X}\boldsymbol{Y}^{-1}$, $\mathrm{Tr}(\cdot)$ is the trace of a matrix, and $\det(\cdot)$ is the determinant of a matrix. For each special case, we provide its corresponding matrix computation and eigenvalue computation forms, which are equivalent. The alpha–beta divergence is

TABLE I
ALPHA–BETA DIVERGENCE WITH DIFFERENT (α, β) VALUES

(α, β) Value	Equivalent SPD Matrix Metric	Matrix Computation Form	Eigenvalue Computation Form
$(\alpha \to 0, \beta \to 0)$	Affine Invariant Metric (AIM) [17]	$\|\text{logm}(X^{-\frac{1}{2}}YX^{\frac{1}{2}})\|_F$	$\frac{1}{2}\sum_{u=1}^{n}\log^2\lambda_u$
$(\alpha = \pm\frac{1}{2}, \beta = \pm\frac{1}{2})$	Stein Divergence [19]	$4(\log(\det(\frac{X+Y}{2})) - \frac{1}{2}\log(det(XY)))$	$4\sum_{u=1}^{n}\log\frac{\lambda_u+1}{2\sqrt{\lambda_u}}$
$(\alpha = \pm 1, \beta \to 0)$	Jeffrey KL Divergence [33]	$\frac{1}{2}\text{Tr}(XY^{-1} + YX^{-1} - 2I)$	$\frac{1}{2}\sum_{u=1}^{n}(\sqrt{\lambda_u} - \frac{1}{\sqrt{\lambda_u}})^2$
$(\alpha = 1, \beta = 1)$	Burg Matrix Divergence [20]	$\text{Tr}(XY^{-1} - I) - \log(\det(XY^{-1}))$	$\sum_{u=1}^{n}(\lambda_u - \log(\lambda_u) - 1)$

invariant to the affine transformation, i.e., $D^{(\alpha,\beta)}(\boldsymbol{X}\|\boldsymbol{Y}) = D^{(\alpha,\beta)}(\boldsymbol{C}^\top\boldsymbol{X}\boldsymbol{C}\|\boldsymbol{C}^\top\boldsymbol{Y}\boldsymbol{C})$, where $\boldsymbol{X}, \boldsymbol{Y} \in \text{Sym}_n^+$, and $\boldsymbol{C} \in \mathbb{R}^{n\times n}$ is any invertible matrix. Theorem 1 gives the affine-invariance property of the alpha–beta divergence.

Theorem 1: Given two SPD matrices $\boldsymbol{X}, \boldsymbol{Y} \in \text{Sym}_n^+$, the alpha–beta divergence $D^{(\alpha,\beta)}(\boldsymbol{X}\|\boldsymbol{Y})$ is invariant to affine transformations, i.e., for any invertible matrix $\boldsymbol{C} \in \mathbb{R}^{n\times n}$, $D^{(\alpha,\beta)}(\boldsymbol{X}\|\boldsymbol{Y}) = D^{(\alpha,\beta)}(\boldsymbol{C}^\top\boldsymbol{X}\boldsymbol{C}\|\boldsymbol{C}^\top\boldsymbol{Y}\boldsymbol{C})$.

Proof: Through (6), we know that the alpha–beta divergence between two SPD matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$ is only related to the eigenvalues of $\boldsymbol{X}\boldsymbol{Y}^{-1}$. Thus, we carry out eigenvalue decomposition to matrix $\boldsymbol{X}\boldsymbol{Y}^{-1}$, $\boldsymbol{X}\boldsymbol{Y}^{-1} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^{-1}$, where $\boldsymbol{U}$ is the eigenvector matrix and $\boldsymbol{\Sigma}$ is the eigenvalue diagonal matrix. The simplification of $\boldsymbol{C}^\top\boldsymbol{X}\boldsymbol{C}(\boldsymbol{C}^\top\boldsymbol{Y}\boldsymbol{C})^{-1}$ is

$$\begin{aligned}\boldsymbol{C}^\top\boldsymbol{X}\boldsymbol{C}(\boldsymbol{C}^\top\boldsymbol{Y}\boldsymbol{C})^{-1} &= \boldsymbol{C}^\top\boldsymbol{X}\boldsymbol{C}\boldsymbol{C}^{-1}\boldsymbol{Y}^{-1}\boldsymbol{C}^{-\top} = \boldsymbol{C}^\top\boldsymbol{X}\boldsymbol{Y}^{-1}\boldsymbol{C}^{-\top}\\ &= \boldsymbol{C}^\top\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^{-1}\boldsymbol{C}^{-\top} = \boldsymbol{C}^\top\boldsymbol{U}\boldsymbol{\Sigma}(\boldsymbol{C}^\top\boldsymbol{U})^{-1}.\end{aligned} \tag{7}$$

As U and C are both invertible matrices, $C^\top U$ is also an invertible matrix, and $\boldsymbol{C}^\top\boldsymbol{U}\boldsymbol{\Sigma}(\boldsymbol{C}^\top\boldsymbol{U})^{-1}$ is similar to the diagonal matrix $\boldsymbol{\Sigma}$. Therefore, $\boldsymbol{\Sigma}$ is also the eigenvalue diagonal matrix of $\boldsymbol{C}^\top\boldsymbol{X}\boldsymbol{C}(\boldsymbol{C}^\top\boldsymbol{Y}\boldsymbol{C})^{-1}$, $\boldsymbol{X}\boldsymbol{Y}^{-1}$ and $\boldsymbol{C}^\top\boldsymbol{X}\boldsymbol{C}(\boldsymbol{C}^\top\boldsymbol{Y}\boldsymbol{C})^{-1}$ have the same eigenvalues, and

$$D^{(\alpha,\beta)}(\boldsymbol{X}\|\boldsymbol{Y}) = D^{(\alpha,\beta)}(\boldsymbol{C}^\top\boldsymbol{X}\boldsymbol{C}\|\boldsymbol{C}^\top\boldsymbol{Y}\boldsymbol{C}). \tag{8}$$

□

IV. PROPOSED SPD DISTANCE MEASURE

A. Problem Definition

We first introduce several mathematical symbols used in our method. The low-dimensional projection $f_{\boldsymbol{W}}(\cdot)$ projects the SPD representation to another manifold, and $\boldsymbol{W}$ is the projection matrix.

The subdistance measure $g_{\boldsymbol{A}}(\cdot,\cdot)$ is a distance measure on a projected low-dimensional manifold, where $\boldsymbol{A}$ is the parameter.

The integration function $h_{\boldsymbol{M}}(\cdot)$ integrates multiple subdistances into one, and $\boldsymbol{M}$ is the integration parameter. The point-to-set transformation $T_s(\cdot)$ projects the input SPD representation to m low-dimensional SPD manifolds.

The set-to-set distance $D_s(\cdot,\cdot)$ is adopted to compute the distance between two SPD sets.

Our goal is to build a robust SPD distance measure, where different characteristics of visual information in the SPD representation are considered. More specifically, our distance measure is composed of a point-to-set transformation $T_s(\cdot)$ and a set-to-set distance measure $D_s(\cdot,\cdot)$, named point-to-set and set-to-set distance measure (PSSSD). Given two SPD representations $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$, we discover discriminative visual information from each SPD representation by $T_s(\cdot)$ which contains m low-dimensional projections $\{f_{\boldsymbol{W}}^k(\cdot)\}_{k=1}^m$. The obtained low-dimensional SPD matrices from one original SPD representation constitute an SPD set $\mathcal{X}_i = \{f_{\boldsymbol{W}}^k(\boldsymbol{X}_i)\}_{k=1}^m$. Then, the distance measure between two original SPD representations is converted to a set-to-set distance measure $D_s(\cdot,\cdot)$. $D_s(\cdot,\cdot)$ integrates m individual subdistance measures $g_{\boldsymbol{A}}^k(\cdot,\cdot)$ between the pair of low-dimensional matrices projected from each involved original SPD representation. Due to different distributions of visual information, we assign the learnable alpha–beta divergence as $g_{\boldsymbol{A}}^k(\cdot,\cdot)$ on different low-dimensional manifolds. To consider different contributions of visual information, the m subdistances are integrated by $h_{\boldsymbol{M}}(\cdot)$ with different weights. The three parameters $\boldsymbol{W}, \boldsymbol{A}$, and $\boldsymbol{M}$ are represented by a parameter set $\Theta = \{\boldsymbol{W}, \boldsymbol{A}, \boldsymbol{M}\}$. Our SPD distance measure PSSSD described above is defined in the following, and the framework is shown in Fig. 1.

Definition 1: Given two SPD representations $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$, our distance $D^\Theta(\boldsymbol{X}_i, \boldsymbol{X}_j)$ is defined as

$$\begin{aligned}&D^\Theta(\boldsymbol{X}_i, \boldsymbol{X}_j)\\ &= D_s(T_s(\boldsymbol{X}_i), T_s(\boldsymbol{X}_j)) = D_s(\mathcal{X}_i, \mathcal{X}_j)\\ &= D_s\big(\{f_{\boldsymbol{W}}^1(\boldsymbol{X}_i), \cdots, f_{\boldsymbol{W}}^m(\boldsymbol{X}_i)\}, \{f_{\boldsymbol{W}}^1(\boldsymbol{X}_j), \cdots, f_{\boldsymbol{W}}^m(\boldsymbol{X}_j)\}\big)\\ &= h_{\boldsymbol{M}}\big(g_{\boldsymbol{A}}^1\big(f_{\boldsymbol{W}}^1(\boldsymbol{X}_i), f_{\boldsymbol{W}}^1(\boldsymbol{X}_j)\big), \cdots, g_{\boldsymbol{A}}^m\big(f_{\boldsymbol{W}}^m(\boldsymbol{X}_i), f_{\boldsymbol{W}}^m(\boldsymbol{X}_j)\big)\big).\end{aligned} \tag{9}$$

The similarity-based classifier learns the distance measure defined in (9). We define the objective function $\mathcal{L}(\Theta, \mathcal{S}, \mathcal{D}, Y)$ on the parameter Θ, the SPD matrix similar pair set $\mathcal{S}$, the SPD dissimilar pair set $\mathcal{D}$, and their labels Y, where $y_{ij} = 1$ means that $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$ are similar; otherwise, $y_{ij} = 0$. Via the training strategy that utilizes sample pairs, the similarity information is embedded into the model [35].

Definition 2: The objective function $\mathcal{L}(\Theta, \mathcal{S}, \mathcal{D}, Y)$ is defined as

$$\begin{aligned}&\mathcal{L}(\Theta, \mathcal{S}, \mathcal{D}, Y)\\ &\quad= \frac{1}{|\mathcal{S}|}\sum_{i,j\in\mathcal{S}} y_{ij}\cdot\max(D^\Theta(\boldsymbol{X}_i, \boldsymbol{X}_j) - \zeta_s, 0)^2\\ &\qquad+ \frac{1}{|\mathcal{D}|}\sum_{i,j\in\mathcal{D}} (1 - y_{ij})\cdot\max(\zeta_d - D^\Theta(\boldsymbol{X}_i, \boldsymbol{X}_j), 0)^2\\ &\qquad+ \xi\cdot\gamma(\boldsymbol{M}), \text{ s.t. } \boldsymbol{M} \in \text{Sym}_m^+, \quad \boldsymbol{W} \in \text{St}(mp, n).\end{aligned} \tag{10}$$

We expect that the distance between two similar SPD representations is smaller than a threshold ζ_s, and the distance

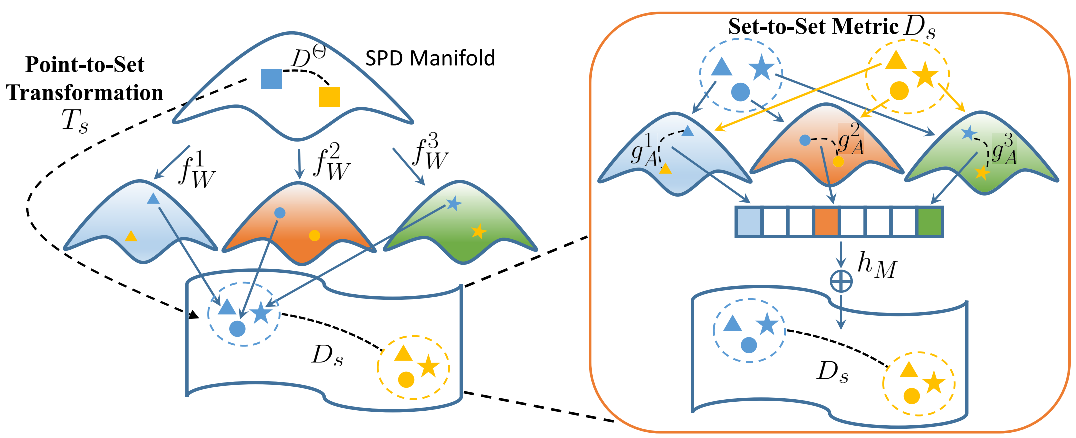


Fig. 1. Flowchart of the proposed distance measure PSSSD between two SPD representations. Different colors indicate different SPD representations, and different shapes indicate SPD matrices on different low-dimensional manifolds. The point-to-set transformation $T_s(\cdot)$ contains multiple projections, $f_{\boldsymbol{W}}^1$, $f_{\boldsymbol{W}}^2$, and $f_{\boldsymbol{W}}^3$, to discover multiple types of visual information. The distance $D^{\Theta}(\cdot,\cdot)$ between two SPD representations is converted to a set-to-set distance $D_s(\cdot,\cdot)$. The detailed set-to-set distance $D_s(\cdot,\cdot)$ is shown in the right part. $g_{\boldsymbol{A}}^1$, $g_{\boldsymbol{A}}^2$, and $g_{\boldsymbol{A}}^3$ are the subdistance measures on low-dimensional manifolds, and $h_{\boldsymbol{M}}$ integrates these subdistances. $\Theta = \{\boldsymbol{W}, \boldsymbol{A}, \boldsymbol{M}\}$ is the learnable parameter set. The point-to-set transformation and set-to-set distance are optimized jointly.

between two dissimilar SPD representations is larger than a threshold ζ_d. We add two coefficients $(1/|\mathcal{S}|)$ and $(1/|\mathcal{D}|)$ to solve the imbalance issue of similar and dissimilar pairs, where $|\mathcal{S}|$ and $|\mathcal{D}|$ are the pair numbers of sets $\mathcal{S}$ and $\mathcal{D}$. We impose the orthogonality constraint on $\boldsymbol{W}$ and the positive-definite constraint on $\boldsymbol{M}$ to obtain a more robust distance measure. Moreover, to avoid overfitting and incorporate prior information with the set-to-set distance [36], [37], we employ the Burgman matrix divergence [20] between $\boldsymbol{M}$ and a prior matrix $\boldsymbol{M}_0$ as the regularization in the objective function in (10), i.e., $\gamma(\boldsymbol{M}) = \mathrm{Tr}(\boldsymbol{M}\boldsymbol{M}_0^{-1}) - \log\det(\boldsymbol{M}\boldsymbol{M}_0^{-1}) - m$. Due to the manifold constraints of W and M, they can be learned via Riemannian optimization methods effectively and efficiently. In Sections IV-B and IV-C, we explain the point-to-set transformation $T_s(\cdot)$ and the set-to-set distance $D_s(\cdot)$.

B. Point-to-Set Transformation

Our point-to-set transformation $T_s(\cdot)$ applies the bilinear projection form to project the original SPD matrix $\boldsymbol{X}_i \in \mathbb{R}^{n\times n}$ into m low-dimensional manifolds

$$\begin{aligned} \boldsymbol{X}_i^1 &= f_{\boldsymbol{W}}^1(\boldsymbol{X}_i) = \boldsymbol{W}_1^\top \boldsymbol{X}_i \boldsymbol{W}_1 \\ &\cdots \\ \boldsymbol{X}_i^m &= f_{\boldsymbol{W}}^m(\boldsymbol{X}_i) = \boldsymbol{W}_m^\top \boldsymbol{X}_i \boldsymbol{W}_m \end{aligned} \tag{11}$$

where $\boldsymbol{X}_i^k \in \mathbb{R}^{p\times p}$ is on the kth low-dimensional SPD manifold, $k \in \{1, 2, \cdots, m\}$, p is the size of the low-dimensional matrix, $f_{\boldsymbol{W}}^k(\cdot)$ is the kth low-dimensional manifold projection, and $\boldsymbol{W}_k \in \mathbb{R}^{n\times p}$ is the kth low-dimensional manifold projection matrix. We combine the m low-dimensional SPD matrices, which constitutes an SPD set $\mathcal{X}_i = \{\boldsymbol{X}_i^k\}_{k=1}^m$. We expect that each low-dimensional matrix $\boldsymbol{X}_i^k$ is guaranteed to still be an SPD matrix with the ability to capture the desirable feature distribution. Based on the linear algebraic theory, a column full-rank matrix $\boldsymbol{W}_k$ guarantees that $\boldsymbol{W}_k^\top \boldsymbol{X}_k \boldsymbol{W}_k$ is an SPD matrix.

The alpha–beta divergence is applied to our low-dimensional manifolds (detailed in Section IV-C). To enforce the requirement that $\boldsymbol{W}_k$ needs to be a column full-rank matrix, we can impose the orthogonality constraint on $\boldsymbol{W}_k$, without loss of generality, which is shown in Theorem 2.

Theorem 2: Given two SPD matrices $\boldsymbol{X}, \boldsymbol{Y} \in \mathrm{Sym}_n^+$, and $\tilde{\boldsymbol{W}} \in \mathbb{R}^{n\times p}$ is a projection matrix with the column full-rank constraint, where $n \geq p$. $D^{(\alpha,\beta)}(\tilde{\boldsymbol{W}}^\top \boldsymbol{X}\tilde{\boldsymbol{W}} \| \tilde{\boldsymbol{W}}^\top \boldsymbol{Y}\tilde{\boldsymbol{W}})$ is the alpha–beta divergence between projected SPD matrices. To optimize any objective function $\mathcal{L}(D^{(\alpha,\beta)}(\tilde{\boldsymbol{W}}^\top \boldsymbol{X}\tilde{\boldsymbol{W}} \| \tilde{\boldsymbol{W}}^\top \boldsymbol{Y}\tilde{\boldsymbol{W}}))$, we can impose the orthogonality constraint on $\boldsymbol{W}_k$, without loss of generality.

Proof: Based on Theorem 1, the alpha–beta divergence is an affine-invariant divergence, i.e., $D^{(\alpha,\beta)}(\boldsymbol{X}\|\boldsymbol{Y}) = D^{(\alpha,\beta)}(\boldsymbol{C}^\top\boldsymbol{X}\boldsymbol{C}\|\boldsymbol{C}^\top\boldsymbol{Y}\boldsymbol{C})$, where $\boldsymbol{X}, \boldsymbol{Y} \in \mathrm{Sym}_n^+$, and $\boldsymbol{C} \in \mathbb{R}^{n\times n}$ is any invertible matrix. Note that, any column full-rank matrix $\tilde{\boldsymbol{W}} \in \mathbb{R}^{n\times p}$ can be factorized into an orthogonal matrix $\boldsymbol{W} \in \mathbb{R}^{n\times p}$ and an upper triangular matrix $\boldsymbol{B} \in \mathbb{R}^{p\times p}$ by the orthogonal-upper-triangular decomposition (QR decomposition), i.e., $\tilde{\boldsymbol{W}} = \boldsymbol{W}\boldsymbol{B}$. As the upper triangular matrix $\boldsymbol{B}$ is a special form of the invertible matrix, resorting to the affine-invariance property, the alpha–beta divergence between $\tilde{\boldsymbol{W}}^\top \boldsymbol{X}\tilde{\boldsymbol{W}}$ and $\tilde{\boldsymbol{W}}^\top \boldsymbol{Y}\tilde{\boldsymbol{W}}$ can be translated to

$$\begin{aligned} &D^{(\alpha,\beta)}(\tilde{\boldsymbol{W}}^\top \boldsymbol{X}\tilde{\boldsymbol{W}} \| \tilde{\boldsymbol{W}}^\top \boldsymbol{Y}\tilde{\boldsymbol{W}}) \\ &\quad= D^{(\alpha,\beta)}(\boldsymbol{B}^\top\boldsymbol{W}^\top\boldsymbol{X}\boldsymbol{W}\boldsymbol{B} \| \boldsymbol{B}^\top\boldsymbol{W}^\top\boldsymbol{Y}\boldsymbol{W}\boldsymbol{B}) \\ &\quad= D^{(\alpha,\beta)}(\boldsymbol{W}^\top\boldsymbol{X}\boldsymbol{W} \| \boldsymbol{W}^\top\boldsymbol{Y}\boldsymbol{W}). \end{aligned} \tag{12}$$

Thus, for any column full rank matrix $\tilde{\boldsymbol{W}}$, we can always find an orthogonal matrix $\boldsymbol{W}$ and guarantee that $\mathcal{L}(D^{(\alpha,\beta)}(\tilde{\boldsymbol{W}}^\top \boldsymbol{X}\tilde{\boldsymbol{W}} \| \tilde{\boldsymbol{W}}^\top \boldsymbol{Y}\tilde{\boldsymbol{W}})) = \mathcal{L}(D^{(\alpha,\beta)}(\boldsymbol{W}^\top\boldsymbol{X}\boldsymbol{W} \| \boldsymbol{W}^\top\boldsymbol{Y}\boldsymbol{W}))$. $\square$

The orthogonality constraint makes $\boldsymbol{W}_k$ on a Stiefel manifold [1]. However, an orthogonal matrix $\boldsymbol{R} \in \mathbb{R}^{p\times p}$ is also an invertible matrix. Resorting to the affine-invariance property, we can obtain $D^{(\alpha,\beta)}(\boldsymbol{R}^\top\boldsymbol{W}_k^\top\boldsymbol{X}_i\boldsymbol{W}_k\boldsymbol{R} \| \boldsymbol{R}^\top\boldsymbol{W}_k^\top\boldsymbol{X}_j\boldsymbol{W}_k\boldsymbol{R}) = D^{(\alpha,\beta)}(\boldsymbol{W}_k^\top\boldsymbol{X}_i\boldsymbol{W}_k \| \boldsymbol{W}_k^\top\boldsymbol{X}_j\boldsymbol{W}_k)$. The objective function is invariant to the right action of any orthogonal matrix on $\boldsymbol{W}_k$. Thus, the projection parameter $\boldsymbol{W}_k$ with the orthogonality constraint is on a Grassmannian manifold [1], which is a quotient space of the Stiefel manifold. To reduce the relevance between any two low-dimensional SPD matrices, we expect that projection parameters $\{\boldsymbol{W}_k\}_{k=1}^m$ are diverse and as large as possible. When we add the orthogonality constraint between two projection parameters $\boldsymbol{W}_k$ and $\boldsymbol{W}_l$, the distance between $\boldsymbol{W}_k$ and $\boldsymbol{W}_l$ reaches the maximum, as $p - \|\boldsymbol{W}_k^\top \boldsymbol{W}_l\|_F$ is the projection distance metric on the Grassmannian manifold [38]. Thus, for any $k \neq l$, we force $\boldsymbol{W}_k^\top \boldsymbol{W}_l = \boldsymbol{0}$, where $\boldsymbol{0} \in \mathbb{R}^{p\times p}$ is a matrix whose elements are all "0"s.

All $\boldsymbol{W}_k$ can compose a total projection matrix $\boldsymbol{W}$, i.e., $\boldsymbol{W} = [\boldsymbol{W}_1, \boldsymbol{W}_2, \cdots, \boldsymbol{W}_m] \in \mathbb{R}^{n\times mp}$, in which $\boldsymbol{W}_k$ is a partitioned matrix of $\boldsymbol{W}$ containing p columns. Note that $\boldsymbol{W}$ is a subunitary matrix as well, i.e., $\boldsymbol{W}^\top \boldsymbol{W} = \boldsymbol{I}_{mp}$, and it is on the Stiefel manifold [1]. We find that, based on Remark 1, our SPD set $\mathcal{X}_i = \{\boldsymbol{X}_i^k\}_{k=1}^m$ is equivalent to a diagonal block matrix $\boldsymbol{Z}_i$, which has a clear matrix structure and simple matrix computation.

Remark 1: In our method, all low-dimensional manifold projection matrices $\{\boldsymbol{W}_k \in \mathbb{R}^{n\times p}\}_{k=1}^m$ in (11) can constitute a matrix $\boldsymbol{W} \in \mathbb{R}^{n\times mp}$, $k \in [1, m]$. For an original SPD representation $\boldsymbol{X}_i \in \mathbb{R}^{n\times n}$, if we apply a diagonal block binary mask on the matrix $\boldsymbol{W}^\top \boldsymbol{X}_i \boldsymbol{W}$, we can obtain a diagonal block matrix $\boldsymbol{Z}_i \in \mathbb{R}^{mp\times mp}$. $\boldsymbol{Z}_i$ is equivalent to the SPD set $\mathcal{X}_i = \{\boldsymbol{X}_i^k \in \mathbb{R}^{p\times p}\}_{k=1}^m$, and diagonal blocks of $\boldsymbol{Z}_i$ are the same low-dimensional SPD matrices as $\{\boldsymbol{X}_i^k\}_{k=1}^m$ in (11), that is,

$$\begin{aligned}\boldsymbol{Z}_i &= \operatorname{mask}(\boldsymbol{W}^\top \boldsymbol{X}_i \boldsymbol{W}) \\ &= \operatorname{mask}\left(\begin{bmatrix} \boldsymbol{W}_1^\top \boldsymbol{X}_i \boldsymbol{W}_1 & \cdots & \boldsymbol{W}_1^\top \boldsymbol{X}_i \boldsymbol{W}_m \\ \vdots & \vdots & \vdots \\ \boldsymbol{W}_m^\top \boldsymbol{X}_i \boldsymbol{W}_1 & \cdots & \boldsymbol{W}_m^\top \boldsymbol{X}_i \boldsymbol{W}_m \end{bmatrix}\right) \\ &= \begin{bmatrix} \boldsymbol{X}_i^1 & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{X}_i^2 & \boldsymbol{0} \\ \vdots & \vdots & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{X}_i^m \end{bmatrix}. \end{aligned} \tag{13}$$

From this point, we transform the original SPD representation $\boldsymbol{X}_i$ to a diagonal block matrix $\boldsymbol{Z}_i$, and the diagonal blocks represent the discovered visual information. The diagonal block matrix clears the matrix structure and reduces unimportant information. In addition, it is more efficient to implement the matrix eigenvalue decomposition and matrix inversion operations on $\boldsymbol{Z}_i$ than $\boldsymbol{W}^\top \boldsymbol{X}_i \boldsymbol{W}$.

C. Set-to-Set Distance

Based on the point-to-set transformation, the distance $D^\Theta(\boldsymbol{X}_i, \boldsymbol{X}_j)$ between two SPD matrices is transformed to a set-to-set distance $D_s(\mathcal{X}_i, \mathcal{X}_j)$. We design the set-to-set distance $D_s(\cdot,\cdot)$ as the integration $h_{\boldsymbol{M}}(\cdot)$ of subdistances $\{g_{\boldsymbol{A}}^k(\cdot,\cdot)\}_{k=1}^m$, which are assigned on m low-dimensional SPD manifolds, where $\boldsymbol{A}$ and $\boldsymbol{M}$ are learnable parameters.

On the kth low-dimensional manifold, we apply an individual learnable alpha–beta divergence [21], [22] as the subdistance measures $g_{\boldsymbol{A}}^k(\cdot,\cdot)$. Thus, the distance d_{ij}^k between a pair of $\boldsymbol{X}_i^k$ and $\boldsymbol{X}_j^k$ in two SPD sets $\mathcal{X}_i$ and $\mathcal{X}_j$ is given by

$$\begin{aligned} d_{ij}^k &= g_{\boldsymbol{A}}^k\big(\boldsymbol{X}_i^k, \boldsymbol{X}_j^k\big) = D^{(\alpha_k,\beta_k)}\big(\boldsymbol{X}_i^k \| \boldsymbol{X}_j^k\big) \\ &= \frac{1}{\alpha_k\beta_k} \sum_{u=1}^{p} \log\left(\frac{\alpha_k\big(\lambda_{iju}^k\big)^{\beta_k} + \beta_k\big(\lambda_{iju}^k\big)^{-\alpha_k}}{\alpha_k+\beta_k}\right) \end{aligned} \tag{14}$$

where λ_{iju}^k is the uth eigenvalue of matrix $\boldsymbol{X}_i^k(\boldsymbol{X}_j^k)^{-1}$, and (α_k, β_k) is the individual parameter of the kth alpha–beta divergence. We denote all alpha–beta divergence parameters as a matrix $\boldsymbol{A} = [(\alpha_1,\beta_1), (\alpha_2,\beta_2), \ldots, (\alpha_m,\beta_m)] \in \mathbb{R}^{m\times 2}$ and a distance vector between $\mathcal{X}_i$ and $\mathcal{X}_j$ as $\boldsymbol{d}_{ij} = [d_{ij}^1, d_{ij}^2, \ldots, d_{ij}^m] \in \mathbb{R}^{m\times 1}$. Because (α_k, β_k) needs to be adaptive to the kth low-dimensional manifold, we exploit a learnable strategy to update (α_k, β_k), which is detailed in Section V.

We argue that subdistances and their relationships all contribute to the SPD distance. Thus, the set-to-set distance $D_s(\mathcal{X}_i, \mathcal{X}_j)$ is designed as a weight sum formulation

$$\begin{aligned} &D^\Theta(\boldsymbol{X}_i, \boldsymbol{X}_j) \\ &\quad = D_s(\mathcal{X}_i, \mathcal{X}_j) = h_{\boldsymbol{M}}\big(d_{ij}^1, d_{ij}^2, \ldots, d_{ij}^m\big) = \boldsymbol{d}_{ij}^\top \boldsymbol{M} \boldsymbol{d}_{ij} \\ &\quad = \sum_{k=1}^m \sum_{l=1}^m \big(d_{ij}^k \cdot M_{kl} \cdot d_{ij}^l\big) \end{aligned} \tag{15}$$

where $\boldsymbol{M} \in \mathbb{R}^{m\times m}$ is the integration parameter, M_{kl} is the element of $\boldsymbol{M}$ in the kth row and lth column, reflecting contributions of visual information and their relationships.

Via (15), we know that the designed set-to-set distance is a quadratic form. If $\boldsymbol{X}_i = \boldsymbol{X}_j$, then $\boldsymbol{d}_{ij}$ is a zero vector, and $D^\Theta(\boldsymbol{X}_i, \boldsymbol{X}_j) = 0$. If $\boldsymbol{X}_i \neq \boldsymbol{X}_j$, then $\boldsymbol{d}_{ij}$ is a nonzero vector, and $D^\Theta(\boldsymbol{X}_i, \boldsymbol{X}_j)$ should be larger than 0. Thus, the nonnegativity of the distance forces $\boldsymbol{M}$ to be an SPD matrix [37], [39], [40]. PSSSD can be analogous to the Mahalanobis metric, where each vector dimension has its own distribution and contribution for the Mahalanobis metric, and our low-dimensional manifolds can be seen as analogies of different vector dimensions. In addition, to explain our distance more clearly, based on Theorem 3, the set-to-set distance $D_s(\mathcal{X}_i, \mathcal{X}_j)$ in (15) can be derived from the squared alpha–beta divergence and is a weighted version.

Theorem 3: The set-to-set distance $D_s(\mathcal{X}_i, \mathcal{X}_j)$ in (15) can be derived from the squared alpha–beta divergence. Specifically, the set-to-set distance is a weighted version of the squared alpha–beta divergence.

Proof: Based on Remark 1, the SPD set $\mathcal{X}_i = \{\boldsymbol{X}_i^k\}_{k=1}^m$ can be established as a diagonal block matrix $\boldsymbol{Z}_i \in \mathbb{R}^{mp\times mp}$, and the diagonal block $\boldsymbol{X}_i^k \in \mathbb{R}^{p\times p}$ is the low-dimensional SPD matrix, representing discovered visual information, that is,

$$\boldsymbol{Z}_i = \begin{bmatrix} \boldsymbol{X}_i^1 & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{X}_i^2 & \boldsymbol{0} \\ \vdots & \vdots & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{X}_i^m \end{bmatrix}. \tag{16}$$

We can utilize the squared alpha–beta divergence to measure the distance between diagonal block matrices Z_i and Z_j, which is

$$D^2 = (D^{(\alpha,\beta)}(\boldsymbol{Z}_i \| \boldsymbol{Z}_j))^2 = \left(\frac{1}{\alpha\beta}\sum_{u=1}^{mp} \log\left(\frac{\alpha\lambda_u^\beta + \beta\lambda_u^{-\alpha}}{\alpha+\beta}\right)\right)^2 \tag{17}$$

where λ_u is the uth eigenvalue of the matrix $\boldsymbol{Z}_i\boldsymbol{Z}_j^{-1}$. Because $\boldsymbol{Z}_i$ and $\boldsymbol{Z}_j$ are diagonal block matrices, based on (13), the matrix inversion operation is flexible. $\boldsymbol{Z}_i\boldsymbol{Z}_j^{-1}$ can be calculated by

$$\boldsymbol{Z}_i\boldsymbol{Z}_j^{-1} = \begin{bmatrix} \boldsymbol{X}_i^1(\boldsymbol{X}_j^1)^{-1} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{X}_i^2(\boldsymbol{X}_j^2)^{-1} & \boldsymbol{0} \\ \vdots & \vdots & \vdots \\ \boldsymbol{0} & \cdots & \boldsymbol{X}_i^m(\boldsymbol{X}_j^m)^{-1} \end{bmatrix} \tag{18}$$

and $\{\lambda_u\}_{u=(k-1)\times p+1}^{k\times p}$ are the eigenvalues of matrix $\boldsymbol{X}_i^k(\boldsymbol{X}_j^k)^{-1}$. We rewrite (17) as

$$D^2 = \left(\sum_{k=1}^{m}\left(\frac{1}{\alpha\beta}\sum_{u=(k-1)\times p+1}^{k\times p}\log\left(\frac{\alpha\lambda_u^{\beta}+\beta\lambda_u^{-\alpha}}{\alpha+\beta}\right)\right)\right)^2. \tag{19}$$

Note that $(1/\alpha\beta)\sum_{u=(k-1)\times p+1}^{k\times p}\log((\alpha\lambda_u^{\beta}+\beta\lambda_u^{-\alpha}/\alpha+\beta))$ is the alpha–beta divergence between $\boldsymbol{X}_i^k$ and $\boldsymbol{X}_j^k$. We denote it as d^k, and the squared distance is

$$D^2 = \left(\sum_{k=1}^{m} d^k\right)^2 = \sum_{k=1}^{m}\sum_{l=1}^{m} d^k d^l. \tag{20}$$

If we assign individual (α_k, β_k) to each d^k to allow for different distributions of diagonal blocks, d^k is equivalent to our subdistance $g_A^k(\cdot,\cdot)$ in (14)

$$d^k = \frac{1}{\alpha_k\beta_k}\sum_{u=(k-1)\times p+1}^{k\times p}\log\left(\frac{\alpha_k\lambda_u^{\beta_k}+\beta_k\lambda_u^{-\alpha_k}}{\alpha_k+\beta_k}\right). \tag{21}$$

If we add different weights w_k to d^k to consider different contributions of diagonal blocks for the distance, D^2 is equivalent to our set-to-set distance in (15)

$$D^2 = \left(\sum_{k=1}^{m} w_k d^k\right)^2 = \sum_{k=1}^{m}\sum_{l=1}^{m} w_k w_l d^k d^l = \sum_{k=1}^{m}\sum_{l=1}^{m} M_{kl} d^k d^l. \tag{22}$$

Thus, our set-to-set distance can be derived from a squared alpha–beta divergence with multiple individual parameters and different weights on diagonal block matrices. □

V. Optimization Method

$\mathcal{L}(\Theta, \mathcal{S}, \mathcal{D}, Y)$ in (10) is not jointly convex with respect to all its learning parameters $\Theta = \{\boldsymbol{W}, \boldsymbol{A}, \boldsymbol{M}\}$. Furthermore, $\boldsymbol{W}$ and $\boldsymbol{M}$ lie on two different Riemannian manifolds and directly utilizing the Euclidean optimization methods (i.e., projected methods) may lead to undesirable behaviors. In our method, parameters $\boldsymbol{A}$, $\boldsymbol{W}$, and $\boldsymbol{M}$ are optimized by the stochastic gradient descent (SGD) method. $\boldsymbol{A}$ is updated by the Euclidean gradient, and $\boldsymbol{W}$ and $\boldsymbol{M}$ are updated by Riemannian gradients due to their manifold constraints. As the number of all pairs can be large, to speed up the training, we update parameters via small minibatches of pairs instead of the whole. To be specific, we first compute the gradient with respect to $\boldsymbol{W}$, $\boldsymbol{A}$, and $\boldsymbol{M}$ as follows.

A. Gradient of $\mathcal{L}$ With Respect to $\boldsymbol{M}$

The gradient of $\mathcal{L}$ with respect to $\boldsymbol{M}$ can be computed by

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{M}} = \frac{1}{|\mathcal{S}|}\sum_{i,j\in\mathcal{S}}\boldsymbol{d}_{ij}\frac{\partial\mathcal{L}}{\partial D_{ij}^{\Theta}}\boldsymbol{d}_{ij}^{\top} + \frac{1}{|\mathcal{D}|}\sum_{i,j\in\mathcal{D}}\boldsymbol{d}_{ij}\frac{\partial\mathcal{L}}{\partial D_{ij}^{\Theta}}\boldsymbol{d}_{ij}^{\top} + \xi\cdot\frac{\partial\gamma(\boldsymbol{M},\boldsymbol{M}_0)}{\partial\boldsymbol{M}} \tag{23}$$

where $(\partial\mathcal{L}/\partial D_{ij}^{\Theta})$ is the gradient of $\mathcal{L}$ with respect to $D^{\Theta}(\boldsymbol{X}_i, \boldsymbol{X}_j)$

$$\frac{\partial\mathcal{L}}{\partial D_{ij}^{\Theta}} = 2\cdot y_{ij}\cdot\max\left(D_{ij}^{\Theta}-\zeta_s, 0\right) + 2\cdot(y_{ij}-1)\cdot\max\left(\zeta_d - D_{ij}^{\Theta}, 0\right) \tag{24}$$

and $(\partial\gamma(\boldsymbol{M},\boldsymbol{M}_0)/\partial\boldsymbol{M})$ is the gradient of $\gamma(\boldsymbol{M},\boldsymbol{M}_0)$ with respect to $\boldsymbol{M}$

$$\frac{\partial\gamma(\boldsymbol{M},\boldsymbol{M}_0)}{\partial\boldsymbol{M}} = \boldsymbol{M}_0^{-1} - \boldsymbol{M}^{-1}. \tag{25}$$

B. Gradient of $\mathcal{L}$ With Respect to $\boldsymbol{A}$

The gradients of $\mathcal{L}$ with respect to α_k and β_k in $\boldsymbol{A}$ are

$$\frac{\partial\mathcal{L}}{\partial\alpha_k} = \frac{1}{|\mathcal{S}|}\sum_{i,j\in\mathcal{S}}\frac{\partial\mathcal{L}}{\partial d_{ij}^k}\cdot\frac{\partial d_{ij}^k}{\partial\alpha_k} + \frac{1}{|\mathcal{D}|}\sum_{i,j\in\mathcal{D}}\frac{\partial\mathcal{L}}{\partial d_{ij}^k}\cdot\frac{\partial d_{ij}^k}{\partial\alpha_k} \tag{26}$$

$$\frac{\partial\mathcal{L}}{\partial\alpha_k} = \frac{1}{|\mathcal{S}|}\sum_{i,j\in\mathcal{S}}\frac{\partial\mathcal{L}}{\partial d_{ij}^k}\cdot\frac{\partial d_{ij}^k}{\partial\beta_k} + \frac{1}{|\mathcal{D}|}\sum_{i,j\in\mathcal{D}}\frac{\partial\mathcal{L}}{\partial d_{ij}^k}\cdot\frac{\partial d_{ij}^k}{\partial\beta_k}. \tag{27}$$

$(\partial\mathcal{L}/\partial d_{ij}^k)$ is the kth element of $(\partial\mathcal{L}/\partial\boldsymbol{d}_{ij})$, which is the gradient of $\mathcal{L}$ with respect to $\boldsymbol{d}_{ij}$

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{d}_{ij}} = \frac{\partial\mathcal{L}}{\partial D_{ij}^{\Theta}}\cdot\frac{\partial D_{ij}^{\Theta}}{\partial\boldsymbol{d}_{ij}} = \frac{\partial\mathcal{L}}{\partial D_{ij}^{\Theta}}\boldsymbol{d}_{ij}^{\top}(M^{\top}+M). \tag{28}$$

$(\partial d_{ij}^k/\partial\alpha_k)$ and $(\partial d_{ij}^k/\partial\beta_k)$ are the gradients of d_{ij}^k with respect to α_k and β_k, respectively,

$$\frac{\partial d_{ij}^k}{\partial\alpha_k} = \frac{1}{\alpha_k^2\beta_k}\sum_{u=1}^{p}\left(\frac{\alpha_k\left(\lambda_{iju}^k\right)^{\beta_k} - \alpha_k\beta_k\left(\lambda_{iju}^k\right)^{-\alpha_k}\log\lambda_{iju}^k}{\alpha_k\left(\lambda_{iju}^k\right)^{\beta_k}+\beta_k\left(\lambda_{iju}^k\right)^{-\alpha_k}} - \frac{\alpha_k}{\alpha_k+\beta_k} - \log\frac{\alpha_k\left(\lambda_{iju}^k\right)^{\beta_k}+\beta_k\left(\lambda_{iju}^k\right)^{-\alpha_k}}{\alpha_k+\beta_k}\right) \tag{29}$$

$$\frac{\partial d_{ij}^k}{\partial\beta_k} = \frac{1}{\alpha_k\beta_k^2}\sum_{u=1}^{p}\left(\frac{\beta_k\left(\lambda_{iju}^k\right)^{-\alpha_k} - \alpha_k\beta_k\left(\lambda_{iju}^k\right)^{\beta_k}\log\lambda_{iju}^k}{\alpha_k\left(\lambda_{iju}^k\right)^{\beta_k}+\beta_k\left(\lambda_{iju}^k\right)^{-\alpha_k}} - \frac{\beta_k}{\alpha_k+\beta_k} - \log\frac{\alpha_k\left(\lambda_{iju}^k\right)^{\beta_k}+\beta_k\left(\lambda_{iju}^k\right)^{-\alpha_k}}{\alpha_k+\beta_k}\right). \tag{30}$$

C. Gradient of $\mathcal{L}$ With Respect to $\boldsymbol{W}$

The gradient of $\mathcal{L}$ with respect to each W_k is

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_k} = \sum_{i,j\in(\mathcal{S}\cup\mathcal{D})}\left(\boldsymbol{X}_i^\top \boldsymbol{W}_k \frac{\partial \mathcal{L}}{\partial \boldsymbol{X}_i^k} + \boldsymbol{X}_i \boldsymbol{W}_k \frac{\partial \mathcal{L}}{\partial \boldsymbol{X}_i^k}^\top + \boldsymbol{X}_j^\top \boldsymbol{W}_k \frac{\partial \mathcal{L}}{\partial \boldsymbol{X}_j^k} + \boldsymbol{X}_j \boldsymbol{W}_k \frac{\partial \mathcal{L}}{\partial \boldsymbol{X}_j^k}^\top\right) \tag{31}$$

where $(\partial\mathcal{L}/\partial\boldsymbol{X}_i^k)$ is the gradient of $\mathcal{L}$ with respect to the low-dimensional SPD matrix $\boldsymbol{X}_i^k$, and $(\partial\mathcal{L}/\partial\boldsymbol{X}_j^k)$ is the gradient of $\mathcal{L}$ with respect to the low-dimensional SPD matrix $\boldsymbol{X}_j^k$. The eigenvalue decomposition of $\boldsymbol{X}_i^k(\boldsymbol{X}_j^k)^{-1}$ is $\boldsymbol{X}_i^k(\boldsymbol{X}_j^k)^{-1} = \boldsymbol{U}_{ij}^k\boldsymbol{\Sigma}_{ij}^k(\boldsymbol{U}_{ij}^k)^\top$. $\boldsymbol{\Sigma}_{ij}^k$ is the eigenvalue diagonal matrix, and λ_{iju}^k is the uth eigenvalue. The gradients $(\partial\mathcal{L}/\partial\boldsymbol{X}_i^k)$ and $(\partial\mathcal{L}/\partial\boldsymbol{X}_j^k)$ can be computed by

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{X}_i^k} = \boldsymbol{U}_{ij}^k \frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}_{ij}^k}\left(\boldsymbol{U}_{ij}^k\right)^\top\left(\boldsymbol{X}_i^k\right)^{-\top} \tag{32}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{X}_j^k} = (-1)\cdot\left(\boldsymbol{X}_j^k\right)^{-\top}\left(\boldsymbol{X}_i^k\right)^\top \boldsymbol{U}_{ij}^k \frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}_{ij}^k}\left(\boldsymbol{U}_{ij}^k\right)^\top\left(\boldsymbol{X}_j^k\right)^{-\top} \tag{33}$$

where $(\partial\mathcal{L}/\partial\boldsymbol{\Sigma}_{ij}^k)$ is the gradient of $\mathcal{L}$ with respect to $\boldsymbol{\Sigma}_{ij}^k$. $(\partial\mathcal{L}/\partial\boldsymbol{\Sigma}_{ij}^k)$ is a diagonal matrix, and the uth element is

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \lambda_{iju}^k} &= \frac{\partial \mathcal{L}}{\partial d_{ij}^k}\cdot\frac{\partial d_{ij}^k}{\partial \lambda_{iju}^k}\\ &= \frac{\partial \mathcal{L}}{\partial d_{ij}^k}\cdot\frac{1}{\alpha_k\beta_k}\frac{\alpha_k\beta_k\left(\lambda_{iju}^k\right)^{\beta_k-1} - \alpha_k\beta_k\left(\lambda_{iju}^k\right)^{-\alpha_k-1}}{\alpha_k\left(\lambda_{iju}^k\right)^{\beta_k} + \beta_k\left(\lambda_{iju}^k\right)^{-\alpha_k}}.\end{aligned} \tag{34}$$

Here, we also provide the gradient of $\mathcal{L}$ with respect to original SPD representations $\boldsymbol{X}_i$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{X}_i} = \sum_{k=1}^{m}\boldsymbol{W}_k^\top\frac{\partial \mathcal{L}}{\partial \boldsymbol{X}_i^k}\boldsymbol{W}_k. \tag{35}$$

With the developments above, the parameters of the model can be updated as follows. At time $t+1$, $\boldsymbol{A}$ is optimized directly by SGD: $\boldsymbol{A}_{t+1} = \boldsymbol{A}_t - \eta(\partial\mathcal{L}/\partial\boldsymbol{A}_t)$. $\boldsymbol{W}$ and $\boldsymbol{M}$ are updated by the Riemannian optimization algorithm [1]. First, the Euclidean gradients $(\partial\mathcal{L}/\partial\boldsymbol{W})$ and $(\partial\mathcal{L}/\partial\boldsymbol{M})$ are converted to the Riemannian gradients $(\partial\mathcal{L}/\partial\boldsymbol{W}^R)$ and $(\partial\mathcal{L}/\partial\boldsymbol{M}^R)$, which are on the corresponding tangent planes. Then, we obtain a new tangent vector from the Riemannian gradient. Finally, the new point on the tangent plane is mapped back to the Riemannian manifold as the updated manifold point. The computational details are given by

$$\begin{cases}\dfrac{\partial \mathcal{L}}{\partial \boldsymbol{W}_t^R} = \dfrac{\partial \mathcal{L}}{\partial \boldsymbol{W}_t} - \boldsymbol{W}_t\dfrac{1}{2}\left(\boldsymbol{W}^\top\dfrac{\partial \mathcal{L}}{\partial \boldsymbol{W}_t} + \dfrac{\partial \mathcal{L}}{\partial \boldsymbol{W}_t}^\top\boldsymbol{W}_t\right)\\ \boldsymbol{W}_{t+1} = \mathrm{q}\left(\boldsymbol{W}_t - \eta\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_t^R}\right)\end{cases} \tag{36}$$

and

$$\begin{cases}\dfrac{\partial \mathcal{L}}{\partial \boldsymbol{M}_t^R} = \boldsymbol{M}_t\frac{1}{2}\left(\dfrac{\partial \mathcal{L}}{\partial \boldsymbol{M}_t} + \dfrac{\partial \mathcal{L}}{\partial \boldsymbol{M}_t}^\top\right)\boldsymbol{M}_t\\ \boldsymbol{M}_{t+1} = \boldsymbol{M}_t^{\frac{1}{2}}\operatorname{expm}\left(-\eta\boldsymbol{M}_t^{-\frac{1}{2}}\dfrac{\partial \mathcal{L}}{\partial \boldsymbol{M}_t^R}\boldsymbol{M}_t^{-\frac{1}{2}}\right)\boldsymbol{M}_t^{\frac{1}{2}}\end{cases} \tag{37}$$

where $(\partial\mathcal{L}/\partial\boldsymbol{W}_t^R)$ and $(\partial\mathcal{L}/\partial\boldsymbol{M}_t^R)$ are the Riemannian gradients with respect to $\boldsymbol{W}_t$ and $\boldsymbol{M}_t$ at time t, and $\boldsymbol{W}_{t+1}$ and $\boldsymbol{M}_{t+1}$ are the manifold points after an update. In (36), $\mathrm{q}(\cdot)$ is the retraction operation mapping the data back to the Stiefel manifold. $\mathrm{q}(\boldsymbol{W})$ denotes the $\boldsymbol{Q}$ part of the QR decomposition of a matrix $\boldsymbol{W}$, i.e., for the matrix $\boldsymbol{W}\in\mathbb{R}^{n\times p}$, $\boldsymbol{W}$ can be decomposed as $\boldsymbol{W} = \boldsymbol{QR}$, where $\boldsymbol{Q}\in\mathbb{R}^{n\times p}$ is an orthogonal matrix and $\boldsymbol{R}\in\mathbb{R}^{p\times p}$ is an upper triangular matrix. In (37), $\operatorname{expm}(\cdot)$ is the matrix exponential function as the retraction operation. The retraction operation can guarantee the new point not only along the descent direction but also on the manifold [4]. The convergence of such SGD on the Stiefel manifold and SPD manifold was proven in [41]. Due to the page limit, the derivations of the Riemannian gradients in (36) and (37) can be found in the seminal book by Absil *et al.* [1].

VI. Computational Complexity

In this section, we comment on the computational complexity of our method. We recall that n is the size of the original SPD matrices, and m and p showing the number of low-dimensional manifolds and their respective dimensionalities. The number of training samples is denoted by N.

A. Computational Complexity of the Objective Function

The computational complexity of the point-to-set transformation in (11) is $\mathcal{O}(mn^2p + mnp^2)$. The computational complexity of inverting a matrix of size $p\times p$ is $\mathcal{O}(p^{2.376})$ via the Coppersmith–Winograd algorithm. Thus, the set-to-set distance in (15) requires $\mathcal{O}(3mp^3 + mp^{2.376} + m^2 + m)$ flops. The objective function in (10) has a computational complexity of $\mathcal{O}(N^2 + (8/3)m^3)$.

B. Computational Complexity of the Gradient

Computation of the gradient with respect to $\boldsymbol{M}$ in (23) requires $\mathcal{O}(N^2\ m + N^2\ m^2 + 2m^{2.376})$ flops. Computation of gradients with respect to all α and β values in (26) and (27) has a complexity of $\mathcal{O}(N^2\ m^2 + 2\ N^2\ m^3 + N^2m^2p)$ and $\mathcal{O}(N^2\ m^2\ p)$, respectively. Finally, the computational complexity of the gradient with respect to $\boldsymbol{W}$ in (31) requires $\mathcal{O}(m^2 + 8mp^3 + 2Nmn^2p + 2Nmnp^2)$ flops.

C. Computational Complexity of the Riemannian Optimization

The computational complexity of updating $\boldsymbol{M}$ in (37) takes $\mathcal{O}(3m^3)$, and the computational complexity of updating $\boldsymbol{W}$ in (36) takes $\mathcal{O}(4nm^2p^2 + 11m^3p^3)$ [37].

VII. Experiments

To evaluate our method, we conduct experiments on six data sets: the ETH-80 data set [42], the MSR-Action3D data set [43], the YouTube Celebrities (YTC) data set [44], the UIUC data set [45], the Describable Textures Data Set (DTD) [46], and the FGVC-aircraft data set [47].

A. Data Sets and Settings

The ETH-80 data set is an image data set, which contains 80 image sets of eight objects. In our experiment, all images of the ETH-80 data set are resized to 20×20 and encoded by intensity features. The YTC data set is a video-based face data set containing 1910 videos of 47 persons. Face regions are detected from each frame by a cascaded face detector and resized to 20×20, followed by the histogram equalized operation, and represented by gray values. The MSR-Action3D data set is a 3-D action data set containing 20 actions. In the experiments, each frame is represented by a 120-D feature vector, which is the 3-D coordinate differences of 20 skeleton joints between this frame and its two neighborhood frames. The UIUC data set contains texture images of 18 categories. We resize each image to 400×400. Then, 128-D dense scale-invariant feature transform (SIFT) features are extracted from each image with strides of 4-pixels and concatenated by 27-D RGB color features from 3×3 patches centered at the SIFT features. The DTD data set is a material data set with 47 categories (120 images per category). The FGVC-aircraft data set is a fine-grained image data set. It contains 10 000 aircraft images of 100 categories.

On the ETH-80, MSR-Action3D, and UIUC data sets, we randomly select half of the samples from each category for training and the rest are used for testing. On the YTC data set, for each person, three videos are randomly selected as the gallery, and six as the probe. For the DTD and FGVC-aircraft data set, we use 3760 and 6667 images for training, respectively, and the remaining images are considered for testing.

On the ETH-80, YTC, UIUC, DTD, and FGVC-aircraft data sets, we compute a covariance matrix $\boldsymbol{C}$ to represent each sample and add a small ridge $\delta\mathbf{I}$ to avoid singularity, where $\delta = 0.001 \times \mathrm{Tr}(\boldsymbol{C})$. On the MSR-Action3D data set, we first compute the covariance matrix $\boldsymbol{C}$ with the size of 120×120, then transform it to a 121×121 Gaussian distribution as the SPD representation, i.e., $\boldsymbol{C} = |\boldsymbol{C}|^{-(1/121)}[\boldsymbol{C} + \boldsymbol{m}\boldsymbol{m}^\top \ \boldsymbol{m}\boldsymbol{m}^\top \ 1]$, where $\boldsymbol{m}$ is the mean vector of 120-D features. Deep features used in our experiments are extracted from the "conv5-3" layer of a fine-tuned VGG-16 model and are aggregated into a 512×512 SPD representation following the work of Yu and Salzmann [48].

In all the experiment, we set $\xi = 0.01$ and $\boldsymbol{M}_0 = \mathbf{I}_m$. For the ETH-80 and YTC data sets, we set ζ_s as 5 and ζ_d as 100. For the MSR-Action3D, UIUC, DTD, and aircraft data sets, we set ζ_s as 10 and ζ_d as 100. In each iteration, we randomly sample 64 pairs into a minibatch. For the ETH-80 data set, the learning rate η is set as 10^{-4}. For the MSR-Action3D, YTC, and UIUC data sets, the learning rate η is set as 10^{-5}. For the DTD and aircraft data sets, the learning rate η is set as 10^{-6}.

B. Evaluation With Hand-Crafted Features

We first compare and contrast our PSSSD against the following baselines: AIM [17], Stein divergence [19], LEM [18], Jeffery KL divergence [33], Burg Matrix divergence [20], SPD-dimensionality reduction (SPD-DR) including AIM-DR and Stein-DR [27], covariance discriminative learning (CDL) [10], Riemannian sparse representation (RSR) [13], LEML [24], and α-CML [25]. In our method, we exploit a nearest neighborhood (NN) classifier to label a query sample.

TABLE II
Classification Accuracy (%) on Four Visual Recognition Tasks. Best Results Are Highlighted in Bold

Method	Eth-80	YTC	MSR-Action3D	UIUC
AIM [17]	85.0	65.25	84.7	35.6
Stein [19]	-	58.16	83.5	35.8
LEM [18]	93.0	63.12	84.7	36.7
Jeffery KL [33]	93.3	61.0	86.2	28.4
Burg Matrix [20]	47.3	59.6	87.0	26.5
AIM-DR [27]	96.0	68.79	93.1	58.3
Stein-DR [27]	-	65.96	94.6	58.1
CDL [10]	94.5	68.09	95.4	54.9
RSR [13]	94.8	68.44	95.0	-
LEML [24]	95.5	67.73	92.3	53.9
α-CML [25]	-	-	92.7	-
PSSSD	**97.5**	**69.50**	**95.8**	**65.8**

Table II reports the accuracy of the proposed method and the baselines on four visual recognition tasks. Some observations can be made here. For example, while the Jeffery KL divergence and the Burg Matrix divergence have better performances than AIM, Stein, and LEM on the MSR-Action3D data set, they fall behind on the UIUC data set. Our method, in contrast, is able to learn the suitable distance for the task in hand by exploiting the family of alpha–beta divergences.

Compared with SPD-DR [27], PSSSD achieves better performances on all studied data sets, demonstrating the power of encoding multiple discriminative manifolds instead of one as done in SPD-DR. LEML [24] and α-CML [25] learn a metric in the tangent space of the manifold. Though featuring fast computations, working purely in a tangent plane may lead to inaccurate modeling, as can be observed by the performance of LEML and α-CML on the MSR-Action3D and UIUC data sets.

For the MSR-Action3D data set, nonlinear kernel methods such as CDL [10] and RSR [13] achieve 95.4% and 95.0%, respectively, outperforming other baselines such as [24], [25], and [27] . Interestingly, our proposed method, without relying on nonlinear embeddings, obtains a better performance than CDL and RSR on the MSR-Action3D data set.

C. Evaluation With Deep Features

We compare PSSSD with state-of-the-art deep methods including VGG-16 [49], B-CNN [50], matrix power normalization (MPN) [51], DeepO2P [52], compact bilinear pooling (CBP) [53], low-rank bilinear pooling (LRBP) [54], statistically motivated second-order pooling (SMSO) [48], Lin *et al.* [55], and robust estimation of approximate infinite dimensional Gaussian (RAID) [56] on DTD and aircraft data set in Table III. In our method, we project the SPD representation to 32 low-dimensional manifolds (low-dimensional SPD matrices are of size 4×4).

Our method outperforms all the baselines on the DTD data set. For example, the SMSO method [48], which directly projects an SPD matrix to a vector representation, yields an

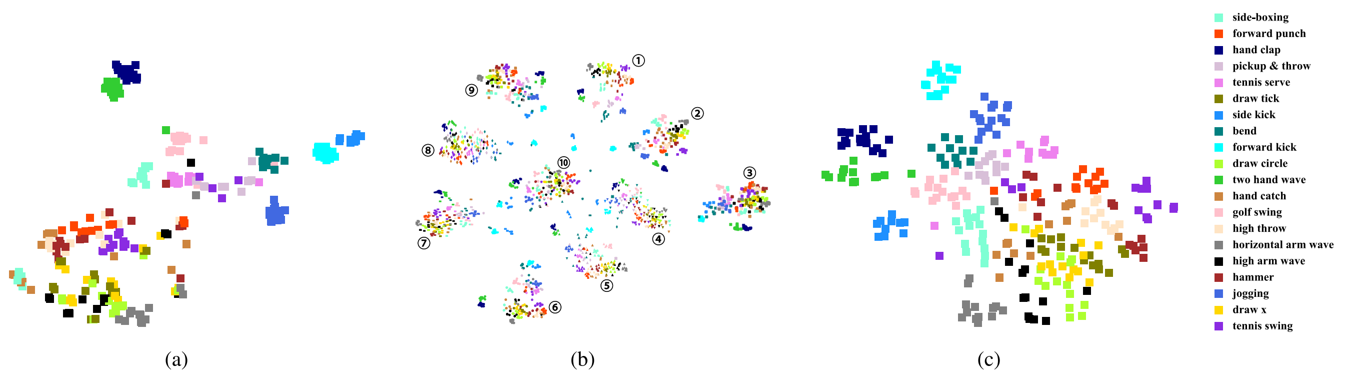


Fig. 2. Visualization of the sample distribution of the MSR-Action3D data set (colors correspond to categories). (a) Sample distribution before our metric learning. (b) Low-dimensional SPD matrix distribution, where shapes indicate different low-dimensional SPD manifolds. (c) Sample distribution after our metric learning. Best viewed in color.

TABLE III

ACCURACIES (%) ON THE DTD AND AIRCRAFT DATA SETS. BEST RESULTS ARE HIGHLIGHTED IN BOLD

Method	DTD	Aircraft
VGG-16 [49]	60.1	72.4
B-CNN [50]	67.5	84.1
MPN [51]	68.0	-
DeepO2P [52]	66.1	-
CBP-TS [53]	67.7	87.2
CBP-RM [53]	63.2	87.1
LRBP [54]	65.8	**87.3**
SMSO [48]	69.3	-
Lin *et al.* [55]	72.2	86.7
RAID [56]	76.4	-
PSSSD + deep features	**78.7**	86.1

accuracy of 69.3%, 9.4% shy of PSSSD. As another example, the work of Lin *et al.* [55], which aggregates convolutional features into a kernel matrix, is outperformed by 6.5% by PSSSD. On the aircraft data set, our method outperforms the B-CNN method [50], which directly utilizes a fully connected layer to classify SPD representations. On the downside, PSSSD performs below state-of-the-art methods such as LRBP on the aircraft data set. However, we note that we did not fine-tune the VGG-16 and only trained our similarity-based model. We believe that better results can be attained by fine-tuning of the deep model.

D. Visualization

In this section, we select the MSR-Action3D data set for the sample distribution visualization experiment, as shown in Fig. 2 (best viewed on high-resolution display). There are a total of 20 categories, and each category is represented by one type of color in the figure. Inspired by the visualization experience, combining more than one dimensionality reduction method leads to a better visualization result. First, we utilize MDS [57] to reduce SPD matrices to 200-D vectors, and then principal component analysis (PCA) [58] is applied to reduce the dimensionality to 40. Finally, the representations are reduced to 2-D vectors via t-distributed stochastic neighbor embedding (t-SNE) [59].

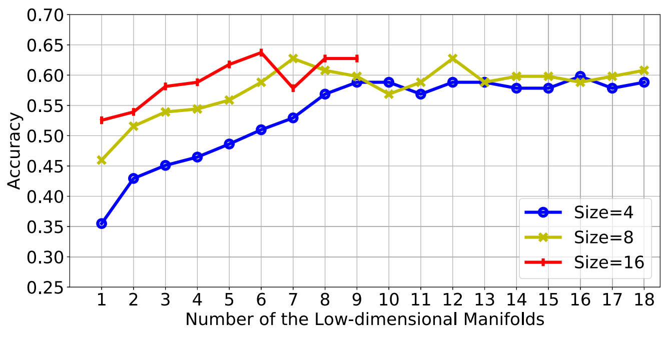
The original sample distribution is shown in Fig. 2(a). We find that only a few categories have desirable distributions, i.e., small intracategory distances and large intercategory distances, such as the "hand clap," "two hand wave," "side kick," "forward kick," and "jogging" actions. Most categories have poor distributions, such as "draw tick," "draw circle, " and "draw x." The three actions are almost-uniform mixing in the 2-D visualization figure.

The sample distribution after our metric learning is shown in Fig. 2(c). Compared with Fig. 2(a), the better locality is shown. Samples from the same category are more likely to fall into the same local region. Most actions can be separated in the 2-D visualization. Even "draw tick," "draw circle," and "draw x" also show the locality to a certain extent. By comparing Fig. 2(c) and (a), we find that after our metric learning, all samples fill the whole 2-D figure instead of locating at only a partial region of the figure.

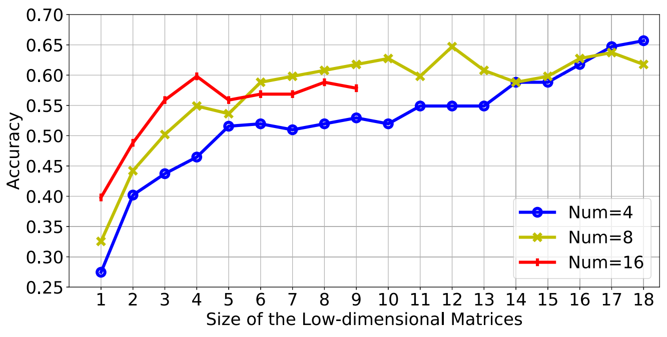
Moreover, we visualize the low-dimensional SPD matrix distribution in Fig. 2(b). In the figure, low-dimensional SPD matrices on different low-dimensional SPD manifolds are represented by different shapes, and low-dimensional SPD matrices projected from the same category are represented by the same color. Low-dimensional SPD matrices projected from the same category are more likely to be neighbors, demonstrating that the low-dimensionality manifold projection is helpful for seeking a more discriminative manifold space. We also find that low-dimensional manifolds have little relevance. Low-dimensional SPD matrices on the same low-dimensional manifold are close; otherwise, they are far apart. In addition, the low-dimensional manifolds have complementarity among them. For example, "draw tick," "draw circle," and "draw x" are three difficult actions to recognize. In some low-dimensional manifolds, they cannot be separated, while in the NO.1 low-dimensional manifold, they can be separated.

E. Ablation

To evaluate the effectiveness of our set-to-set distance and demonstrate our motivation, we conduct nine ablation experiments on the MSR-Action3D and UIUC data sets. We first constrain all (α, β) of alpha–beta divergences to $(0, 0)$, $(1, 0)$, $(1, 1)$, and $(0.5, 0.5)$. The four (α, β) parameters correspond to AIM, the Stein divergence, the Jeffrey KL divergence, and the



(a)



(b)

Fig. 3. (a) Performance of different numbers of low-dimensional manifolds with a fixed matrix size. (b) Performance of different low-dimensional manifold sizes with a fixed manifold number.

TABLE IV

ABLATION EXPERIMENTS ON MSR-ACTION3D AND UIUC DATA SETS. BEST RESULTS ARE HIGHLIGHTED IN BOLD

Method	MSR-Action3D	UIUC
The same subdistances, $(\alpha, \beta) = (0, 0)$	92.3	56.9
The same subdistances, $(\alpha, \beta) = (1, 0)$	89.7	57.8
The same subdistances, $(\alpha, \beta) = (1, 1)$	92.0	55.9
The same subdistances, $(\alpha, \beta) = (0.5, 0.5)$	92.3	56.9
The same subdistances, learnable (α, β)	94.3	61.8
Identity matrix $\boldsymbol{M}$	92.3	49.1
Diagonal matrix $\boldsymbol{M}$	92.7	55.9
w/o $\boldsymbol{W}_k^\top \boldsymbol{W}_l = \mathbf{0}$	94.6	60.8
The shortest distance	88.1	58.9
PSSSD	**95.8**	**65.8**

TABLE V

TRAINING TIME (SECONDS) ON THE MSR-ACTION3D AND UIUC DATA SETS

Method	**MSR-Action3D**		**UIUC**	
	Train	Test	Train	Test
AIM [17]	-	1372.3	-	575.2
Stein [19]	-	45.1	-	-
LEM [18]	-	15.1	-	9.4
Jeffery KL [33]	-	95.2	-	19.7
Burg Matrix [20]	-	111.5	-	21.6
AIM-DR [27]	3566.1	238.3	704.1	156.5
Stein-DR [27]	204.9	35.7	50.8	6.9
CDL [10]	1.9	0.073	7.2	0.03
RSR [13]	44.9	0.10	-	-
LEML [24]	35.0	11.1	53.0	3.1
PSSSD	420.9	92.4	583.8	18.3

Burg matrix divergence, respectively, as shown in Table I. We also relax the constraint that all subdistance measures learn a common (α, β) parameter. Then, we conduct experiments to compare different subdistance integration approaches. We successively constrain $\boldsymbol{M}$ as an identity matrix and a diagonal matrix. Through (15), the identity matrix $\boldsymbol{M}$ represents that the visual information contributions are treated equally for the SPD distance. The diagonal matrix $\boldsymbol{M}$ represents that the visual information contributions are different and can be learned, while the subdistance relationships have no contribution to the SPD distance. We also evaluate the influence of the relevance of visual information, i.e., the orthogonality of multiple projection matrices $\boldsymbol{W}_k$ in (11). After eliminating the constraints among $\boldsymbol{W}_k$, the experimental configuration is represented as "w/o $\boldsymbol{W}_k^\top \boldsymbol{W}_l = \mathbf{0}$". Finally, we try another set-to-set distance, which is designed as the shortest distance between low-dimensional SPD matrices. The ablation experiment results are shown in Table IV.

In Table IV, the poor performance of fixing $\boldsymbol{A}$ shows that visual information has different distributions and assigning individual subdistances is necessary. In the action recognition task, the Jeffrey KL divergence performs poorly and only achieves 89.7%, 6.1% lower than using the proposed set-to-set distance. The other three distance measures perform better but are still 3% lower than PSSSD. If subdistances utilize the same learnable (α, β) value, the distance measure can achieve 94.3%, an obvious improvement over the using a nonlearnable subdistance measure, which demonstrates that for a specific task, using a fixed SPD matrix distance measure may result in nonoptimal performance. If subdistance measures learn their own (α, β) values, the result is 95.8%. For the texture

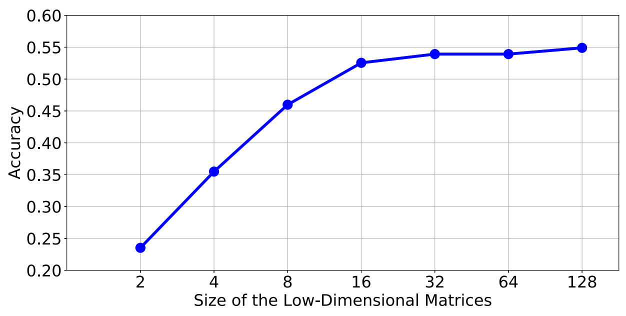
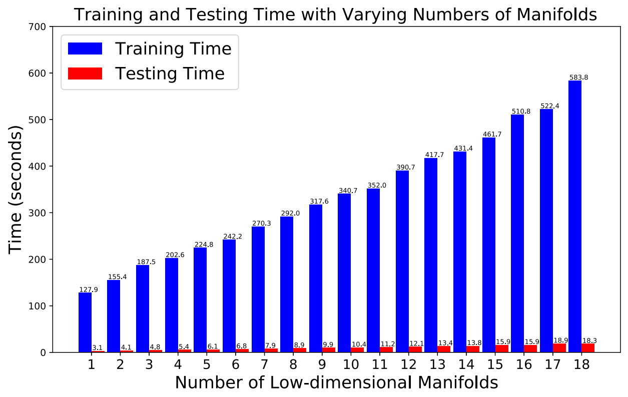


Fig. 4. Performance of only one low-dimensional manifold with varying matrix sizes.

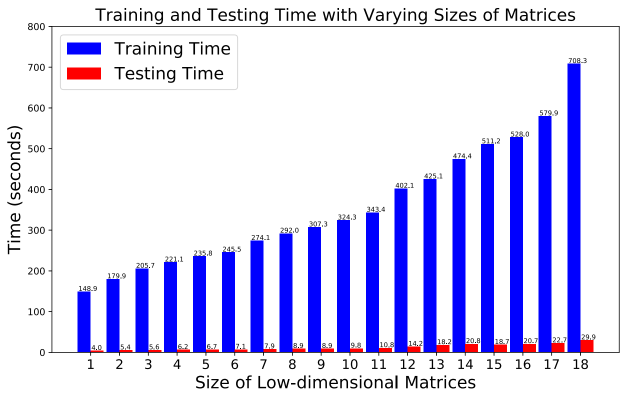
classification task, the performance of the first five ablation experiments is similar to that of the action recognition task.

Fixing $\boldsymbol{M}$ also leads to inferior performance. When $\boldsymbol{M}$ is fixed as a diagonal matrix, it performs better than $\boldsymbol{M}$ fixed as an identity matrix, which shows that visual information has different contributions. We also find that without the relationships of visual information, it achieves 92.7% and 55.9% on the two tasks, 3.1% and 9.9% lower, respectively, than our set-to-set distance. Through these experimental results, we argue that even low-dimensional SPD matrices are unrelated, and the relationships of subdistances still contribute to the SPD distance. Through the ablation experiments on $\boldsymbol{A}$ and $\boldsymbol{M}$, we demonstrate that the two parameters have different effects on our set-to-set distance, and both of them are needed.

In the "w/o $\boldsymbol{W}_k^\top \boldsymbol{W}_l$" experiment, the performance on the two tasks is 94.6% and 60.8%, worse than PSSSD, showing that the visual information relevance also influences



(a)



(b)

Fig. 5. (a) Time consumption of different numbers of low-dimensional manifolds while the matrix size is fixed as 8. (b) Time consumption of different low-dimensional manifold sizes while the manifold number is fixed as 8.

TABLE VI
TRAINING TIME (SECONDS) ON THE DTD AND AIRCRAFT DATA SETS

Method	DTD	Aircraft
VGG-16	1168	2039
PSSSD + deep features	834.2	3309.2

the distance. The performance of "the shortest distance" is not good enough. As multiple types of visual information are individual and one of them cannot be expressed by the others, only considering the shortest subdistance is insufficient, and other visual information is also helpful to the distance.

F. Efficiency

We compare the training and testing time of our method with other baselines on the MSR-Action3D and UIUC data sets. The performance is measured on using an Intel i5-8250 1.80-GHz CPU (see Table V). Note that we report the time to run 2000 iterations of our method here.

Among AIM [17], LEM [18], Stein [19], Jeffery KL [33], and Burg Matrix [20] divergences, AIM is slower than PSSSD, Jeffery KL and Burg Matrix divergences achieve comparable time with PSSSD, but our accuracy is much better. Our method is faster compared with AIM-DR [27] which utilizes the manifold projection and preserves the manifold structure as well. In comparison to CDL [10], RSR [13], LEML [24], and Stein-DR [27], PSSSD is slower. We note that CDL and RSR are kernel-based methods, and LEML learns the metric on a Euclidean space. Furthermore, the three methods do not involve the Riemannian optimization. Stein-DR does not need to implement the eigenvalue decomposition and matrix inversion operations. That said, the higher accuracy of our method justifies the extra time needed to learn the projections and the metric.

Comparisons, in terms of training time, between our method and a VGG-16 model on the DTD and aircraft data sets are shown in Table VI. Interestingly, the time required to train PSSSD is not very off (and sometimes even shorter) than the training time of the VGG-16 model, while PSSSD brings 18.6% and 13.7% improvement in accuracy, as shown in Table III.

G. Dimensionality and Number of Submanifolds

In this section, we investigate the number m and the size p of low-dimensional SPD matrices from the accuracy and time consumption on the UIUC data set.

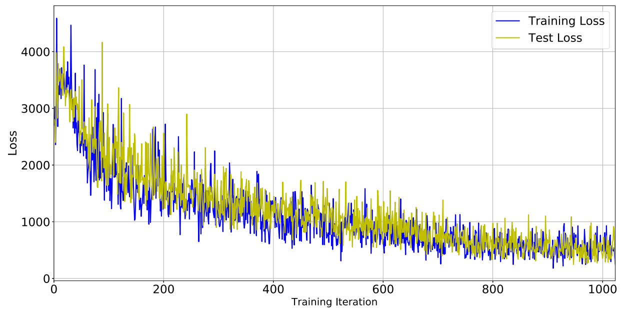
1) Accuracy Analysis: First, we fix the size of the low-dimensional SPD matrices as 4, 8, and 16 and evaluate different low-dimensional SPD manifold numbers from 1 to 18, where "Size=4" represents that the size of the low-dimensional SPD matrices is 4×4. Due to the projection matrix column full-rank constraint, when "Size=16," the manifold number must be smaller than 9. The results are shown in Fig. 3(a). We divide the three lines into two stages, i.e., the line rise and line stability. In the beginning, the accuracies continue to improve with the increase in the manifold number, then the lines tend to stabilize, and the accuracies stabilize around 60%. As the discriminative visual information is too small in the beginning stage, it is difficult to correctly classify all samples. In the stable stage, the discriminative visual information is sufficient, and the visual information redundancy occurs. We find that even when there are few low-dimensional matrices, PSSSD can still achieve state-of-the-art performance, which also shows that the original SPD representation has information redundancy.

Then, we test the performance of different low-dimensional SPD matrix sizes. We fix the low-dimensional manifold number as 4, 8, and 16, and the size of the low-dimensional matrices is changed from 1 to 18. The results are shown in Fig. 3(b). The three lines in this figure have the same trend as the evaluation of different low-dimensional SPD manifold numbers shown in Fig. 3(a). In the line rise stage, with the fixed matrix size, the larger the number of low-dimensional manifolds is, the better this experimental configuration performs. In the line stability stage, fewer manifolds perform better, which contains less redundancy. When the number of low-dimensional manifolds is fixed as 4 and the matrix size is 18, we achieve the best performance, which is over 65%. When the number of manifolds is fixed as 8, the best performance is achieved when the matrix size is 12.

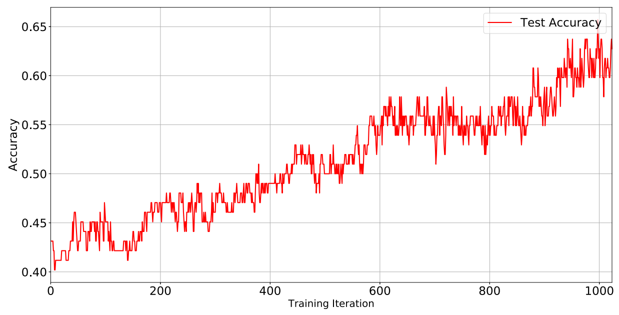
We conduct experiments to evaluate the performance when the set-to-set distance measure reduces to an alpha–beta divergence, i.e., there is only one low-dimensional manifold,

TABLE VII
ACCURACY(%) FOR EACH LOW-DIMENSIONAL MANIFOLD ON THE UIUC DATA SET

Manifold Number	Total	C-1	C-2	C-3	C-4	C-5	C-6	C-7	C-8	C-9	C-10	C-11	C-12	C-13	C-14	C-15	C-16	C-17	C-18
No.1	39.2	60	50	40	0	33.3	100	16.7	33.3	16.7	16.7	33.3	20	100	33.3	80	33.3	40	16.7
No.2	43.1	100	83.3	80	33.3	16.7	100	0	33.3	33.3	0	0	60	83.3	33.3	20	50	60	16.7
No.3	40.2	60	66.7	60	16.7	16.7	100	0	50	33.3	16.7	16.7	20	100	33.3	40	33.3	60	16.7
No.4	45.1	80	33.3	80	50	0	80	16.7	50	16.7	16.7	16.7	20	83.3	50	80	66.7	80	16.7
No.5	32.4	60	16.7	60	16.7	16.7	100	66.7	16.7	16.7	0	16.7	20	50	66.7	20	16.7	20	16.7
No.6	41.2	80	66.7	80	33.3	50	80	0	16.7	16.7	0	16.7	20	83.3	16.7	60	50	80	16.7
No.7	48	100	83.3	60	33.3	33.3	80	50	66.7	16.7	0	0	40	66.7	50	60	33.3	80	33.3
No.8	44	80	50	60	33.3	16.7	100	16.7	33.3	16.7	0	33.3	40	100	50	80	50	40	16.7
No.9	49	60	83.3	60	50	0	100	50	50	16.7	0	50	40	100	0	80	66.7	40	50
No.10	44.1	60	66.7	80	33.3	16.7	100	0	33.3	16.7	0	16.7	40	100	50	20	83.3	60	33.3
No.11	46.1	100	83.3	60	33.3	16.7	100	16.7	66.7	33.3	33.3	33.3	40	83.3	0	20	50	60	16.7
No.12	44.1	60	66.7	80	33.3	0	100	16.7	50	16.7	0	33.3	40	100	50	60	16.7	60	33.3
No.13	38.2	80	50	60	33.3	0	80	16.7	33.3	16.7	16.7	33.3	40	100	16.7	60	50	20	0
No.14	42.2	60	66.7	60	33.3	0	100	33.3	33.3	33.3	0	33.3	40	100	16.7	60	50	40	16.7
No.15	40.2	60	50	80	33.3	0	100	0	66.7	33.3	16.7	16.7	40	83.3	33.3	40	50	40	0
No.16	39.2	60	50	80	33.3	16.7	100	0	33.3	16.7	0	16.7	40	100	33.3	80	33.3	40	0
No.17	38.2	80	50	80	16.7	16.7	80	16.7	16.7	33.3	16.7	0	20	83.3	16.7	20	66.7	80	16.7
No.18	48	60	66.7	80	33.3	0	100	83.3	33.3	16.7	16.7	0	40	100	50	80	33.3	60	33.3
PSSSD	60.8	100	100	80	66.7	50	100	50	83.3	16.7	16.7	50	60	83.3	33.3	20	66.7	100	33.3



(a)



(b)

Fig. 6. (a) Line of the training loss and the test loss to the training iteration. (b) Line of the test accuracy to the training iteration.

$m = 1$, and p varies from 2 to 128, as shown in Fig. 4. In the beginning, as the size of the matrix increases, the performance improves. When p is larger than 16, the performance reaches a bottleneck, which shows that the matrix obtains less extra useful information and more redundancy as the matrix size increases. When $p = 128$, the model achieves the best performance 55%, but it is still 5.8% lower than the model using 18 low-dimensional manifolds (as shown in Table II), showing that using only one manifold is insufficient. Discovering and analyzing different types of visual information via multiple manifolds is helpful for obtaining a robust SPD distance measure.

2) *Time Consumption Analysis:* The number m of low-dimensional manifolds and the size p of low-dimensional matrices are two important hyperparameters for time consumption. We conduct experiments to evaluate the time consumption with varying m and p on the UIUC data set, as shown in Fig. 5(a) and (b), respectively. In Fig. 5(a), p is fixed as 8, and in Fig. 5(b), m is fixed as 8. As m or p increases, the time consumption also increases. In Fig. 5(a), the training time consumption increases from 127.9 to 583.8 s, and the testing time consumption increases from 3.1 to 18.3 s. The reported time consumption 583.8 and 18.3 s in Table V is our method's longest time, as shown in Fig. 5(a). In Fig. 5(b), the training time consumption increases from 148.9 to 708.3 s, and the testing time consumption increases from 4.0 to 29.9 s. In our experiment, the trends of training time and testing time consumption do not strictly conform to the computational complexity analyses, as MATLAB implements parallel computing on many matrix operations, such as matrix multiplication.

H. Low-Dimensional Manifold Effect

In this section, we conduct experiments to show different effects of multiple individual low-dimensional manifolds. After training our model on the UIUC data set, where the number of low-dimensional manifolds is 18 and the size of low-dimensional matrices is 8×8, we evaluate the classification performance by each individual subdistance, as shown in Table VII. In Table VII, there are a total of 18 low-dimensional manifolds, "No.x" represents the subdistance on the xth low-dimensional manifold, and x varies from 1 to 18. "Total" means the accuracy on the whole data set. "C-y" indicates the accuracy of the yth category, and y varies from 1 to 18.

We make the following observations.

1) Among the 18 categories, categories 4, 5, 9, and 10 are to distinguish, and categories 1, 6, and 13 are easy to distinguish.
2) Low-dimensional manifolds are complementary, and they focus on different aspects of the task. For example, manifold 2 is good at category 1 and performs poorly on category 15, while the manifold 16 is the contrary.
3) Integrating different low-dimensional subdistances is useful. In most categories, the performance of integrating all subdistances is better than that of the individual manifold, such as in categories 2 and 4. PSSSD can

discover discriminative visual information in the SPD representation and considers the visual information separately for similarity-based classification.

I. Convergence Analyses

To better understand the behavior of our method, we plotted the training/testing loss and accuracy curves for the task of texture classification in Fig. 6. We set the size and the number of low-dimensional manifolds as 8×8 and 12, respectively. Updating the parameters of the model with SGD via minibatches results in fluctuations in the loss and accuracy. However and trendwise, we observe that both measures (loss/accuracy) are improving over the course of training.

VIII. Conclusion

In this article, we have proposed a novel distance measure for similarity-based classification on SPD manifolds. In our method, we observed that complementary information can be captured by the lower dimensional manifolds, yielding a robust similarity-based classifier. Thus, considering different visual information and applying multiple individual subdistance measures are necessary. Extensive experiments on four visual recognition tasks showed that our method outperformed existing state-of-the-art methods on the SPD manifold. As a future work, we will explore deep learning architectures that can accommodate our design, given the fact that our framework is fully differentiable with respect to its parameters and inputs.

References

[1] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*. NJ, USA: Princeton Univ. Press, 2009.

[2] Z. Huang, R. Wang, S. Shan, L. Van Gool, and X. Chen, “Cross euclidean-to-Riemannian metric learning with application to face recognition from video,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 2827–2840, Dec. 2018.

[3] T. Zhang, W. Zheng, Z. Cui, Y. Zong, and Y. Li, “Deep manifold-to-manifold transforming network for action recognition,” Sep. 2017, *arXiv:1705.10732*. [Online]. Available: https://arxiv.org/abs/1705.10732

[4] S. Herath, M. Harandi, and F. Porikli, “Learning an invariant Hilbert space for domain adaptation,” in *Proc. CVPR*, Jul. 2017, pp. 3845–3854.

[5] H. Wang, Q. Wang, M. Gao, P. Li, and W. Zuo, “Multi-scale location-aware Kernel representation for object detection,” in *Proc. CVPR*, vol. 1, Jun. 2018, pp. 1248–1257.

[6] J. Zhang, L. Zhou, L. Wang, and W. Li, “Functional brain network classification with compact representation of SICE matrices,” *IEEE Trans. Biomed. Eng.*, vol. 62, no. 6, pp. 1623–1634, Jun. 2015.

[7] Z. Dong, S. Jia, C. Zhang, M. Pei, and Y. Wu, “Deep manifold learning of symmetric positive definite matrices with application to face recognition,” in *Proc. 31st AAAI Conf. Artif. Intell.*, Feb. 2017, pp. 4009–4015.

[8] Z. Huang and L. J. Van Gool, “A Riemannian network for SPD matrix learning,” in *Proc. 31st AAAI Conf. Artif. Intell.*, Feb. 2017, vol. 2, no. 4, pp. 2036–2042.

[9] P. Li, Q. Wang, W. Zuo, and L. Zhang, “Log-euclidean Kernels for sparse representation and dictionary learning,” in *Proc. ICCV*, Dec. 2013, pp. 1601–1608.

[10] R. Wang, H. Guo, L. S. Davis, and Q. Dai, “Covariance discriminative learning: A natural and efficient approach to image set classification,” in *Proc. CVPR*, Jun. 2012, pp. 2496–2503.

[11] R. Vemulapalli, J. K. Pillai, and R. Chellappa, “Kernel learning for extrinsic classification of manifold features,” in *Proc. CVPR*, Jun. 2013, pp. 1782–1789.

[12] J. Zhang, L. Wang, L. Zhou, and W. Li, “Learning discriminative stein kernel for SPD matrices and its applications,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 5, pp. 1020–1033, May 2015.

[13] M. T. Harandi, C. Sanderson, R. Hartley, and B. C. Lovell, “Sparse coding and dictionary learning for symmetric positive definite matrices: A Kernel approach,” in *Proc. Eur. Conf. Comput. Vis.*, 2012, pp. 216–229.

[14] M. Harandi and M. Salzmann, “Riemannian coding and dictionary learning: Kernels to the rescue,” in *Proc. CVPR*, Jun. 2015, pp. 3926–3935.

[15] A. Cherian, P. Stanitsas, M. Harandi, V. Morellas, and N. Papanikolopoulos, “Learning discriminative ab-divergences for positive definite matrices,” in *Proc. ICCV*, Oct. 2017, pp. 4270–4279.

[16] Y. Chen, E. K. Garcia, M. R. Gupta, A. Rahimi, and L. Cazzanti, “Similarity-based classification: Concepts and algorithms,” *J. Mach. Learn. Res.*, vol. 10, pp. 747–776, Mar. 2009.

[17] X. Pennec, P. Fillard, and N. Ayache, “A Riemannian framework for tensor computing,” *Int. J. Comput. Vis.*, vol. 66, no. 1, pp. 41–66, 2006.

[18] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache, “Log-Euclidean metrics for fast and simple calculus on diffusion tensors,” *Magn. Reson. Med.*, vol. 56, no. 2, pp. 411–421, 2006.

[19] S. Sra, “A new metric on the manifold of kernel matrices with application to matrix geometric means,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 144–152.

[20] B. Kulis, M. Sustik, and I. Dhillon, “Learning low-rank Kernel matrices,” in *Proc. 23rd Int. Conf. Mach. Learn.*, Jun. 2006, pp. 505–512.

[21] A. Cichocki, S. Cruces, and S.-I. Amari, “Log-determinant divergences revisited: Alpha-beta and gamma log-Det divergences,” *Entropy*, vol. 17, no. 5, pp. 2988–3034, 2015.

[22] D. Thiyam, S. Cruces, J. Olias, and A. Cichocki, “Optimization of alpha-beta log-det divergences and their application in the spatial filtering of two class motor imagery movements,” *Entropy*, vol. 19, no. 3, p. 89, Mar. 2017.

[23] R. Vemulapalli and D. W. Jacobs, “Riemannian metric learning for symmetric positive definite matrices,” Jan. 2015, *arXiv:1501.02393*. [Online]. Available: https://arxiv.org/abs/1501.02393

[24] Z. Huang, R. Wang, S. Shan, X. Li, and X. Chen, “Log-Euclidean metric learning on symmetric positive definite manifold with application to image set classification,” in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 720–729.

[25] L. Zhou, L. Wang, J. Zhang, Y. Shi, and Y. Gao, “Revisiting metric learning for SPD matrix based visual representation,” in *Proc. CVPR*, Jul. 2017, pp. 7111–7119.

[26] M. H. Quang, M. San Biagio, and V. Murino, “Log-Hilbert-Schmidt metric between positive definite operators on Hilbert spaces,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 388–396.

[27] M. Harandi, M. Salzmann, and R. Hartley, “Dimensionality reduction on SPD manifolds: The emergence of geometry-aware methods,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 1, pp. 48–62, Jan. 2018.

[28] H.-J. Ye, D.-C. Zhan, Y. Jiang, and Z.-H. Zhou, “What makes objects similar: A unified multi-metric learning approach,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 5, pp. 1257–1270, May 2019.

[29] J. Lu, J. Hu, and Y.-P. Tan, “Discriminative deep metric learning for face and kinship verification,” *IEEE Trans. Image Process.*, vol. 26, no. 9, pp. 4269–4282, Sep. 2017.

[30] M. Liu, R. Wang, S. Li, S. Shan, Z. Huang, and X. Chen, “Combining multiple Kernel methods on Riemannian manifold for emotion recognition in the wild,” in *Proc. 16th Int. Conf. Multimodal Interact.*, Nov. 2014, pp. 494–501.

[31] H. Yan, “Collaborative discriminative multi-metric learning for facial expression recognition in video,” *Pattern Recognit.*, vol. 75, pp. 33–40, Mar. 2018.

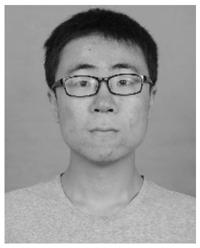
[32] Y. Zhang, H. Zhang, N. M. Nasrabadi, and T. S. Huang, “Multi-metric learning for multi-sensor fusion based classification,” *Inf. Fusion*, vol. 14, no. 4, pp. 431–440, Oct. 2013.

[33] M. Moakher and P. G. Batchelor, “Symmetric positive-definite matrices: From geometry to applications and visualization,” in *Visualization and Processing of Tensor Fields*, vol. 17, J. Weickert and H. Hagen, Eds. Berlin, Germany: Springer, 2006, pp. 285–298.

[34] A. Cichocki and S.-I. Amari, “Families of alpha- beta- and gamma-divergences: Flexible and robust measures of similarities,” *Entropy*, vol. 12, no. 6, pp. 1532–1568, 2010.

[35] X. Chen, J. Weng, W. Lu, J. Xu, and J. Weng, “Deep manifold learning combined with convolutional neural networks for action recognition,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 9, pp. 3938–3952, Sep. 2018.

[36] J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon, "Information-theoretic metric learning," in *Proc. 24th Int. Conf. Mach. Learn.*, Jun. 2007, pp. 209–216.
[37] M. Harandi, M. Salzmann, and R. Hartley, "Joint dimensionality reduction and metric learning: A geometric take," in *Proc. 34th Int. Conf. Mach. Learn.*, Aug. 2017, pp. 1404–1413.
[38] J. Hamm and D. D. Lee, "Grassmann discriminant analysis: A unifying view on subspace-based learning," in *Proc. 25th Int. Conf. Mach. Learn.*, Jul. 2008, pp. 376–383.
[39] S. Ying, Z. Wen, J. Shi, Y. Peng, J. Peng, and H. Qiao, "Manifold preserving: An intrinsic approach for semisupervised distance metric learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 7, pp. 2731–2742, Jul. 2018.
[40] M. Faraki, M. T. Harandi, and F. Porikli, "Large-scale metric learning: A voyage from shallow to deep," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 9, pp. 4339–4346, Sep. 2018.
[41] S. Bonnabel, "Stochastic gradient descent on Riemannian manifolds," *IEEE Trans. Autom. Control*, vol. 58, no. 9, pp. 2217–2229, Sep. 2013.
[42] B. Leibe and B. Schiele, "Analyzing appearance and contour based methods for object categorization," in *Proc. CVPR*, vol. 2, Jun. 2003, p. II–409.
[43] W. Li, Z. Zhang, and Z. Liu, "Action recognition based on a bag of 3D points," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 9–14.
[44] M. Kim, S. Kumar, V. Pavlovic, and H. Rowley, "Face tracking and recognition with visual constraints in real-world videos," in *Proc. CVPR*, Jun. 2008, pp. 1–8.
[45] Z. Liao, J. Rock, Y. Wang, and D. Forsyth, "Non-parametric filtering for geometric detail extraction and material representation," in *Proc. CVPR*, Jun. 2013, pp. 963–970.
[46] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, "Describing textures in the wild," in *Proc. CVPR*, Jun. 2014, pp. 3606–3613.
[47] S. Maji, E. Rahtu, J. Kannala, M. Blaschko, and A. Vedaldi, "Fine-grained visual classification of aircraft," Jun. 2013, *arXiv:1306.5151*. [Online]. Available: https://arxiv.org/abs/1306.5151
[48] K. Yu and M. Salzmann, "Statistically-motivated Second-order Pooling," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 600–616.
[49] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," Sep. 2014, *arXiv:1409.1556*. [Online]. Available: https://arxiv.org/abs/1409.1556
[50] T.-Y. Lin, A. Roy-Chowdhury, and S. Maji, "Bilinear CNN models for fine-grained visual recognition," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1449–1457.
[51] P. Li, J. Xie, Q. Wang, and W. Zuo, "Is Second-order information helpful for large-scale visual recognition?" in *Proc. ICCV*, Oct. 2017, pp. 2089–2097.
[52] C. Ionescu, O. Vantzos, and C. Sminchisescu, "Matrix backpropagation for deep networks with structured layers," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 2965–2973.
[53] Y. Gao, O. Beijbom, N. Zhang, and T. Darrell, "Compact bilinear pooling," in *Proc. CVPR*, Jun. 2016, pp. 317–326.
[54] S. Kong and C. Fowlkes, "Low-rank bilinear pooling for fine-grained classification," in *Proc. CVPR*, Jul. 2017, pp. 7025–7034.
[55] T.-Y. Lin, S. Maji, and P. Koniusz, "Second-order democratic aggregation," in *Proc. Eur. Conf. Comput. Vis.*, Sep. 2018, pp. 620–636.
[56] Q. Wang, P. Li, W. Zuo, and L. Zhang, "RAID-G: Robust estimation of approximate infinite dimensional Gaussian with application to material recognition," in *Proc. CVPR*, Jun. 2016, pp. 4433–4441.
[57] J. B. Kruskal, "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis," *Psychometrika*, vol. 29, no. 1, pp. 1–27, Mar. 1964.
[58] I. Jolliffe, *Principal Component Analysis* (Springer Series in Statistics), 2nd ed. New York, NY, USA: Springer-Verlag, 2002.
[59] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.



Zhi Gao received the B.S. degree in computer science from the Beijing Institute of Technology (BIT), Beijing, China, in 2017, where he is currently pursuing the Ph.D. degree with the Beijing Laboratory of Intelligent Information Technology.

His current research interests include pattern recognition, machine learning, and computer vision on the Riemannian manifold.

Yuwei Wu (M'19) received the Ph.D. degree in computer science from the Beijing Institute of Technology (BIT), Beijing, China, in 2014.

He is currently an Assistant Professor with the School of Computer Science, BIT. From August 2014 to August 2016, he was a Post-Doctoral Research Fellow with the School of Electrical and Electronic Engineering (EEE), Nanyang Technological University (NTU), Singapore. His current research interests include computer vision, information retrieval, and machine learning.

Dr. Wu was a recipient of the Outstanding Ph.D. Thesis Award from BIT and the Distinguished Dissertation Award Nominee from China Association for Artificial Intelligence (CAAI).

Mehrtash Harandi (M'10) is currently a Senior Lecturer with the Department of Electrical and Computer Systems Engineering, Monash University, Melbourne, VIC, Australia. He is also a Contributing Research Scientist with the Machine Learning Research Group (MLRG), Data61/CSIRO and an Associated Investigator with the Australian Center for Robotic Vision (ACRV), Brisbane, QLD, Australia. His current research interests include theoretical and computational methods in machine learning, computer vision, signal processing, and Riemannian geometry.

Yunde Jia (M'11) received the B.S., M.S., and Ph.D. degrees from the Beijing Institute of Technology (BIT), Beijing, China, in 1983, 1986, and 2000, respectively.

He was a Visiting Scientist with the Robotics Institute, Carnegie Mellon University (CMU), Pittsburgh, PA, USA, from 1995 to 1997. He is currently a Professor with the School of Computer Science, BIT, and the Team Head of BIT, with a focus on innovation of vision and media computing. He serves as the Director for the Beijing Laboratory of Intelligent Information Technology. His current research interests include computer vision, vision-based HCI and HRI, and intelligent robotics.