Neural Networks 151 (2022) 94–110

Contents lists available at ScienceDirect

Neural Networks

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

Learning a discriminative SPD manifold neural network for image set classification

Rui Wang [a,b], Xiao-Jun Wu [a,b,*], Ziheng Chen [a,b], Tianyang Xu [a,b], Josef Kittler [a,c,1]

[a] School of Artificial Intelligence and Computer Science, Jiangnan University, Wuxi 214122, China
[b] Jiangsu Provincial Engineering Laboratory of Pattern Recognition and Computational Intelligence, Jiangnan University, Wuxi 214122, China
[c] Centre for Vision, Speech and Signal Processing (CVSSP), University of Surrey, Guildford GU2 7XH, UK

ARTICLE INFO

Article history:
Received 2 November 2021
Received in revised form 15 February 2022
Accepted 7 March 2022
Available online 16 March 2022

Keywords:
SPD manifold neural network
Image set classification
Metric learning
Riemannian barycenter
Riemannian optimization

ABSTRACT

Performing pattern analysis over the symmetric positive definite (SPD) manifold requires specific mathematical computations, characterizing the non-Euclidian property of the involved data points and learning tasks, such as the image set classification problem. Accompanied with the advanced neural networking techniques, several architectures for processing the SPD matrices have recently been studied to obtain fine-grained structured representations. However, existing approaches are challenged by the diversely changing appearance of the data points, begging the question of how to learn invariant representations for improved performance with supportive theories. Therefore, this paper designs two Riemannian operation modules for SPD manifold neural network. Specifically, a Riemannian batch regularization (RBR) layer is firstly proposed for the purpose of training a discriminative manifold-to-manifold transforming network with a novelly-designed metric learning regularization term. The second module realizes the Riemannian pooling operation with geometric computations on the Riemannian manifolds, notably the Riemannian barycenter, metric learning, and Riemannian optimization. Extensive experiments on five benchmarking datasets show the efficacy of the proposed approach.

1. Introduction

Image set classification has made continuous progress in many practical applications, such as face recognition (Cheng, Zhou, & Han, 2017; Harandi, Salzmann, & Hartley, 2018; Huang, Wang, Shan, Li and Chen, 2015; Lu, Liong, & Zhou, 2017; Rao, Lu, & Zhou, 2019), facial emotional recognition (Brooks, Schwander, Barbaresco, Schneider, & Cord, 2019; Huang & Van Gool, 2017; Wang, Wu, & Kittler, 2020), action recognition (Gao, Wu, Harandi, & Jia, 2019; Harandi, Salzmann, & Hartley, 2017; Nguyen, Brun, Lézoray, & Bougleux, 2019), and dynamic scene classification (Sun, Zhen, Zheng, Yang, Yin, & Li, 2017; Wang, Wu and Kittler, 2021). The image set, consisting of a set of image instances of the same visual content, is more flexible than the single still image in providing data variability information for regions of interest description. Therefore, distinguishing among image sets is substantially different from the conventional classification problem where the decision making is based on a single still image. Considering the natural changes of morphology, illumination, viewpoint, and other possible conditions in the data capturing process, image sets usually contain large intra-subject diversity and inter-subject ambiguity. Accordingly, how to effectively perform image set modeling and invariant pattern information extraction become a key challenge in the community of computer vision and pattern recognition.

Among the existing statistical approaches, covariance matrix has achieved remarkable advance for image set representation with the capacity to straightforwardly characterize the spatiotemporal fluctuations of data sequences (Harandi et al., 2018; Huang & Van Gool, 2017; Huang, Wang, Shan, Li et al., 2015; Wang, Guo, Davis, & Dai, 2012), which is selected to model the image sets in this paper. Since the distinct geometry spanned by a group of SPD matrices of the same dimensionality is a curved Riemannian manifold, *i.e.*, SPD manifold (Arsigny, Fillard, Pennec, & Ayache, 2007; Tang, Feng, Tino, Si, & Ji, 2021), tools from Euclidean geometry cannot be successfully used for classification. Driven by the well-studied Riemannian metrics, including Log-Euclidean Metric (LEM) (Arsigny et al., 2007) and Affine-Invariant Riemannian Metric (AIRM) (Harandi & Salzmann, 2015; Harandi, Sanderson, Hartley, & Lovell, 2012; Huang, Wang, Shan, & Chen, 2015a; Pennec, Fillard, & Ayache, 2006; Vemulapalli, Pillai, & Chellappa, 2013; Wang et al., 2012) suggest to exploit Riemannian kernel functions to embed the original SPD matrices into a Reproducing Kernel Hilbert Space (RKHS), providing a feasible

* Corresponding author at: School of Artificial Intelligence and Computer Science, Jiangnan University, Wuxi 214122, China.
E-mail address: wu_xiaojun@jiangnan.edu.cn (X.-J. Wu).

[1] Life Member, IEEE.

https://doi.org/10.1016/j.neunet.2022.03.012
0893-6080/

path for generalizing Euclidean computations to SPD manifolds. However, this approach could not yield optimal results as it distorts the geometry of the original data manifold by the process of SPD matrix transformation.

As a countermeasure, several recent studies (Gao et al., 2019; Harandi et al., 2018; Huang, Wang, Shan, Li et al., 2015; Wang, Wu, Liu and Kittler, 2021; Zhou, Wang, Zhang, Shi, & Gao, 2017) take the Riemannian geometry of the resulting space into consideration during Riemannian matrix learning. To be specific, they jointly learn an orthogonal mapping and a similarity metric for the original data manifold, such that a low-dimensional manifold with maximum discriminatory power can be generated. In parallel with the above developments, an SPD manifold neural network has recently been proposed (SPDNet Huang & Van Gool, 2017), which endows the SPD matrix with deep and nonlinear learning scheme. The network structure of SPDNet is comprised of a number of trainable blocks, each of which contains an SPD matrix bilinear mapping (convolutional-like) layer and an activation layer for feature transformation and regularization. As a consequence, the Riemannian geometry of the input data is layer-wisely maintained. Subsequently, a Riemannian–Euclidean embedding layer embeds the learned feature manifolds into an Euclidean space for classification. More architectures have followed thereafter (Brooks et al., 2019; Nguyen et al., 2019; Wang et al., 2020; Wang, Wu, Kittler, 2021; Zhang et al., 2020), designing substitutes to the elementary building blocks. However, the challenge of within-class diversity and between-class ambiguity of the representations remains unsolved, and the used single cross-entropy loss is insufficient to supervise a powerful Riemannian network embedding. In this sense, learning a discriminative lower-dimensional feature manifold for the input SPD matrices is a promising proposition, reflecting better inter-class separability and smaller intra-class diversity.

In this paper, two new layers for analyzing and processing data points on the Riemannian manifolds are designed, for the sake of improving the nonlinear representation learning capacity of the original SPD manifold neural network (SPDNet). Specifically, the first module, located before the BiMap layer, can be regarded as a Riemannian batch regularization layer (RBR). It is used to generate more appropriate initial filters for the BiMap layer in each forward propagation by encoding the intra-class variability information and the inter-class similarity information within a batch of data using a novelly-designed metric learning regularization term. Since these to-be-learned transformation matrices are endowed with label- and distribution-based supervisory information, the resulting feature manifolds will embody a large within-class compactness and between-class separability. Similarly, in each forward pass, the initial weights of the RBR layer are provided by the corresponding BiMap layer. Hence, this can be considered as a mutual promotion mechanism, and the RBR module is arbitrarily pluggable.

The second module realizes the Riemannian pooling operation through the following three steps in each forward pass. The first step is to compute the Riemannian barycenter within a batch of samples. Then, the pooling operation can be formulated as the problem of finding a projection that generates a low-dimensional manifold with a maximum variance of the data in an unsupervised scenario. Finally, the learning of the mapping is transformed into an optimization problem on the Grassmannian manifold, solved by the Grassmannian conjugate gradient method. Besides, the learned projection also needs to be optimized in each backward pass. The effectiveness of the proposed method is evaluated on three image set classification tasks: dynamic scene classification, facial emotion recognition, and action recognition, respectively. Extensive experimental results indicate that the proposed two Riemannian learning algorithms can lead to a considerable advancement in classification performance of the original SPD manifold neural network.

This article is organized as follows: Firstly, the related works are introduced in Section 2. Then, the preliminaries, the basic architecture of SPDNet, and the proposed two Riemannian operation modules are presented in Section 3. Next, Section 4 introduces the Riemannian matrix backpropagation for training the refurbished SPD manifold neural network, followed by the experimental study in Section 5. Afterward, an application of the suggested method to the skeleton-based human action recognition with unmanned aerial vehicles is shown in Section 6. Finally, this paper is summarized in Section 7.

2. Related works

Although image set-based visual classification problem has attracted great attention in the CV&PR community, the wide range of data variations caused by the changes in view appearance, illumination, and other natural conditions makes it rather a challenging task. To learn useful statistical representations from image set data, the deep learning-related feature selection methods have achieved considerable progress in image set classification. To name a few, Sun et al. (2017) first compute the deep local match kernels upon arc-cosine similarity to effectively measure the similarity between images. Since these local match kernels have different representation capacities, an anchor-based aggregation mechanism is then proposed to form a discriminative global match kernel between image sets via kernel alignment learning. Based on the advantages of deep learning and metric learning, Lu, Wang, Deng, Moulin, and Zhou (2015) first pass each image set into multiple layers of deep neural network to generate class-specific fine-grained representations. Then, the maximal manifold margin criterion is applied to increase intra-class compactness and inter-class separability of the features for better decision making. Similar to Lu et al. (2015) and Cheng et al. (2017) construct a stacked autoencoder architecture coupled with two successive metric learning stages to generate powerful representations for effective similarity measurement between image sets. However, one of the main limitations of deep learning methods in practice is the classification over small-scale datasets. Recently, Nadeem, Shah, Bennamoun, Togneri, and Sohel (0000) have proposed a linear regression model for image set classification with limited data, which does not require any training or feature extraction procedures. Specifically, they first treat each gallery set as a subspace manifold in a high-dimensional Euclidean space. Then, the least squares-based solution is exploited to estimate the regression model for each image of the test set, followed by the projection of the test images on the gallery subspaces using the learned parameters. Finally, the Euclidean residual between the original and the projected test images is used as the distance metric. To improve the discriminatory power of the learned feature representations, Zhang, Yang, Zheng, Luo, and Zhang (2021) propose to jointly learn an embedding mapping and a dictionary for the original training sets based on the Fisher criterion, such that the intra-class dispersion will be narrowed and the inter-class separability will be enlarged at the same time. Considering the linear representation learned above is unable to sufficiently parse the nonlinear structure of the data in the complicated scenarios, the authors respectively generalize the designed framework to the kernel and multi-layer versions to further improve the classification performance. As Zhang et al. (2021) does not make use of the global statistical properties of image set data during dictionary learning, Cheng, Yap, and Wen (2021) propose a joint statistical and spatial sparse representation framework to learn the coupled dictionaries for both local image patch features and the global Gaussian distribution

embedded features. Different from the aforementioned methods that perform feature extraction and classification in the Euclidean domain, Riemannian manifold learning-based image set classification methods have demonstrated superb capability in encoding and analyzing the geometrical structure of image set data. A review of Riemannian manifold learning-based methods is as follows:

2.1. Riemannian kernel-based discriminant analysis approach

This type of approach first embeds the original SPD manifold-valued data into an explicit kernel space via the well-studied Riemannian kernel functions. Then, the Euclidean computations can be applied to carry out discriminant learning and classification. Therein, Wang et al. (2012) exploit the LEM-derived Riemannian kernel function to map the data from the original SPD manifold to RKHS, followed by the Kernel Discriminant Analysis (KDA) (Baudat & Anouar, 2000) for the generation of a discriminative decision space. Since Dictionary Learning and Sparse Coding (DLSC) have made impressive success in face recognition and other computer vision problems, Harandi et al. (2012) generalize DLSC to the context of SPD manifolds by utilizing the kernel function based on the Stein divergence (Zhang, Wang, Zhou, & Li, 2015) to embed SPD matrices into RKHS. Vemulapalli et al. (2013) propose to jointly learn the kernel function and the classifier by solving a convex optimization problem using the multi-kernel learning approach, which alleviates the impact of the issue of kernel-selection on the classification performance. Harandi and Salzmann (2015) design a general kernel-based Riemannian coding framework to make DLSC can be generalized to different types of manifolds. Since the statistical correlation between query image set and training sets tends to be weak and the methods mentioned above do not explicitly model and learn such information, Yan, Sun, Sun, and Li (2020) propose a semi-supervised learning framework to tackle this issue. Specifically, the authors first construct a fully-trusted graph with the labeled training Gaussian components and the available unlabeled test Gaussian components for the purpose of obtaining the data-dependent kernels. Since the underlying relationships between Gaussian components w.r.t the original image sets are explicitly reflected in such kernels, a weighted kernel fuzzy discriminant analysis algorithm is suggested to reduce the fuzzified intra-class dispersion and magnify the fuzzified inter-class dissimilarity for improved classification. To characterize image set data from multi-geometric perspective, Huang et al. (2015a) encode each given image set simultaneously with mean, covariance matrix, and Gaussian distribution. For the obtained complementary but heterogeneous statistical features, three data-specific kernel functions are first applied to embed them into RKHSs. Then, a hybrid Euclidean-and-Riemannian metric learning framework is proposed to merge different kernel features into a unified subspace for classification. Inspired by Huang et al. (2015a) and Chen, Xu, Wu, Wang, and Kittler (2021) also learn multiple robust distance measures for different statistical distributions using multi-kernel metric learning framework. But, Chen et al. (2021) substitutes the mean vector used in Huang et al. (2015a) with linear subspace to acquire more geometric information. Besides, Chen et al. (2021) inject the multiple sparse graph embedding mechanism into the metric learning framework to generate different attentions for different local kernel regions. In this way, a more discriminative subspace can be yielded for effective similarity measurement.

2.2. Riemannian dimensionality reduction approach

However, the aforementioned kernel-based approach may lead to undesirable results as the Riemannian geometry of the original data manifold has not been fully preserved by the feature transformation process. To tackle this problem, a slice of geometry-aware discriminant analysis methods that jointly perform embedding mapping learning and similarity metric learning directly on the original Riemannian manifolds have been proposed recently. Wherein, Harandi et al. (2018) build a graph embedding scheme-guided LEM learning framework to learn an orthogonal mapping for the generation of an efficient and compact low dimensional SPD manifold. Motivated by Harandi et al. (2018) and Wang, Wu, Liu et al. (2021) design a graph embedding mechanism-guided Projection Metric (PM) (Huang, Wang, Shan, & Chen, 2015b) Learning framework for discriminative Grassmannian dimensionality reduction. In Wang, Wu, Liu et al. (2021), the authors exploit the collaborative representation mechanism to adaptively learn w_{ij} of a pair of vertices, while Harandi et al. (2018) fixes them to 1, which will lose some discriminative information. Different from Harandi et al. (2018) and Huang, Wang, Shan, Li et al. (2015) design an LEM learning framework in the domain of SPD matrix logarithms, reducing the computational burden. Considering the fact that LEML (Huang, Wang, Shan, Li et al., 2015) needs to learn a $d \times k$ projection matrix for the $d \times d$ SPD matrices, Zhou et al. (2017) propose a lightweight algorithm that learns only d parameters by adjusting the eigenvalues of SPD matrices. Motivated by the data-specific transformation learning mechanism of Zhou et al. (2017) and Gao et al. (2019) propose to characterize the multi-view geometric information of SPD matrices by jointly learning multiple point-to-set projections and set-to-set distances to probe a more robust similarity measure. However, the inherent shallow linear feature transformation mechanism makes this kind of Riemannian learning approach unable to faithfully characterize the geometrical structure of the original data manifold, thus may result in suboptimal solutions.

2.3. Riemannian deep learning approach

In the context of SPD manifold neural networks, most of the developments focus on refurbishing existing functional modules to adapt to different application scenarios of computer vision and machine learning. Specifically, Nguyen et al. (2019) generalize the paradigm of SPD manifold deep learning to the domain of Gaussian embedded Riemannian manifolds, and design two Gaussian aggregation sub-networks from both spatial and temporal domains. As a result, more informative SPD matrices w.r.t the original skeletal data can be generated for classification. Inspired by the merits of Euclidean batch normalization, Brooks et al. (2019) design a Riemannian batch normalization layer for SPDNet using parallel transport technique (Absil, Mahony, & Sepulchre, 2009) to guide the learning of a more appropriate network embedding. Considering that the end-to-end SPD neural networks require complicated Riemannian matrix backpropagation to train, Wang, Wu, Kittler (2021) design a lightweight feedforward neural network trained by the shallow optimization algorithm for SPD matrix nonlinear learning, showing competitive classification performance, especially with limited data. Based on the architecture of GrNet (Huang, Wu, & Van Gool, 2018; Wang et al., 2020) prolongate the philosophy of Wang, Wu, Kittler (2021) to the context of Grassmannian manifolds, and design a lightweight feedforward model for subspace learning. Whereafter, a graph embedding scheme-guided multi-kernel metric learning framework is proposed for the purpose of increasing the representation capacity of such network in the complex visual scenarios. Different from the aforementioned SPD architectures that utilize bilinear mapping to compress SPD matrices, Zhang et al. (2020) propose an SPD matrix 2D convolutional layer, requiring each convolutional kernel also to be SPD.

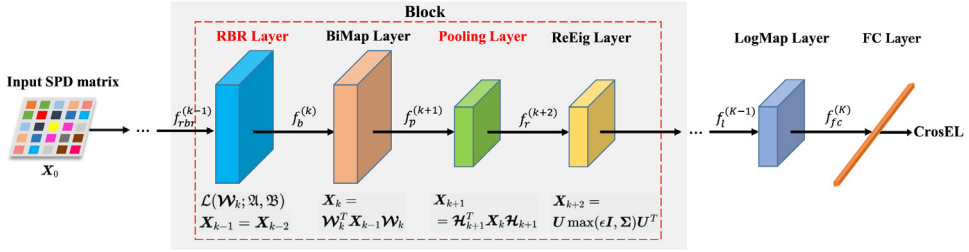


Fig. 1. Schematic diagram of the proposed SPDNetRP architecture, which is constructed on the basis of the original SPD manifold neural network (SPDNet). Here, CrosEL represents the cross-entropy loss.

Table 1
List of symbols and their corresponding explanations.

Symbols	Explanation
$\boldsymbol{v}$	Non-zero vector of d-dimensional
$\boldsymbol{X}$	A real SPD matrix of the size $d \times d$
$\mathcal{S}_{++}^d$	The space of SPD real $d \times d$ matrices
$\mathcal{T}_{\boldsymbol{X}_i}\mathcal{S}_{++}^d$	The tangent space of $\mathcal{S}_{++}^d$ at $\boldsymbol{X}_i$
K	The number of layers of the network
k	$k = 1 \rightarrow K$
$\mathcal{W}_k$	To-be-learned transformation matrix of the kth layer
$\boldsymbol{X}_k$	The resulting matrix of the kth layer
$\boldsymbol{S}_i$	The ith image set
$\boldsymbol{s}_i^r$	The rth vectorized frame of $\boldsymbol{S}_i$
$\boldsymbol{u}_i$	The mean of $\boldsymbol{S}_i$
η	Perturbation parameter
$\mathfrak{A}, \mathfrak{B}$	Two sets of similar and dissimilar computing pairs
$N_{\mathfrak{A}}, N_{\mathfrak{B}}$	The number of sample pairs contained in $\mathfrak{A}$ and $\mathfrak{B}$
$\boldsymbol{X}_k^{i,o}$	The anchor point of a mined triplet in the kth layer
$\boldsymbol{X}_k^{j,+}$	The positive point with the same label as $\boldsymbol{X}_k^{i,o}$
$\boldsymbol{X}_k^{j,-}$	The negative point with different label from $\boldsymbol{X}_k^{i,o}$
$\xi_{\mathfrak{A}}$	The intra-manifold distance margin
$\xi_{\mathfrak{B}}$	The inter-manifold distance margin
$\mathcal{D}_{lem}$	The Log-Euclidean distance
$St(d_k, d_{k-1})$	A compact Stiefel manifold
B	The batch size
m_k	The number of feature maps in the kth layer
$\boldsymbol{\mathcal{Z}}_k$	To-be-learned Riemannian mean of the kth layer
$\mathcal{H}_k$	To-be-learned embedding mapping of the kth pooling layer
g	An immersion or submersion mapping from $\mathcal{S}_{++}^d$ to $\mathcal{S}_{++}^k$
$Dg(\boldsymbol{X})$	The tangent map from $\mathcal{T}_{\boldsymbol{X}}\mathcal{S}_{++}^d$ to $\mathcal{T}_{g(\boldsymbol{X})}\mathcal{S}_{++}^k$
ρ	A continuously differentiable map from $\mathcal{T}_{\boldsymbol{I}}\mathcal{S}_{++}^d$ to $\mathcal{T}_{\boldsymbol{I}}\mathcal{S}_{++}^k$
ψ	A linear isomorphism between $\mathcal{T}_{\boldsymbol{X}}\mathcal{S}_{++}^d$ and $\mathcal{T}_{\boldsymbol{I}}\mathcal{S}_{++}^d$
ψ'	A linear isomorphism between $\mathcal{T}_{g(\boldsymbol{X})}\mathcal{S}_{++}^k$ and $\mathcal{T}_{\boldsymbol{I}}\mathcal{S}_{++}^k$
$L^{(k)}$	The loss function of the kth layer
ℓ	The cross-entropy loss of the last output layer
$\nabla L_{\mathcal{W}_k}^{(k)}$	The Euclidean gradient of $L^{(k)}$ w.r.t $\mathcal{W}_k$ in the kth layer
$\tilde{\nabla} L_{\mathcal{W}_k}^{(k)}$	The tangential component to the Stiefel manifold
Γ	The retraction operation

Table 2
List of acronyms and their corresponding full names.

Acronyms	Full names
SPD	Symmetric Positive Definite
RBR	Riemannian Batch Regularization
LEM	Log-Euclidean Metric
AIRM	Affine-Invariant Riemannian Metric
RKHS	Reproducing Kernel Hilbert Space
KDA	Kernel Discriminant Analysis
DLSC	Dictionary Learning and Sparse Coding
BiMap	Bilinear Mapping
ReEig	Eigenvalue Rectification
LogEig	Eigenvalue Logarithm
ExpEig	Eigenvalue Exponential
AFEW	Acted Facial Expression in Wild
MDSD	Modeling Dynamic Scenes Dataset
CG	Cambridge Gestures
FPHA	First-Person Hand Action
GCG	Grassmannian Conjugate Gradient
FML	Fréchet mean-based Metric Learning
LPE	Log-Pooling-Exp
M-LPE	Mean pooling-based LPE
LMKML	Localized Multi-Kernel Metric Learning
PML	Projection Metric Learning
LEML	Log-Euclidean Metric Learning
HERML	Hybrid Euclidean Riemannian Metric Learning
SPDML	SPD manifold Metric Learning
GEMKML	Graph-Embedding Multi-Kernel Metric Learning
DeepO2P	Deep Second-Order Pooling
SPDNet	SPD manifold Network
SymNet	Symmetric positive definite manifold Network
HRNN	Hierarchical Recurrent Neural Network
LSTM	Long Short-Term Memory
JOULE	JOintly heterogeneoUs features LEarning
TF	Transition Forests
TCN	Temporal Convolutional Network
ST-GCN	Spatial-Temporal Graph Convolutional Network
H+O	Hand and Object model
TTN	Temporal Transformer Network
SPDNetR	SPDNet with RBR module
SPDNetP	SPDNet with Riemannian Pooling module
SPDNetRP	SPDNet with both RBR and Riemannian Pooling modules

3. Proposed method

The suggested two Riemannian operation modules for SPD matrix discriminative learning are detailedly introduced in this section. The schematic diagram of the proposed approach is illustrated in Fig. 1. To make the readers self-contained, the Riemannian manifold of SPD matrices and the Log-Euclidean Metric on the SPD manifold are briefly reviewed firstly. Then, this paper recalls the basic layers of SPDNet (Huang & Van Gool, 2017). For simplicity, the used symbols in this article and their corresponding explanations are listed in Table 1. Besides, the acronyms and their corresponding full names used in this paper are summarized in Table 2.

3.1. Preliminaries

The Riemannian Manifold of SPD Matrices: For all non-zero $\boldsymbol{v} \in \mathbb{R}^d$, a real SPD matrix $\boldsymbol{X} \in \mathbb{R}^{d\times d}$ has an intrinsic property, which is $\boldsymbol{v}^T\boldsymbol{X}\boldsymbol{v} > 0$. The space formed by a family of d-by-d SPD matrices is the interior of a convex cone in the $d(d+1)/2$-dimensional Euclidean space, denoted as $\mathcal{S}_{++}^d$. As studied in Arsigny et al. (2007), when endowing $\mathcal{S}_{++}^d$ with an appropriate Riemannian metric, a specific Riemannian manifold, *i.e.*, SPD manifold, can be generated. Since the topological space of SPD manifold locally conforms to the Euclidean geometry with globally defined differential structure, the derivatives of the curves at point $\boldsymbol{X}_i$ ($\boldsymbol{X}_i \in \mathcal{S}_{++}^d$) can be defined under the matrix logarithm map: $\log_{\boldsymbol{X}_i} : \mathcal{S}_{++}^d \rightarrow \mathcal{T}_{\boldsymbol{X}_i}\mathcal{S}_{++}^d$, where $\mathcal{T}_{\boldsymbol{X}_i}\mathcal{S}_{++}^d$ represents the tangent space of $\mathcal{S}_{++}^d$ at point $\boldsymbol{X}_i$. The group of

inner products $\langle , \rangle_{\boldsymbol{X}_i}$ on all the tangent spaces is known as the Riemannian metric.

Log-Euclidean Metric on the SPD Manifold: As studied in Arsigny et al. (2007), under the group operation $\boldsymbol{X}_i \odot \boldsymbol{X}_j := \exp(\log(\boldsymbol{X}_i)+\log(\boldsymbol{X}_j))$ ($\boldsymbol{X}_i, \boldsymbol{X}_j \in \mathcal{S}^d_{++}$ and $\exp(\cdot)$, $\log(\cdot)$ respectively represent the normal matrix exponential and logarithm operators), the space of SPD matrices is given a commutative Lie group structure. Due to any bi-invariant metric $\langle , \rangle$ on the Lie group of SPD matrices corresponds to a Euclidean metric in the SPD matrix logarithmic domain, *i.e.*, the tangent space at identity matrix $\boldsymbol{I}$, it is also called a Log-Euclidean Metric.

Specifically, for any two tangent elements $\mathcal{T}_i, \mathcal{T}_j$, their scalar product in $\mathcal{T}_{\boldsymbol{X}}\mathcal{S}^d_{++}$ is given as:

$$\langle \mathcal{T}_i, \mathcal{T}_j \rangle_{\boldsymbol{X}} = \langle D_{\boldsymbol{X}}\mathcal{M}_{\boldsymbol{X}^{-1}}.\mathcal{T}_i, D_{\boldsymbol{X}}\mathcal{M}_{\boldsymbol{X}^{-1}}.\mathcal{T}_j \rangle, \tag{1}$$

where $\mathcal{M}_{\boldsymbol{X}} : \mathcal{S}^d_{++} \to \mathcal{S}^d_{++}$ is the logarithmic multiplication by $\boldsymbol{X}$. Due to $D_{\boldsymbol{X}}\mathcal{M}_{\boldsymbol{X}^{-1}} = D_{\boldsymbol{X}} \log$, Eq. (1) can be rewritten as:

$$\langle \mathcal{T}_i, \mathcal{T}_j \rangle_{\boldsymbol{X}} = \langle D_{\boldsymbol{X}} \log. \mathcal{T}_i, D_{\boldsymbol{X}} \log. \mathcal{T}_j \rangle_{\boldsymbol{I}}, \tag{2}$$

where $D_{\boldsymbol{X}} \log. \mathcal{T}$ is the directional derivative of the matrix logarithm at $\boldsymbol{X}$ along $\mathcal{T}$. The logarithm map associated to the Riemannian metric is defined in terms of matrix logarithms

$$\log_{\boldsymbol{X}_i}(\boldsymbol{X}_j) = D_{\log(\boldsymbol{X}_i)}\exp.(\log(\boldsymbol{X}_j) - \log(\boldsymbol{X}_i)). \tag{3}$$

Due to the differentiation of the equality $\log \circ \exp = \boldsymbol{I}$, $D_{\log_{\boldsymbol{X}}}\exp. = (D_{\boldsymbol{X}} \log.)^{-1}$. Similarly, the matrix exponential map can be expressed as:

$$\exp_{\boldsymbol{X}_i}(\mathcal{T}_j) = \exp(\log(\boldsymbol{X}_i) + D_{\boldsymbol{X}_i} \log. \mathcal{T}_j). \tag{4}$$

Combining Eqs. (2), (3), and (4), the LEM on the SPD manifold can be formulated as:

$$\begin{aligned} \mathcal{D}_{lem}(\boldsymbol{X}_i, \boldsymbol{X}_j) &= \langle \log_{\boldsymbol{X}_i}(\boldsymbol{X}_j), \log_{\boldsymbol{X}_i}(\boldsymbol{X}_j) \rangle_{\boldsymbol{X}_i} \\ &= \| \log(\boldsymbol{X}_j) - \log(\boldsymbol{X}_i) \|^2_{\mathrm{F}}. \end{aligned} \tag{5}$$

Compared to AIRM, LEM works directly in the space of SPD matrices logarithms, resulting in higher computational efficiency. Accordingly, it is selected as the similarity measurement in this paper. For more detailed information about this Riemannian metric, please kindly refer to Arsigny et al. (2007) and Absil et al. (2009).

3.2. Basic architecture of SPDNet

The SPDNet architecture is generalized from the paradigm of Euclidean networks with a two-stage representation learning. The first stage is a stack of trainable blocks, each of which containing a BiMap layer and a ReEig layer for SPD matrix transformation and nonlinear activation. The second stage makes up of a LogEig layer for the generation of Euclidean representations, followed by a normally used fully connected layer for visual classification tasks. Assuming $\boldsymbol{X}_{k-1} \in \mathbb{R}^{d_{k-1}\times d_{k-1}}$ be the input SPD matrix of the kth layer, the operation layers of the SPDNet are defined as follows:

BiMap Layer: This layer is similar to the usual dense layer, used to project each input SPD matrix into a new one via a bilinear mapping f_b, expressed as $\boldsymbol{X}_k = f_b^{(k)}(\boldsymbol{\mathcal{W}}_k, \boldsymbol{X}_{k-1}) = \boldsymbol{\mathcal{W}}_k^T \boldsymbol{X}_{k-1} \boldsymbol{\mathcal{W}}_k$, where $\boldsymbol{\mathcal{W}}_k \in \mathbb{R}^{d_{k-1}\times d_k}$ ($d_k < d_{k-1}$) is the to-be-learned transformation matrix and $\boldsymbol{X}_k \in \mathbb{R}^{d_k \times d_k}$ is the resulting matrix. To guarantee the SPD property of $\boldsymbol{X}_k$, $\boldsymbol{\mathcal{W}}_k$ is required to be column full-rank (proved in Section III-A of Wang, Wu, Kittler (2021)).

ReEig Layer: This layer plays an analogous role as that introduced in the classical ReLU-like layers. It is designed to inject nonlinearity for SPDNet by tuning-up the small positive eigenvalues of each input SPD matrix with a nonlinear rectification function f_r, formulated as $\boldsymbol{X}_k = f_r^{(k)}(\boldsymbol{X}_{k-1}) = \boldsymbol{U}\max(\epsilon \boldsymbol{I}, \boldsymbol{\Sigma})\boldsymbol{U}^T$. Wherein, ϵ is a small activation threshold and $\boldsymbol{X}_{k-1} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^T$ denotes the eigenvalue decomposition.

LogEig Layer: To embed the output feature manifolds into a flat space for further Euclidean computations, the LogEig layer is designed with the following mathematical form: $\boldsymbol{X}_k = f_l^{(k)}(\boldsymbol{X}_{k-1}) = \boldsymbol{U} \log(\boldsymbol{\Sigma})\boldsymbol{U}^T$, where $\boldsymbol{X}_{k-1} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^T$ denotes the eigenvalue decomposition. In SPDNet, the LogEig layer is followed by the conventional dense layer and the cross-entropy loss ℓ for visual classification tasks.

In this paper, the basic architecture of the SPD manifold neural network is constructed by 8 operation layers: $\boldsymbol{X}_0 \to f_b^{(1)} \to f_r^{(2)} \to f_b^{(3)} \to f_r^{(4)} \to f_b^{(5)} \to f_l^{(6)} \to f_{fc}^{(7)} \to f_{cl}^{(8)}$, where f_b, f_r, f_l, f_{fc}, and f_{cl} denote the layers of BiMap, ReEig, LogEig, FC, and cross-entropy loss, respectively. Besides, the connection weights of the BiMap layers are initialized as random semi-orthogonal matrices, and their appropriate sizes on the used AFEW, MDSD, CG, and FPHA datasets are determined by cross-validation in the experiments.

3.3. Riemannian batch regularization layer

Let $\mathcal{G} = [\boldsymbol{S}_1, \boldsymbol{S}_2, \ldots, \boldsymbol{S}_N]$ be the gallery composed by N image sets and $L = [l_1, l_2, \ldots, l_N] \in \mathbb{R}^{1\times N}$ be the corresponding label vector. In the training set $\mathcal{G}$, $\boldsymbol{S}_i = [\boldsymbol{s}_i^1, \boldsymbol{s}_i^2, \ldots, \boldsymbol{s}_i^{n_i}] \in \mathbb{R}^{d\times n_i}$ denotes the ith image set (video sequence) with n_i instances, where $i = 1 \to N$ and $\boldsymbol{s}_i^r \in \mathbb{R}^{d\times 1}$ represents the rth vectorized frame. With these definitions, the covariance representation of $\boldsymbol{S}_i$ can be computed by: $\boldsymbol{X}_i = \frac{1}{n_i - 1}\sum_{r=1}^{n_i}(\boldsymbol{s}_i^r - \boldsymbol{\mu}_i)(\boldsymbol{s}_i^r - \boldsymbol{\mu}_i)^T$. To ensure that $\boldsymbol{X}_i$ meets positive definiteness, a widely used trick is imposed on it, *i.e* $\boldsymbol{X}_i \leftarrow \boldsymbol{X}_i + \eta \boldsymbol{I}_d$, where η is a perturbation parameter, configured as $10^{-3} \times trace(\boldsymbol{X}_i)$ in the experiments. In this paper, the SPD representations of $\mathcal{G}$ is denoted by $\boldsymbol{\mathcal{X}} = [\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_N]$.

In the context of SPD manifold neural networks, most architectures just exploit a single cross-entropy loss to supervise the whole network, neglecting the peculiarities of the intra- and inter-class data distribution in the SPD matrix learning process. As a consequence, the variability information conveyed by the inputs has not been encoded explicitly during training, making the resulting lower-dimensional geometric representations of each transformation layer may not be qualified to yield a powerful network. Inspired by the fact that metric learning explores the metric space as a function of data distribution to find the best way for similarity measurement, this paper designs a Riemannian operation module for SPDNet to alleviate the issue mentioned above. This module is located before the BiMap layer, mainly used to provide more proper initial filters for the BiMap layers in each forward pass by encoding and learning the variations of representations. As a consequence, a discriminative feature manifold can be generated, in which SPD points from the same category are mapped closely to each other and those from different categories are separated by a large manifold margin. This requirement could be formulated in terms of the following metric learning regularization term:

$$\begin{aligned} \mathcal{L}(\boldsymbol{\mathcal{W}}_k; \mathfrak{A}, \mathfrak{B}) = &\frac{1}{N_{\mathfrak{A}}} \sum_{i,j\in\mathfrak{A}} \max(\mathcal{D}_{lem}^{\boldsymbol{\mathcal{W}}_k}(\boldsymbol{X}_k^{i,o}, \boldsymbol{X}_k^{j,+}), \xi_{\mathfrak{A}}) \\ &- \frac{1}{N_{\mathfrak{B}}} \sum_{i,j\in\mathfrak{B}} \max(\xi_{\mathfrak{B}} - \mathcal{D}_{lem}^{\boldsymbol{\mathcal{W}}_k}(\boldsymbol{X}_k^{i,o}, \boldsymbol{X}_k^{j,-}), 0) \\ &+ \gamma \|\boldsymbol{\mathcal{W}}_k\|^2_{\mathrm{F}}, \end{aligned} \tag{6}$$

where $(\boldsymbol{X}_k^{i,o}, \boldsymbol{X}_k^{j,+}, \boldsymbol{X}_k^{j,-})$ is a triplet, and the RBR layer mines all the possible and unduplicated triplets in a batch of SPD matrices in the kth layer. This objective function expects that the Riemannian distance of the pair $(\boldsymbol{X}_k^{i,o}, \boldsymbol{X}_k^{j,+})$ is smaller than a manifold margin $\xi_{\mathfrak{A}}$, while the Riemannian distance of the pair $(\boldsymbol{X}_k^{i,o}, \boldsymbol{X}_k^{j,-})$ is larger

Table 3
Accuracy comparison (%) on the AFEW, MDSD, CG, and FPHA datasets.

Methods	AFEW	MDSD	CG	FPHA
SPDNet-$\mathcal{L}$-v1	34.84	36.92	88.19	86.96
SPDNet-$\mathcal{L}$-v2, *i.e.*, SPDNet-RBR	**35.31**	**38.46**	**88.75**	**87.57**

than another manifold margin $\xi_{\mathfrak{B}}$ simultaneously. The specific form of $\mathcal{D}_{lem}^{\mathcal{W}_k}(\boldsymbol{X}_k^{i,o}, \boldsymbol{\Omega})$ is given as below, where $\boldsymbol{\Omega}$ denotes $\boldsymbol{X}_k^{j,+}$ or $\boldsymbol{X}_k^{j,-}$.

$$\mathcal{D}_{lem}^{\mathcal{W}_k}(\boldsymbol{X}_k^{i,o}, \boldsymbol{\Omega}) = \|\mathcal{W}_k^T \log(\boldsymbol{X}_k^{i,o})\mathcal{W}_k - \mathcal{W}_k^T \log(\boldsymbol{\Omega})\mathcal{W}_k\|_{\mathrm{F}}^2. \tag{7}$$

To obtain the optimal solution of $\mathcal{W}_k$, this work follows Huang and Van Gool (2017) to assume them to be semi-orthogonal, *i.e.*, $\mathcal{W}_k^T\mathcal{W}_k = I_k$, such that they lie in a compact Stiefel manifold $St(d_k, d_{k-1})$ (Edelman, Arias, & Smith, 1998). Moreover, a Frobenius norm constraint is imposed on the target transformation matrices to restrict their scale.

In the algorithm design, the proposed RBR module is trained in conjunction with SPDNet through Riemannian matrix backpropagation. Accordingly, the initial weights of the RBR layers are set to be the same as that of the corresponding BiMap layers. It is clear to see that the proposed metric learning-based Riemannian batch regularization is different from the fashion of conventional deep metric learning that aims to train a discriminative network embedding by encoding and learning high-level data distributions. To better demonstrate the motivation of designing the RBR module, this work carries out classification experiments on the AFEW, MDSD, CG, and FPHA datasets (see Section 5.2 for details) to measure the influence of different supervision mechanisms on the model learning ability. The results are tabulated in Table 3. From this table, it is evident that the classification scores of SPDNet-$\mathcal{L}$-v1 that appends the metric learning regularization term to the last BiMap layer of SPDNet and supervises the whole network together with cross-entropy loss is lower than that of SPDNet-RBR studied in this paper. The essential reason is that the original data variations embodied in the high-level structured representations with richer semantic information is inconspicuous, inhibiting SPDNet-$\mathcal{L}$-v1 to adequately capture, encode, and analyze the geometry of data distributions. In contrast, the proposed RBR method performs metric learning in different feature levels of SPDNet, being qualified to yield new feature manifolds with improved discriminability.

3.4. RiemannIan pooling layer

In the context of Euclidean networks, pooling layers with mean, max, and min metrics are often utilized to cut down the number of parameters and introduce robustness to registration errors. In this scenario, designing similar operations for the Riemannian neural networks may also be imperative. Considering the non-Euclidean geometry of the representations, the introduced pooling layers should be faithful to the geometrical structure of SPD matrices. As an exploration, the Riemannian pooling operation proposed in this paper is implemented in three steps. Considering the strong theoretical and practical interest of Riemannian barycenter in Riemannian data analysis (Pennec et al., 2006), the first step is to compute the Riemannian barycenter $\mathcal{Z}_k$ of a batch of input SPD matrices $\{{}^b\boldsymbol{X}_{k-1}^m\}_{b=1}^B$ using the Fréchet formulation instead of the arithmetic mean. This can be formulated as:

$$\mathcal{Z}_k^* = \arg\min_{\mathcal{Z}_k \in \mathcal{S}_{++}^{d_{k-1}}} \sum_{b=1}^{B}\sum_{m=1}^{m_k} \mathcal{D}_{lem}({}^b\boldsymbol{X}_{k-1}^m, \mathcal{Z}_k), \tag{8}$$

where ${}^b\boldsymbol{X}_{k-1}^m$ represents the generated mth ($m = 1 \to m_k$) feature map in the $(k-1)$th layer with respect to the bth sample in a given batch.

Theorem 1. *The Fréchet mean of a batch of SPD matrices ${}^b\boldsymbol{X}_{k-1}^m$ with respect to $\mathcal{D}_{lem}$ is:*

$$\mathcal{Z}_k^* = \exp\left[\frac{1}{Bm_k}\sum_{b=1}^{B}\sum_{m=1}^{m_k}\log({}^b\boldsymbol{X}_{k-1}^m)\right]. \tag{9}$$

Proof. For Eq. (8), the Fréchet mean must satisfy

$$\frac{\partial \sum_{b=1}^{B}\sum_{m=1}^{m_k}\mathcal{D}_{lem}({}^b\boldsymbol{X}_{k-1}^m, \mathcal{Z}_k)}{\partial \mathcal{Z}_k} = 0. \tag{10}$$

Given that

$$\frac{\partial \mathcal{D}_{lem}({}^b\boldsymbol{X}_{k-1}^m, \mathcal{Z}_k)}{\partial \mathcal{Z}_k} = -2\mathcal{Z}_k^{-1}[\log({}^b\boldsymbol{X}_{k-1}^m) - \log(\mathcal{Z}_k)], \tag{11}$$

the result shown in Eq. (9) can be obtained. □

In the second step, the pooling operation can be expressed as the following unsupervised metric learning problem that aims to find an orthogonal mapping for the generation of a lower-dimensional manifold with maximum data variance:

$$\begin{aligned}&\mathcal{H}_k^* = \arg\max_{\mathcal{H}_k \in \mathbb{R}^{d_{k-1}\times d_k}} \sum_{b=1}^{B}\sum_{m=1}^{m_k} J(\mathcal{H}_k),\\ &\text{s.t. } \mathcal{H}_k^T\mathcal{H}_k = \boldsymbol{I}_{d_k},\end{aligned} \tag{12}$$

where $J(\mathcal{H}_k) = \mathcal{D}_{lem}^{\mathcal{H}_k}({}^b\boldsymbol{X}_{k-1}^m, \mathcal{Z}_k) = \|\mathcal{H}_k^T \log({}^b\boldsymbol{X}_{k-1}^m)\mathcal{H}_k - \mathcal{H}_k^T \log(\mathcal{Z}_k)\mathcal{H}_k\|_{\mathrm{F}}^2$.

In the third step, the task is to learn the embedding mapping $\mathcal{H}_k$. It is easy to check that Eq. (12) is an optimization problem on the Grassmannian manifold ($J(\mathcal{H}_k) = J(\mathcal{H}_k\boldsymbol{P})$ for any d_k-by-d_k orthogonal matrix $\boldsymbol{P}$) that can be solved by utilizing the Grassmannian Conjugate Gradient (GCG) method (see Section 3.3 of Edelman et al. (1998)). With the following Euclidean gradient

$$\begin{aligned}&\nabla_{\mathcal{H}_k}\left[\sum_{b=1}^{B}\sum_{m=1}^{m_k}J(\mathcal{H}_k)\right]\\ &= \nabla_{\mathcal{H}_k} trace\left[\mathcal{H}_k^T\left(\sum_{b=1}^{B}\sum_{m=1}^{m_k}\boldsymbol{\Upsilon}\mathcal{H}_k\mathcal{H}_k^T\boldsymbol{\Upsilon}\right)\mathcal{H}_k\right]\\ &= 4\sum_{b=1}^{B}\sum_{m=1}^{m_k}\boldsymbol{\Upsilon}\mathcal{H}_k\mathcal{H}_k^T\boldsymbol{\Upsilon}\mathcal{H}_k,\end{aligned} \tag{13}$$

where $\boldsymbol{\Upsilon} = \log({}^b\boldsymbol{X}_{k-1}^m) - \log(\mathcal{Z}_k)$, the optimization procedure can be easily and effectively implemented in the experiments by making use of the *manopt* package (Boumal, Mishra, Absil, & Sepulchre, 2014).

Finally, the output SPD matrices of this pooling layer are described as: $\boldsymbol{X}_k = f_p^{(k)}(\mathcal{H}_k, \boldsymbol{X}_{k-1}) = \mathcal{H}_k^T\boldsymbol{X}_{k-1}\mathcal{H}_k$. This pooling method is called FML (Fréchet Mean-based Metric Learning) in the experiments. Since $\mathcal{W}_k^T\mathcal{Z}_k\mathcal{W}_k \neq \arg\min\sum_{b=1}^{B}\sum_{m=1}^{m_k}\mathcal{D}(\mathcal{W}_k^T({}^b\boldsymbol{X}_{k-1}^m)\mathcal{W}_k, \mathcal{Z}_k)$, if directly applying the BiMap operation to $\mathcal{Z}_k$ of the kth layer, the generated new point does not necessarily denote the Fréchet mean (Riemannian barycenter) of the data in the transformed lower-dimensional space $\mathcal{S}_{++}^{d_k}$, thus may lead to sub-optimal solutions. Based on this consideration, the low-rank projection is imposed on the learned Riemannian mean to achieve the maximum data variance via Eq. (12), which is theoretically capable of capturing more pivotal data variations.

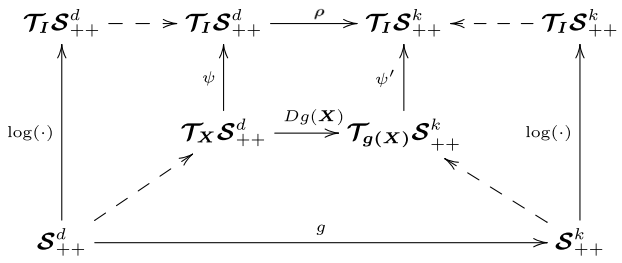


Fig. 2. Conceptual illustration of the proposed Riemannian pooling method in the domain of SPD matrix logarithms. With a tangent map $Dg(\boldsymbol{X}) : \mathcal{T}_{\boldsymbol{X}}\mathcal{S}^d_{++} \to \mathcal{T}_g(\boldsymbol{X})\mathcal{S}^k_{++}$ from the original tangent space $\mathcal{T}_{\boldsymbol{X}}\mathcal{S}^d_{++}$ to a more discriminative tangent space $\mathcal{T}_g(\boldsymbol{X})\mathcal{S}^k_{++}$, the corresponding manifold-to-manifold mapping $g : \mathcal{S}^d_{++} \to \mathcal{S}^k_{++}$ can be derived. However, this requires that the mapping $Dg(\boldsymbol{X})$ is an injection (or surjection) for $\mathcal{T}_{\boldsymbol{X}}\mathcal{S}^d_{++}$ at every $\boldsymbol{X} \in \mathcal{S}^d_{++}$. Since $\mathcal{T}_{\boldsymbol{X}}\mathcal{S}^d_{++}$ and $\mathcal{T}_{\boldsymbol{I}}\mathcal{S}^d_{++}$ at the identity matrix are isomorphic, it is only necessary to construct a tangent map $\rho : \mathcal{T}_{\boldsymbol{I}}\mathcal{S}^d_{++} \to \mathcal{T}_{\boldsymbol{I}}\mathcal{S}^k_{++}$ of injectivity or surjectivity to induce an immersion (or submersion) g. Here, ψ and ψ' denote linear isomorphic mapping.

Due to the Riemannian tangent space complies with the Euclidean geometry, according to the following lemmas, the Riemannian pooling operation also can be carried out in the domain of SPD matrix logarithms.

Lemma 1. *If and only if the tangent map $Dg(\boldsymbol{X}) : \mathcal{T}_{\boldsymbol{X}}\mathcal{S}^d_{++} \to \mathcal{T}_g(\boldsymbol{X})\mathcal{S}^k_{++}$ is an injection (or surjection) for every $\boldsymbol{X} \in \mathcal{S}^d_{++}$, then the mapping $g : \mathcal{S}^d_{++} \to \mathcal{S}^k_{++}$ from manifold to manifold is an immersion (or submersion) (Arsigny et al., 2007; Huang, Wang, Shan, Li et al., 2015).*

According to Lemma 1, Lemma 2 can be derived.

Lemma 2. *If a continuously differentiable map $\rho : \mathcal{T}_{\boldsymbol{I}}\mathcal{S}^d_{++} \to \mathcal{T}_{\boldsymbol{I}}\mathcal{S}^k_{++}$ is an injection (or surjection), then an immersion (or submersion) mapping $g : \mathcal{S}^d_{++} \to \mathcal{S}^k_{++}$ from manifold to manifold can be derived.*

Proof. For a better understanding, this paper constructs a commutative graph, shown in Fig. 2. For every $\boldsymbol{X} \in \mathcal{S}^d_{++}$, it can be obtained that $\boldsymbol{X}' = \exp(\rho(\log(\boldsymbol{X}))) \in \mathcal{S}^k_{++}$. Since $\mathcal{T}_{\boldsymbol{I}}\mathcal{S}^d_{++}$ and $\mathcal{T}_{\boldsymbol{I}}\mathcal{S}^k$ are Euclidean spaces, their tangent spaces are themselves. Due to the diffeomorphism of $\log(\cdot)$, there exists a linear isomorphism ψ between the tangent spaces $\mathcal{T}_{\boldsymbol{X}}\mathcal{S}^d_{++}$ and $\mathcal{T}_{\boldsymbol{I}}\mathcal{S}^d_{++}$. For the same reason, ψ' is a linear isomorphism. Therefore, the tangent map $Dg(\boldsymbol{X})$ can be derived via the composition mapping $\psi'^{-1} \circ \rho \circ \psi$. Owing to the injectivity (or surjectivity) of the inferred $Dg(\boldsymbol{X})$ for every $\boldsymbol{X} \in \mathcal{S}^d_{++}$, the immersion (or submersion) $g : \mathcal{S}^d_{++} \to \mathcal{S}^k_{++}$ exists. □

To this end, three auxiliary layers need to be constructed. In particular, the input SPD matrices $\boldsymbol{X}_{k-1}$ are first mapped to the logarithmic domain (tangent space at identity matrix: $\mathcal{T}_{\boldsymbol{I}}\mathcal{S}^d_{++}$) by the LogEig operation introduced before. As the resulting matrices $\boldsymbol{X}_k$ lie on Euclidean space, any injective or surjective pooling function that satisfies continuous differentiability can be applied to process them. Finally, an ExpEig layer is designed to map these pooled data back onto the SPD manifold via the matrix exponential function. The pooling method studied here is named LPE (Log-Pooling-Exp). Considering that the widely used Euclidean mean pooling function meets the conditions (surjective and continuously differentiable), the mean-based LPE (M-LPE) strategy is utilized to make experiments, and the pooling window size is set to 2×2.

ExpEig Layer: This auxiliary layer is used to transform the corresponding matrix representations $\boldsymbol{X}_k$ from the tangent space to the SPD manifold via the matrix exponential function. It is defined as $\boldsymbol{X}_{k+1} = f_e^{(k+1)}(\boldsymbol{X}_k) = \boldsymbol{U}\exp(\Sigma)\boldsymbol{U}^T$, where $\boldsymbol{X}_k = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^T$ is the eigenvalue decomposition, and $\exp(\boldsymbol{\Sigma})$ represents the exponential operation on the diagonal elements of $\boldsymbol{\Sigma}$.

3.5. Cross-entropy loss

In this paper, the cross-entropy loss is defined as the objective function of the proposed network, and its specific form is given below:

$$\ell(\boldsymbol{\theta}, \mathcal{U}_{fc}; \boldsymbol{\mathcal{X}}) = -\sum_{i=1}^{N}\sum_{\alpha=1}^{c} \tau(l_i, \alpha) \times \log \frac{e^{\mathcal{U}_{fc}^{\alpha}\tilde{\boldsymbol{X}}_{K-1}^{i}}}{\sum_o e^{\mathcal{U}_{fc}^{o}\tilde{\boldsymbol{X}}_{K-1}^{i}}}, \tag{14}$$

where $\boldsymbol{\theta}$ signifies the key parameters of the SPD network to be learned, $\mathcal{U}_{fc}^{\alpha}$ represents the α-th row of the to-be-learned projection matrix $\mathcal{U}_{fc}$ of the FC layer, $\tilde{\boldsymbol{X}}_{K-1}^{i}$ denotes the vectorized form of the ith output $\boldsymbol{X}_{K-1}^{i}$ of the $(K-1)$th layer (LogEig layer), and $\tau(l_i, \alpha)$ is an indicator defined as:

$$\tau(l_i, \alpha) = \begin{cases} 1, & \text{if } l_i = \alpha, \\ 0, & \text{otherwise.} \end{cases} \tag{15}$$

In the test stage, a given test image set $\boldsymbol{S}_{te}$ is firstly encoded as an SPD matrix $\boldsymbol{X}_{te}$. Then, by feeding $\boldsymbol{X}_{te}$ into the well-trained SPD manifold neural network, a low dimensional and compact feature matrix can be produced at the LogEig layer, denoted as $\mathfrak{X}_{te}$. Afterward, the probability of $\boldsymbol{S}_{te}$ belonging to class α can be computed by:

$$P(\alpha|\boldsymbol{S}_{te}) = \frac{e^{\mathcal{U}_{fc}^{\alpha}\tilde{\mathfrak{X}}_{te}}}{\sum_o e^{\mathcal{U}_{fc}^{o}\tilde{\mathfrak{X}}_{te}}}, \tag{16}$$

where $\tilde{\mathfrak{X}}_{te}$ represents the vectorized form of $\mathfrak{X}_{te}$. Finally, the candidate class which has the maximum probability is declared as the label of $\boldsymbol{S}_{te}$. This can be mathematically expressed as: $l_{te} = \arg\max_{\alpha} P(\alpha|\boldsymbol{S}_{te})$.

4. Backpropagation procedures

The SPD manifold neural network introduced in this paper can be considered as the data embedding model $(f, \mathcal{W})$, where $f = f^{(K)} \circ f^{(K-1)} \circ \cdots \circ f^{(2)} \circ f^{(1)}$ represents a series of successive function compositions and $\mathcal{W} = \{\mathcal{W}^K, \mathcal{W}^{K-1}, \ldots, \mathcal{W}^2, \mathcal{W}^1\}$ denotes the parameter tuple, satisfying the properties of metric spaces. Here, K signifies the number of layers of the network, $f^{(k)}$ and $\mathcal{W}^k$ are respectively the operation function and weight parameter of the kth layer. The loss of the kth layer can be formulated as $L^{(k)} = \ell \circ f^{(K)} \circ \ldots \circ f^{(k)}$.

BiMap Layer: Since the differentiation of $\boldsymbol{X}_k$ in the BiMap layer is computed by:

$$d\boldsymbol{X}_k = d\mathcal{W}_k^T\boldsymbol{X}_{k-1}\mathcal{W}_k + \mathcal{W}_k^T d\boldsymbol{X}_{k-1}\mathcal{W}_k + \mathcal{W}_k^T\boldsymbol{X}_{k-1}d\mathcal{W}_k, \tag{17}$$

the following chain rule can be obtained on the basis of the invariance of the first-order differential:

$$\frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k} : d\boldsymbol{X}_k = \frac{\partial L^{(k)}}{\partial \mathcal{W}_k} : d\mathcal{W}_k + \frac{\partial L^{(k)}}{\partial \boldsymbol{X}_{k-1}} : d\boldsymbol{X}_{k-1}. \tag{18}$$

Substituting the left-hand side of Eq. (18) with Eq. (17) and according to the matrix inner product ":" properties (Ionescu, Vantzos, & Sminchisescu, 2015), the following three equations can be obtained:

$$\frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k} : d\mathcal{W}_k^T\boldsymbol{X}_{k-1}\mathcal{W}_k = \frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k}\boldsymbol{X}_{k-1}\mathcal{W}_k : d\mathcal{W}_k, \tag{19}$$

$$\frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k} : \mathcal{W}_k^T\boldsymbol{X}_{k-1}d\mathcal{W}_k = \frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k}\boldsymbol{X}_{k-1}\mathcal{W}_k : d\mathcal{W}_k, \tag{20}$$

$$\frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k} : \mathcal{W}_k^T d\boldsymbol{X}_{k-1}\mathcal{W}_k = \mathcal{W}_k \frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k}\mathcal{W}_k^T : d\boldsymbol{X}_{k-1}. \tag{21}$$

At this point, the left-hand side of Eq. (18) can be rewritten as:

$$2\frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k}\boldsymbol{X}_{k-1}\mathcal{W}_k : d\mathcal{W}_k + \mathcal{W}_k \frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k}\mathcal{W}_k^T : d\boldsymbol{X}_{k-1}. \tag{22}$$

As Eq. (22) equal to the right-hand of Eq. (18), the following partial derivatives can be derived:

$$\frac{\partial L^{(k)}}{\partial \mathcal{W}_k} = 2\boldsymbol{X}_{k-1}\mathcal{W}_k \frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k}, \tag{23}$$

$$\frac{\partial L^{(k)}}{\partial \boldsymbol{X}_{k-1}} = \mathcal{W}_k \frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k}\mathcal{W}_k^T. \tag{24}$$

To make the readers self-contained, the updating rules of $\mathcal{W}_k$ are given below (Huang & Van Gool, 2017):

$$\tilde{\nabla} L^{(k)}_{\mathcal{W}_k} = \nabla L^{(k)}_{\mathcal{W}_k} - \nabla L^{(k)}_{\mathcal{W}_k}\mathcal{W}_k^T\mathcal{W}_k, \tag{25}$$

$$\mathcal{W}_k^{(t+1)} = \Gamma(\mathcal{W}_k^t - \alpha\tilde{\nabla} L^{(k)}_{\mathcal{W}_k}), \tag{26}$$

where $\tilde{\nabla} L^{(k)}_{\mathcal{W}_k}$ is the tangential component to the stiefel manifold, obtained by subtracting the normal component of the Euclidean gradient $\nabla L^{(k)}_{\mathcal{W}_k}$ computed by Eq. (23), $\mathcal{W}_k^t$ is the updated weight at the tth iteration, and α is the learning rate. For detailed information about the optimization procedures, please kindly refer to Absil et al. (2009).

RBR Layer: The partial derivatives of $\mathcal{L}(\mathcal{W}_k; \mathfrak{A}, \mathfrak{B})$ with respect $\mathcal{W}_k$ and $\boldsymbol{X}_k^{i,o}$ are computed as follows:

$$\frac{\partial \mathcal{L}}{\partial \mathcal{W}_k} = \frac{1}{N_{\mathfrak{A}}}\sum_{i,j\in\mathfrak{A}} \boldsymbol{\pi}_1^{(\boldsymbol{X}_k^{i,o},\mathcal{W}_k)} - \frac{1}{N_{\mathfrak{B}}}\sum_{i,j\in\mathfrak{B}} \boldsymbol{\pi}_2^{(\boldsymbol{X}_k^{i,o},\mathcal{W}_k)}, \tag{27}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{X}_k^{i,o}} = \frac{2}{N_{\mathfrak{A}}}\sum_{j\in\mathfrak{A}} \boldsymbol{\phi}_1^{(\boldsymbol{X}_k^{i,o},\mathcal{W}_k)} - \frac{2}{N_{\mathfrak{B}}}\sum_{j\in\mathfrak{B}} \boldsymbol{\phi}_2^{(\boldsymbol{X}_k^{i,o},\mathcal{W}_k)}, \tag{28}$$

where

$$\boldsymbol{\pi}_1 = \begin{cases} \mathcal{F}_1(\mathcal{W}_k\mathcal{W}_k^T)\mathcal{F}_1\mathcal{W}_k, & \text{if } \mathcal{D}_{lem}^{\mathcal{W}_k}(\boldsymbol{X}_k^{i,o}, \boldsymbol{X}_k^{j,+}) > \xi_{\mathfrak{A}}, \\ 0, & \text{otherwise.} \end{cases} \tag{29}$$

$$\boldsymbol{\pi}_2 = \begin{cases} \mathcal{F}_2(\mathcal{W}_k\mathcal{W}_k^T)\mathcal{F}_2\mathcal{W}_k, & \text{if } \mathcal{D}_{lem}^{\mathcal{W}_k}(\boldsymbol{X}_k^{i,o}, \boldsymbol{X}_k^{j,-}) < \xi_{\mathfrak{B}}, \\ 0, & \text{otherwise.} \end{cases} \tag{30}$$

$$\boldsymbol{\phi}_1 = \begin{cases} (\mathcal{W}_k\mathcal{W}_k^T)\mathcal{F}_1(\boldsymbol{X}_k^{i,o})^{-1}, & \text{if } \mathcal{D}_{lem}^{\mathcal{W}_k}(\boldsymbol{X}_k^{i,o}, \boldsymbol{X}_k^{j,+}) > \xi_{\mathfrak{A}}, \\ 0, & \text{otherwise.} \end{cases} \tag{31}$$

$$\boldsymbol{\phi}_2 = \begin{cases} (\mathcal{W}_k\mathcal{W}_k^T)\mathcal{F}_2(\boldsymbol{X}_k^{i,o})^{-1}, & \text{if } \mathcal{D}_{lem}^{\mathcal{W}_k}(\boldsymbol{X}_k^{i,o}, \boldsymbol{X}_k^{j,-}) < \xi_{\mathfrak{B}}, \\ 0, & \text{otherwise.} \end{cases} \tag{32}$$

Here, $\mathcal{F}_1 = \log(\boldsymbol{X}_k^{i,o}) - \log(\boldsymbol{X}_k^{j,+})$ and $\mathcal{F}_2 = \log(\boldsymbol{X}_k^{i,o}) - \log(\boldsymbol{X}_k^{j,-})$.

Given the input gradient $\frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k}$, the output gradient of the RBR layer is computed by $\frac{\partial L^{(k)}}{\partial \boldsymbol{X}_k^{\lambda}} = \frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k^{\lambda}} + \xi\frac{\partial \mathcal{L}}{\partial \boldsymbol{X}_k^{\lambda}}$, where ξ is a trade-off parameter and $\lambda = 1 \to B$. If $\lambda \in \mathfrak{A}$, $\frac{\partial \mathcal{L}}{\partial \boldsymbol{X}_k^{\lambda}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{X}_k^{i,o}}$, and 0 otherwise.

ExpEig Layer: Due to this layer involves SVD operation, a transition layer k^* is firstly introduced to receive $\boldsymbol{X}_{k-1}$ as input and output $\boldsymbol{U}$ and $\boldsymbol{\Sigma}$, *i.e.*, $\boldsymbol{X}_{k^*} = f_e^{(k^*)}(\boldsymbol{X}_{k-1}) = (\boldsymbol{U}, \boldsymbol{\Sigma})$. Accordingly, the chain rule of Eq. (18) in this scenario can be modified as:

$$\frac{\partial L^{(k)}}{\partial \boldsymbol{X}_{k-1}} : d\boldsymbol{X}_{k-1} = \frac{\partial L^{(k^*)}}{\partial \boldsymbol{U}} : d\boldsymbol{U} + \frac{\partial L^{(k^*)}}{\partial \boldsymbol{\Sigma}} : d\boldsymbol{\Sigma}. \tag{33}$$

Since $d\boldsymbol{X}_{k-1} = d\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^T + \boldsymbol{U}d\boldsymbol{\Sigma}\boldsymbol{U}^T + \boldsymbol{U}\boldsymbol{\Sigma}d\boldsymbol{U}^T$, the differentiations of $d\boldsymbol{U}$ and $d\boldsymbol{\Sigma}$ are formulated as (Huang & Van Gool, 2017; Ionescu et al., 2015):

$$d\boldsymbol{U} = 2\boldsymbol{U}(\boldsymbol{\Psi}^T \circ (\boldsymbol{\Sigma}^T\boldsymbol{U}^T d\boldsymbol{X}_{k-1}\boldsymbol{U})_{sym}) \tag{34}$$

$$d\boldsymbol{\Sigma} = (\boldsymbol{U}^T d\boldsymbol{X}_{k-1}\boldsymbol{U})_{diag}, \tag{35}$$

where $\circ$ represents the Hadamard product and $C_{sym} = (C + C^T)/2$. Putting Eqs. (34) and (35) into Eq. (33), the partial derivative of $L^{(k)}$ with respect to $\boldsymbol{X}_{k-1}$ for this layer can be computed by:

$$\frac{\partial L^{(k)}}{\partial \boldsymbol{X}_{k-1}} = 2\boldsymbol{U}(\boldsymbol{\Psi} \circ (\boldsymbol{U}^T\boldsymbol{Q}_1)_{sym})\boldsymbol{U}^T + \boldsymbol{U}\boldsymbol{Q}_2\boldsymbol{U}^T, \tag{36}$$

where $\boldsymbol{Q}_1 = \frac{\partial L^{(k^*)}}{\partial \boldsymbol{U}}$, $\boldsymbol{Q}_2 = (\frac{\partial L^{(k^*)}}{\partial \boldsymbol{\Sigma}})_{diag}$, and $\boldsymbol{\Psi}$ is expressed as:

$$\boldsymbol{\Psi}_{ij} = \begin{cases} \frac{1}{\sigma_i^2 - \sigma_j^2}, & i \neq j, \\ 0, & i = j, \end{cases} \tag{37}$$

where σ_i denotes the ith ($i = 1 \to d_{k-1}$) eigenvalue of $\boldsymbol{\Sigma}$.

Then, the kth layer receives $\boldsymbol{U}$ and $\boldsymbol{\Sigma}$ as its input and outputs $\boldsymbol{X}_k = \boldsymbol{U}\exp(\boldsymbol{\Sigma})\boldsymbol{U}^T$. Due to the variation of $\boldsymbol{X}_k$ is: $d\boldsymbol{X}_k = 2(d\boldsymbol{U}\exp(\boldsymbol{\Sigma})\boldsymbol{U}^T)_{sym} + (\boldsymbol{U}\exp(\boldsymbol{\Sigma})d\boldsymbol{\Sigma}\boldsymbol{U}^T)_{sym}$, the following partial derivatives can be derived by imitating the chain rule of Eq. (33):

$$\frac{\partial L^{(k^*)}}{\partial \boldsymbol{U}} = 2\left(\frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k}\right)_{sym}\boldsymbol{U}\exp(\boldsymbol{\Sigma}), \tag{38}$$

$$\frac{\partial L^{(k^*)}}{\partial \boldsymbol{\Sigma}} = \exp(\boldsymbol{\Sigma})\boldsymbol{U}^T\left(\frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k}\right)_{sym}\boldsymbol{U}. \tag{39}$$

For the gradients of SPD matrices in the ReEig and LogEig layers, the readers are referred to Huang and Van Gool (2017). With these ingredients, the *Riemannian matrix backpropagation* can therefore be exploited to train the SPD manifold neural network embedded with the designed modules. The main implementation details of the proposed approach are summarized into **Algorithm** 1.

5. Experiments

In this section, the effectiveness of the proposed approach[2] is evaluated on three different visual classification tasks, *i.e.*, video-based emotion recognition on the AFEW (Dhall, Goecke, Joshi, Sikka, & Gedeon, 2014) dataset, dynamic scene classification on the MDSD (Shroff, Turaga, & Chellappa, 2010) dataset, and hand action recognition on the CG (Kim & Cipolla, 2008) and FPHA (Garcia-Hernando, Yuan, Baek, & Kim, 2018) datasets, respectively.

5.1. Implementations details

In this paper, the learning rate λ is set to 0.01 on the four used datasets. The batch size B is set to 20 on the MDSD and CG datasets, and 30 on the AFEW and FPHA datasets. The activation threshold ϵ on the MDSD, AFEW, and CG datasets is set to 1e–4, and that on the FPHA dataset is configured as 1e–5. Besides, the trade-off parameter γ of Eq. (6) is configured as 0.2 on all the used datasets. To train the network, an i7-9700 (3.4 GHz) PC with 16 GB RAM is used.

5.2. Comparative methods and settings

For better evaluations, the following representative algorithms are selected for comparison, with the parameters of each method are tuned according to the original recommendations. To be specific, for LMKML (Lu, Wang, & Moulin, 2013), the learning rate α is set to 10^{-6} on all the involved datasets. For PML (Huang et al., 2015b), the values of the target dimensionality d and the

[2] The source code will be released on: https://github.com/GitWR.

Algorithm 1 Training the Refurbished SPD Neural Network

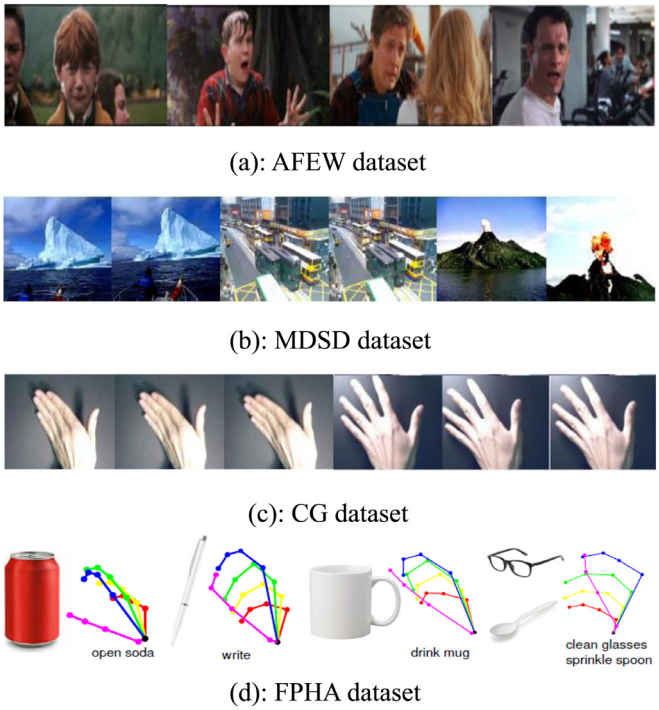
Input: Training set $\mathcal{G}$, training epochs E, learning rate η, batch size B, number of B of a given dataset $\mathbb{b}$, threshold ϵ, number of layers K, and trade-off coefficients γ, ξ.
Initialization:
1: ***for*** $i \leftarrow 1$ **to** N ***do***:
2: Computing $\boldsymbol{X}_i$ from $\boldsymbol{S}_i$ via $\boldsymbol{X}_i = \frac{1}{n_i-1}\sum_{r=1}^{n_i}(\boldsymbol{s}_i^r - \boldsymbol{\mu}_i)(\boldsymbol{s}_i^r - \boldsymbol{\mu}_i)^T$
3: Using $\boldsymbol{X}_i \leftarrow \boldsymbol{X}_i + \eta \boldsymbol{I}_d$ to ensure that $\boldsymbol{X}_i$ is SPD.
4: ***end***
5: ***for*** $k \leftarrow 1$ **to** K ***do***:
6: The SVD operation is applied to initialize $\mathcal{W}^k$ of the BiMap layer and
$\mathcal{H}_k$ of the Pooling layer.
7: Assigning $\mathcal{W}^k$ of the BiMap layer to the corresponding RBR layer.
8: ***end***
Training:
9: ***for*** $t \leftarrow 1$ **to** E ***do***:
10: Randomly and equally divide the new training set $\mathcal{X}$ into $\mathbb{b}$ batches:
$\mathcal{X} = \text{random}[\mathcal{X}_1, \mathcal{X}_2, ..., \mathcal{X}_{\mathbb{b}}]$.
11: ***for*** $s \leftarrow 1$ **to** $\mathbb{b}$ ***do***:
12: ***for*** $k \leftarrow 1$ **to** K ***do***:

Forward Pass:
case 'RBR layer'
13: Computing the distance $\mathcal{D}_{lem}^{\mathcal{W}_k}(\boldsymbol{X}_k^{i,o}, \boldsymbol{\Omega})$ using Eq. (7).
14: Computing the metric loss $\mathcal{L}(\mathcal{W}_k; \mathfrak{A}, \mathfrak{B})$ using Eq. (6).
case 'BiMap layer'
15: $\boldsymbol{X}_k = f_b^{(k)}(\mathcal{W}_k, \boldsymbol{X}_{k-1}) = \mathcal{W}_k^T \boldsymbol{X}_{k-1} \mathcal{W}_k$.
case 'Pooling layer'
16: $\mathcal{Z}_k^* = \exp[\frac{1}{Bm_k}\sum_{b=1}^{B}\sum_{m=1}^{m_k}\log({}^b\boldsymbol{X}_{k-1}^m)]$.
17: Computing the metric loss $J(\mathcal{H}_k)$ using Eq. (12).
18: $\boldsymbol{X}_k = f_p^{(k)}(\mathcal{H}_k, \boldsymbol{X}_{k-1}) = \mathcal{H}_k^T \boldsymbol{X}_{k-1} \mathcal{H}_k$.
case 'ReEig layer'
19: $\boldsymbol{X}_k = f_r^{(k)}(\boldsymbol{X}_{k-1}) = \boldsymbol{U}\max(\epsilon \boldsymbol{I}, \boldsymbol{\Sigma})\boldsymbol{U}^T$.
case 'LogEig layer'
20: $\boldsymbol{X}_k = f_l^{(k)}(\boldsymbol{X}_{k-1}) = \boldsymbol{U}\log(\boldsymbol{\Sigma})\boldsymbol{U}^T$.
case 'Cross-Entropy layer'
21: $\ell = -\sum_{i=1}^{N}\sum_{\alpha=1}^{c}\tau(l_i, \alpha) \times \log(P(\alpha|\boldsymbol{X}_i))$.

Backward Pass:
case 'RBR layer'
22: Using Eqs. (25-27) to update $\mathcal{W}_k$.
case 'BiMap layer'
23: Updating $\mathcal{W}_k$ using Eq. (23) and Eqs. (25-26).
24: Swap($\mathcal{W}_k^{RBR\uparrow}, \mathcal{W}_k^{BiMap\uparrow}$) for the associated RBR and BiMap layers.
case 'Pooling layer'
25: $\mathcal{H}_k^{\uparrow} \leftarrow \text{GCG}(\mathcal{H}_k, -\lambda \times \nabla_{\mathcal{H}_k})$.
26: Further update $\mathcal{H}_k^{\uparrow}$ via the rules of Eq. (23) and Eqs. (25-26).
case 'ReEig layer'
27: Computing $\frac{\partial L^{(k)}}{\partial \boldsymbol{X}_{k-1}}$ using Eqs. (15–17) of Huang and Van Gool (2017).
case 'LogEig layer'
28: Computing $\frac{\partial L^{(k)}}{\partial \boldsymbol{X}_{k-1}}$ using Eq. (15) and Eqs. (19–20) of Huang and Van Gool (2017).
case 'Cross-Entropy layer'
29: $\frac{\partial L^{(K)}}{\partial \boldsymbol{X}_{K-1}} = P(\alpha|\boldsymbol{X}_i) - 1$.

30: ***end***
31: ***end***
32: ***end***
Output: $\mathcal{W}$ and $\mathcal{H}$.

trade-off parameter α were determined by cross-validation. In LEML (Huang, Wang, Shan, Li et al., 2015), the values of η and ζ were selected in the scopes of [0.1, 1, 10] and [0.1 : 0.1 : 1], respectively. The values of γ and ζ in HERML (Huang et al., 2015a) were chosen in the sets [0.001, 0.01, 0.1, 1, 10, 100, 1000] and [0.1 : 0.1 : 1], respectively. For SPDML (Harandi et al., 2018),



(a): AFEW dataset
(b): MDSD dataset
(c): CG dataset
(d): FPHA dataset

Fig. 3. Some sample images of the used four benchmarking datasets.

the two graph parameters v_w and v_b were determined by cross-validation. In GEMKML (Wang et al., 2020), the settings of the subspace dimension q and the graph parameters R_w and R_b are consistent with those in Wang et al. (2020) on the AFEW, MDSD, and FPHA datasets, while their values are determined by cross-validation on the CG dataset. For SPDNet (Huang & Van Gool, 2017), the sizes of the transformation matrices and the values of other parameters, *e.g.*, learning rate and batch size, were determined by cross-validation on the MDSD, CG, and FPHA datasets, while the settings on the AFEW dataset are the same as those of Huang and Van Gool (2017). In SymNet (Wang, Wu, Kittler, 2021), the sizes of the filters and the values of the thresholds ϵ and η are configured as recommended in Wang, Wu, Kittler (2021).

5.3. Datasets description and settings

AFEW dataset: This dataset involves 1345 video sequences of 7 different natural facial expressions collected from movies. Fig. 3(a) illustrates some examples of this dataset. In the experiments, these training sequences are firstly split into 1746 small clips in line with the standard protocols of Huang and Van Gool (2017). Then, the recognition results are reported on the validation set, as the groundtruth of the test set is unavailable. **MDSD dataset**: This dataset is comprised of 13 different categories of dynamic scenes, each of which containing 10 video sequences collected in an unconstrained scenario. Some scene images of this dataset are shown in Fig. 3(b). **CG dataset**: This dataset is constituted by 9 hand gestures, each of which consisting of 100 video sequences. Fig. 3(c) presents some examples of this dataset.

To coincide with the previous works (Huang, Wang, Shan, Li et al., 2015; Wang et al., 2020; Wang, Wu, Kittler, 2021) in data generation, each subject of the MDSD dataset has 7 randomly chosen video sequences for gallery and the rest 3 for probes, respectively. For the CG dataset, its training set is comprised of 20 randomly picked objects per class, whilst its test set is made up of the remaining 80 ones. In addition, all the images of the

Table 4
Accuracy comparison (%) on the AFEW, MDSD, and CG datasets.

Methods	Source	AFEW	MDSD	CG
PML (Huang et al., 2015b)	CVPR'15	28.98	29.67	82.64
LEML (Huang, Wang, Shan, Li et al., 2015)	ICML'15	25.13	29.30	75.14
LMKML (Lu et al., 2013)	ICCV'13	N/A	32.37	86.25
SPDML (Harandi et al., 2018)	TPAMI'18	24.55	29.81	82.62
GEMKML (Wang et al., 2020)	TMM'20	35.71	35.89	87.64
DeepO2P (Ionescu et al., 2015)	ICCV'15	28.54	N/A	N/A
GrNet (Huang et al., 2018)	AAAI'18	34.23	31.25	85.69
SymNet (Wang, Wu, Kittler, 2021)	TNNLS'21	32.70	35.58	88.78
SPDNet (Huang & Van Gool, 2017)	AAAI'17	34.23	32.05	86.67
SPDNetR		35.31	38.46	88.75
SPDNetP		35.58	34.36	88.33
SPDNetRP		**35.85**	**39.49**	**89.17**

MDSD and AFEW datasets are resized into 20×20 grayscale ones, while those of the CG dataset are resized into 10×10 intensity images. As a consequence, a series of 400×400 and 100×100 covariance matrices can be computed for image set representation.

FPHA dataset: This dataset includes 1175 hand action videos, belonging to 45 different categories. Fig. 3(d) displays some examples of this dataset. For the evaluation, in addition to five representative Riemannian learning algorithms (Harandi et al., 2018; Huang & Van Gool, 2017; Huang, Wang, Shan, Li et al., 2015; Lu et al., 2013; Wang, Wu, Kittler, 2021), the proposed method is also compared with some state-of-the-art hand action recognition models, such as the Two Streams (Feichtenhofer, Pinz, & Zisserman, 2016), Novel View (Rahmani & Mian, 2016), Lie Group (Vemulapalli, Arrate, & Chellappa, 2014), HRNN (Du, Wang, & Wang, 2015), LSTM (Garcia-Hernando et al., 2018), JOULE (Hu, Zheng, Lai, & Zhang, 2015), Gram Matrix (Zhang, Wang, Gou, Sznaier, & Camps, 2016), TF (Garcia-Hernando & Kim, 2017), TCN (Kim & Reiter, 2017), ST-GCN (Yan, Xiong, & Lin, 2018), H+O (Tekin, Bogo, & Pollefeys, 2019), and TTN (Lohit, Wang, & Turaga, 2019). Furthermore, on the basis of the criterion of Garcia-Hernando et al. (2018), each action frame is firstly transformed into a 63-dimensional feature vector using the provided 3D coordinates of 21 hand joints, such that a 63×63 covariance matrix can be computed for the modeling of an action sequence. Then, the 1:1 setting, *i.e.*, 600 sequences are designated for training and the rest 575 for testing, is applied to train the designed network.

5.4. Results

In general, SPDNetRP uses both the RBR and the Riemannian pooling modules. Similarly, SPDNetR and SPDNetP only contain the RBR and Riemannian pooling layers, respectively. By running the publicly available source codes of the different methods (except for LMKML and DeepO2P) provided by the original authors, the classification results reported in Table 4 can be obtained. Since the source code of LMKML is not available yet, a careful reimplementation is carried out in line with Lu et al. (2013) in this paper. Due to LMKML is very time-consuming in computing the scatter matrices, it has not been tested on the relatively large-scale AFEW dataset. For DeepO2P, its classification result on the AFEW dataset is provided by Huang and Van Gool (2017). As can be evidently seen from Tables 4 and 5, the proposed SPDNetRP is the best performer on all the four used datasets. Additionally, although the classification performance of SPDNetR and SPDNetP is inferior to that of SPDNetRP, they outperform almost all the competitors in the experimental scenario. These experimental results not only certify the effectiveness of the designed Riemannian batch regularization and pooling algorithms in improving the image set classification performance of the baseline SPDNet, but also demonstrate that these two modules play the complementary role in promoting the SPD manifold neural networks to generate more discriminative feature manifolds with lower dimensionality. In what follows, some further investigations about the characteristics of the two algorithms are conducted.

5.5. Ablation study for the Riemannian batch regularization (RBR) module

In this part, several experiments are made on the AFEW, MDSD, CG, and FPHA datasets to investigate the impact of the number of RBR layers on the classification performance of the proposed approach. The experimental results of different cases are tabulated in Table 6. In this table, SPDNetRP-*x*RBR denotes that the RBR module is added before the *x*th BiMap layer. For example, $x = 23$ means that the RBR module is appended before the 2nd and 3rd BiMap layers, respectively. As can be computed from Table 6 that the standard deviations of the classification scores achieved by all the studied cases on the AFEW, MDSD, CG, and FPHA datasets are 0.36, 1.78, 0.58, and 0.51, respectively. This demonstrates that the proposed approach is less sensitive to the number of the RBR layers on the AFEW, CG, and FPHA datasets. The underlying reason for the relatively large fluctuations of these candidates in classification accuracy on the MDSD dataset is the limited training samples. Another interesting observation in Table 6 is that the classification performance of the proposed approach is not proportional to the number of RBR modules on the four used datasets. The main reason is that since the AFEW dataset is made up of natural facial expressions of large intra-class diversity and inter-class ambiguity, the high-level features with rich semantic information are more useful for decision making. In contrast, as the MDSD, CG, and FPHA datasets involve a wide range of inter-class appearance variations, the low-level features are more efficient for classification. In this scenario, more RBR layers may lead to the encoding and learning of some interference information, thus weakening the model capacity.

5.6. Ablation study for the Riemannian pooling methods

Although Section 3.4 theoretically explains the feasibility of the M-LPE method for Riemannian pooling, it lacks verification in the experimental scenarios. Besides, the size of the transformation matrix of the BiMap layer and the position of the Riemannian pooling module embedded in the proposed architecture also have influence on the discrimination of the generated features. Accordingly, this article conducts classification experiments on the AFEW, MDSD, CG, and FPHA datasets to compare the effectiveness of the two Riemannian pooling methods in feature selection (*the M-LPE method is inapplicable to the FPHA dataset because of the sizes of the feature maps do not support the given pooling window size*). The classification accuracy of the different architectures on the four used datasets is listed in Table 7, where $d_{k-1} \times d_k$, *e.g.*, 100×90, denotes the size of the transformation matrix $\boldsymbol{\mathcal{W}}_k$ of the *k*th BiMap layer, the FML layer at different positions represents that the size of the projection matrix $\boldsymbol{\mathcal{H}}_k$ used for low-rank projection is different, and the M-LPE layer at different positions indicates that the size of the tangent space $\mathcal{T}_{\mathcal{I}}\mathcal{S}^d_{++}$ used for mean pooling is different. From Table 7, it is evident that the classification scores of the architectures embedded with the FML module are significantly superior to that of the models embedded with the M-LPE module. The main reason is that the implicit manifold to manifold mapping g of the M-LPE algorithm is derived by a fixed function ρ (mean pooling function in this paper) in the SPD manifold tangent space, while that of the FML method is directly learned on the SPD manifold. Since the pooling

Table 5
Accuracy comparison (%) on the FPHA dataset.

Methods	Source	Color	Depth	Pose	Acc.
JOULE-pose (Hu et al., 2015)	CVPR'15	✗	✗	✓	74.60
JOULE-all (Hu et al., 2015)	CVPR'15	✓	✓	✓	78.78
Two streams-color (Feichtenhofer et al., 2016)	CVPR'16	✓	✗	✗	61.56
Two streams-flow (Feichtenhofer et al., 2016)	CVPR'16	✓	✗	✗	69.91
Two streams-all (Feichtenhofer et al., 2016)	CVPR'16	✓	✗	✗	75.30
Novel view (Rahmani & Mian, 2016)	CVPR'16	✗	✓	✗	69.21
HBRNN (Du et al., 2015)	CVPR'15	✗	✗	✓	77.40
TF (Garcia-Hernando & Kim, 2017)	CVPR'17	✗	✗	✓	80.69
LSTM (Garcia-Hernando et al., 2018)	CVPR'18	✗	✗	✓	80.14
TCN (Kim & Reiter, 2017)	CVPRW'17	✗	✗	✓	78.57
ST-GCN (Yan et al., 2018)	AAAI'18	✗	✗	✓	81.30
H+O (Tekin et al., 2019)	CVPR'19	✓	✗	✗	82.43
TTN (Lohit et al., 2019)	CVPR'19	✗	✗	✓	83.10
Gram matrix (Zhang et al., 2016)	CVPR'16	✗	✗	✓	85.39
Lie group (Vemulapalli et al., 2014)	CVPR'14	✗	✗	✓	82.69
LMKML (Lu et al., 2013)	ICCV'13	✗	✗	✓	76.17
LEML (Huang, Wang, Shan, Li et al., 2015)	ICML'15	✗	✗	✓	79.48
SPDML (Harandi et al., 2018)	TPAMI'18	✗	✗	✓	81.74
GrNet (Huang et al., 2018)	AAAI'18	✗	✗	✓	77.57
SymNet (Wang, Wu, Kittler, 2021)	TNNLS'21	✗	✗	✓	82.96
SPDNet (Huang & Van Gool, 2017)	AAAI'17	✗	✗	✓	86.26
SPDNetR		✗	✗	✓	87.57
SPDNetP		✗	✗	✓	88.35
SPDNetRP		✗	✗	✓	**88.70**

Table 6
Accuracy comparison (%) on the AFEW, MDSD, CG, and FPHA datasets.

Methods	AFEW	MDSD	CG	FPHA
SPDNetRP-1RBR	35.58	38.14	88.19	87.30
SPDNetRP-2RBR	35.04	35.26	87.50	87.48
SPDNetRP-3RBR	35.31	35.26	88.19	87.33
SPDNetRP-12RBR	35.58	**39.49**	87.50	**88.70**
SPDNetRP-13RBR	34.77	38.14	**89.17**	87.33
SPDNetRP-23RBR	**35.85**	35.26	88.47	87.33
SPDNetRP-123RBR	35.31	38.14	88.06	87.39

Table 7
Accuracy comparison (%) on the AFEW, MDSD, and CG datasets.

Datasets	Architectures	Acc.
CG	$100 \times 90 \rightarrow 90 \times 60 \rightarrow$ **FML** $\rightarrow 40 \times 20$	87.92
	$100 \times 90 \rightarrow$ **FML** $\rightarrow 60 \times 40 \rightarrow 40 \times 20$	**89.17**
	$100 \times 80 \rightarrow$ **FML** $\rightarrow 40 \times 30 \rightarrow 30 \times 20$	87.08
	$100 \times 80 \rightarrow$ **M−LPE** $\rightarrow 40 \times 30 \rightarrow 30 \times 20$	77.08
	$100 \times 80 \rightarrow 80 \times 60 \rightarrow$ **M−LPE** $\rightarrow 30 \times 20$	75.28
MDSD	$400 \times 300 \rightarrow 300 \times 200 \rightarrow$ **FML** $\rightarrow 100 \times 50$	36.92
	$400 \times 300 \rightarrow$ **FML** $\rightarrow 200 \times 100 \rightarrow 100 \times 50$	**39.49**
	$400 \times 300 \rightarrow$ **FML** $\rightarrow 150 \times 100 \rightarrow 100 \times 50$	28.21
	$400 \times 300 \rightarrow$ **M−LPE** $\rightarrow 150 \times 100 \rightarrow 100 \times 50$	34.61
	$400 \times 300 \rightarrow 300 \times 200 \rightarrow$ **M−LPE** $\rightarrow 100 \times 50$	33.33
AFEW	$400 \times 200 \rightarrow 200 \times 100 \rightarrow$ **FML** $\rightarrow 80 \times 50$	**35.85**
	$400 \times 200 \rightarrow$ **FML** $\rightarrow 150 \times 100 \rightarrow 100 \times 50$	34.23
	$400 \times 300 \rightarrow$ **FML** $\rightarrow 200 \times 100 \rightarrow 100 \times 50$	34.77
	$400 \times 200 \rightarrow$ **M−LPE** $\rightarrow 100 \times 80 \rightarrow 80 \times 50$	26.15
	$400 \times 200 \rightarrow 200 \times 160 \rightarrow$ **M−LPE** $\rightarrow 80 \times 50$	25.07
FPHA	$63 \times 53 \rightarrow 53 \times 43 \rightarrow$ **FML** $\rightarrow 38 \times 33$	88.00
	$63 \times 58 \rightarrow$ **FML** $\rightarrow 53 \times 43 \rightarrow 43 \times 33$	**88.70**
	$63 \times 53 \rightarrow$ **FML** $\rightarrow 48 \times 43 \rightarrow 43 \times 33$	88.35

tactic of LPE can be defined as a learnable function in the domain of a function family, the designed LPE algorithm can be treated as a functional optimization problem. Therefore, how to determine a proper function family for finding an optimal ρ (M-LPE is a special case of LPE, and ρ of M-LPE may not be optimal) that can infer a better immersion (or submersion) g is a key problem for the machine learning community. From Table 7, it also can be noted that the FML module with a larger projection matrix size attached to the end of the first BiMap layer shows better performance than that of other cases on the CG, MDSD, and FPHA datasets, while this phenomenon is reversed on the AFEW dataset. Based on these experimental observations, the proper sizes of the connection weights of the BiMap layers are set to 400×200, 200×100, and 80×50 on the AFEW dataset, 400×300, 200×100, and 100×50 on the MDSD dataset, 100×90, 60×40, and 40×20 on the CG dataset, and 63×58, 53×43, and 43×33 on the FPHA dataset, respectively. In addition, the suitable sizes of the projection matrices are respectively configured as 100×80 on the AFEW dataset, 300×200 on the MDSD dataset, 90×60 on the CG dataset, and 58×53 on the FPHA dataset. All in all, these experimental observations demonstrate that the unsupervised metric learning-guided pooling scheme conveyed by the FML method is eligible to capture pivotal data variations for improved classification.

5.7. Time comparison:

In this part, some experiments are performed on all the datasets involved in this article to analyze the influence of the Riemannian batch regularization and pooling algorithms on the computation time of the baseline network. Table 8 shows that the training time of SPDNetRP per epoch is respectively 0.79 s, 1.63 s, 27.85 s, and 2.46 s slower than that of SPDNet on the CG, FPHA, AFEW, and MDSD datasets. It also can be inferred from Table 8 that the computational burden brought by the RBR operation is higher than that of the Riemannian pooling operation. The main reason is that the RBR algorithm needs to pay a plenty of time to traverse all the possible triplets and compute the corresponding intra- and inter-manifold Riemannian distances within these triplets. Another interesting observation from Table 8 is that on the four used datasets, the test time (classification of one image set) of the baseline network is not affected by the two algorithms proposed. The underlying reasons lie in two aspects: (1) the RBR module is not used in the test stage; (2) in the test phase, the low-rank projection of the Riemannian pooling layer is basically not time-consuming.

Based on the above experimental results, further experiments are made to compare the computational efficiency of the proposed approach with some representative SPD manifold learning-based image set classification methods, choosing the relatively large-scale AFEW dataset as an example. The results are listed in Table 9, where the test time is defined by measuring the time

Table 8
Time (s/epoch) comparison on the CG and FPHA datasets.

Methods	CG		FPHA		AFEW		MDSD	
	Train	Test	Train	Test	Train	Test	Train	Test
SPDNet (Huang & Van Gool, 2017)	**0.45**	0.008	**1.45**	0.0013	27.38	0.011	1.72	0.012
SPDNetR	1.18	0.008	3.02	0.0013	50.06	0.011	3.57	0.012
SPDNetRP	1.24	0.008	3.08	0.0013	55.23	0.011	4.18	0.012

Table 9
Time comparison of the different methods on the AFEW dataset.

Time	PML (Huang et al., 2015b)	LEML (Huang, Wang, Shan, Li et al., 2015)	SPDML (Harandi et al., 2018)	GEMKML (Wang et al., 2020)	SymNet (Wang, Wu, Kittler, 2021)	SPDNet (Huang & Van Gool, 2017)	SPDNetRP
Train (h)	5.33	0.39	7.68	1.69	0.025	1.58	3.15
Test (s)	0.008	0.57	0.12	0.023	0.055	0.011	0.011

Table 10
Comparison (%) on the MDSD dataset under different validation metrics.

Methods	Precision_{mi}	Recall_{mi}	Specificity_{mi}	F1-Score
PML (Huang et al., 2015b)	29.67	29.67	94.14	27.58
LEML (Huang, Wang, Shan, Li et al., 2015)	29.30	29.30	94.11	27.29
LMKML (Lu et al., 2013)	32.37	32.37	94.36	31.67
GEMKML (Wang et al., 2020)	35.89	35.89	94.66	36.66
SPDML (Harandi et al., 2018)	30.04	30.04	94.17	30.96
SPDNet (Huang & Van Gool, 2017)	32.05	32.05	94.34	29.01
SymNet (Wang, Wu, Kittler, 2021)	35.58	35.58	94.63	35.61
SPDNetR	38.46	38.46	94.87	37.05
SPDNetP	34.36	34.36	94.53	32.60
SPDNetRP	**39.49**	**39.49**	**94.96**	**38.63**

required to assign one image set into its category. From this table, it is evident that the training time of PML and SPDML is significantly higher than that of other competitors. The main reason is that PML needs to pay $\mathcal{O}(N^2d^3)$ to compute the high-dimensional intra- and inter-class scatter matrices, and the computational burden will change exponentially as the number of training samples increases. In addition to $\mathcal{O}(N^2d^3)$ required to construct the intra- and inter-class similarity graphs, SMDML also involves multiple matrix inversion, so its training time is longer. It also can be noted from Table 9 that the test time of the proposed SPDNetRP is 0.003 s slower than that of PML, mainly because SPDNetRP needs to apply multi-stage feature transformation and Log-Euclidean embedding to a test sample before classification. In contrast, PML is a shallow learning algorithm, which just contains a low-rank projection before classification. In spite of this, Table 9 demonstrates that the suggested approach has a competitive computational efficiency, especially compared with some traditional shallow learning methods.

5.8. Comparison under other validation metrics

The aforementioned experimental results justify the validity of the proposed Riemannian operation modules in improving the capacity of the SPD manifold neural network for image set classification. In order to further evaluate the designed model, some representative Riemannian learning methods are selected for comparison on the MDSD dataset using other validation metrics, such as Precision micro averaging (Precision_{mi}), Recall micro averaging (Recall_{mi}), Specificity micro averaging (Specificity_{mi}), and F1-Sore. The final classification scores of the different algorithms on this dataset are tabulated in Table 10. From this table, it is evident that the performance of SPDNetR and SPDNetP are better than that of SPDNet under the four validation criteria. Besides, in this experiment, the proposed SPDNetRP is still the best performer. These experimental findings again certify the significance of the proposed two innovations in guiding the SPD manifold neural network to extract more discriminative invariant representations from a wide range of data variations.

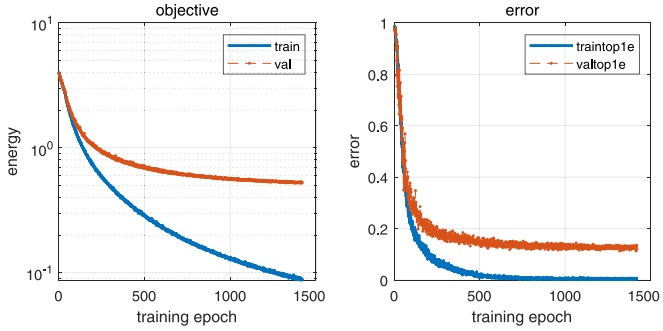


Fig. 4. Convergence behavior on the FPHA dataset.

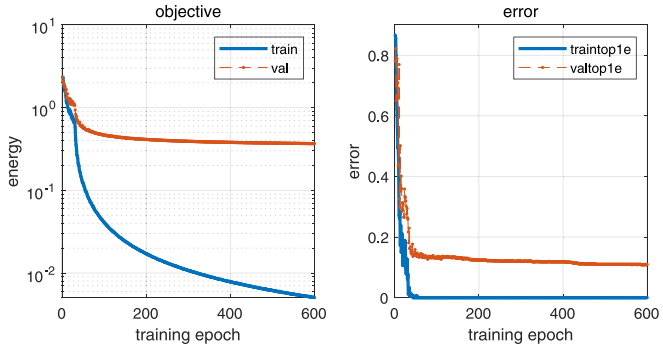


Fig. 5. Convergence behavior on the CG dataset.

5.9. Convergence analysis

In this part, several experiments are carried out to study the convergence behavior of the proposed SPDNetRP, choosing the FPHA and CG datasets as two examples. The cost value and the recognition error of the suggested approach versus the number of training epochs are displayed in Figs. 4 and 5, respectively.

As can be apparently seen from these two figures, the proposed model can converge to a desirable solution in less than 1500 and 600 epochs on these two datasets, manifesting that it has a good convergence behavior. In the experiments, the maximum of training epochs are respectively set to 1415 and 600 for the FPHA and CG datasets, and that for the MDSD and AFEW datasets are configured as 300 and 205, respectively.

5.10. Discussions

According to the experimental results given in Table 4, it is clear that the classification scores of the proposed SPDNetR and SPDNetP are higher than those of the baseline model, *i.e.*, SPDNet (Huang & Van Gool, 2017), on the AFEW, MDSD, and CG datasets. The fundamental reason is that SPDNet only utilizes a single cross-entropy loss to supervise the whole network, while neglecting the peculiarities of the data distribution. In consequence, the variability information conveyed by the inputs cannot be explicitly encoded and analyzed during training, suppressing the discriminability of the learned deep representations. In contrast, the core mechanism of the suggested two Riemannian operation modules is metric learning, which is qualified to capture and learn pivotal intra- and inter-class data distributions, thus improving the discriminability of the generated features. As a consequence, the combined use of these two modules can enable the softmax classifier to learn a hypersphere with a more reasonable probability distribution for different categories, thus making the designed SPDNetRP achieve the best classification results on the three benchmarking datasets. From Table 4, it is also evident that the classification performance of SPDNetP is inferior to that of SymNet (Wang, Wu, Kittler, 2021) and GEMKML (Wang et al., 2020) on the MDSD dataset. The reasons lie in three aspects: (1) MDSD is a small-scale dataset, restraining the baseline SPDNet from fully mining the fine-grained geometric semantic information of the data; (2) SymNet and GEMKML perform feature extraction based on the shallow optimization algorithm-trained lightweight Riemannian backbones, being able to capture pivotal feature variations, especially with limited data; (3) the Riemannian pooling layer conducts metric learning-based dimensionality reduction in an unsupervised scenario, which may fail to learn efficient data distributions compared to the supervised ones studied in SymNet and GEMKML. As can be clearly seen from Table 5, the Riemannian learning-based visual classification methods, *e.g.*, Lie Group (Vemulapalli et al., 2014), LEML (Huang, Wang, Shan, Li et al., 2015), SPDML (Harandi et al., 2018), GrNet (Huang et al., 2018), SymNet (Wang, Wu, Kittler, 2021), and SPDNet (Huang & Van Gool, 2017), achieve competitive recognition performance on the FPHA dataset. Since the underlying space in which the 3D skeleton data resides is a low-dimensional manifold, characterizing such data sequences with Riemannian geometry is capable of reflecting the global spatiotemporal fluctuations of action sequences more faithfully. From Table 5, it is also evident that the recognition scores of SPDNetR and SPDNetP are superior to that of SPDNet on this dataset, further confirming the effectiveness of the two metric learning-based Riemannian operations in mitigating the within-class dispersion and between-class ambiguity of representations. Furthermore, on the FPHA dataset, the classification performance of SPDNetRP is better than that of all the competitors. This again demonstrates that the proposed RBR and Riemannian pooling modules are complementary to each other in assisting the network to probe a more effective probability distribution w.r.t the original visual data for improved classification.

Although the superiority of the proposed method has been justified by a series of experiments made on the four used datasets, Table 8 shows that its training time per epoch is more than twice than that of the baseline SPDNet. As mentioned above, the main computational burden comes from the RBR module, as it needs to step through all the possible triplets within a batch of samples and compute the corresponding intra- and inter-manifold distances. Hence, in the future, more efforts will be made to refurbish the data sampling mechanism of the designed metric learning term or introduce new metric learning frameworks with comparatively higher computational efficiency for the RBR layer.

It is widely acknowledged that deep learning methods have achieved remarkable progress in a variety of computer vision and medical image analysis tasks, evidently becoming the state-of-the-art techniques. The superiority of deep learning is to learn powerful feature representations and then map the high dimensional data into an array of outputs. However, these embedding mappings are often taken blindly and assumed to be accurate, which is not always the case. Basically, almost all the machine learning (ML) and deep learning (DL) methods "*do not know*" what exactly "*they know*". Based on this reason, the learning systems may certainly classify one sample with a high predictive probability (the softmax output) which they have never seen before. Therefore, a blind trust in the results obtained by traditional ML and DL methods will lead to the loss of some important features and information (Abdar et al., 2021; van Amersfoort, Smith, Jesson, Key, & Gal, 2021). UQ-based methods play an important role in alleviating the impact of uncertainties in the processes of optimization and decision making, which have been applied to solve many real-world problems in science and engineering domains (Abdar et al., 2021). There are two main types of uncertainty one can model, *i.e.*, aleatoric and epistemic uncertainties (Abdar, Pourpanah et al., 2021; Kendall & Gal, 2017). The *aleatoric* (aka data) uncertainty captures noise inherent in the observations, which could be for example sensor noise or motion noise. Consequently, this type of uncertainty cannot be reduced even if more data were to be collected. In contrast, *epistemic* uncertainty refers to uncertainty caused by a lack of knowledge (about the best model). Different from the data uncertainty caused by randomness, the epistemic uncertainty can in principle be reduced given enough data, and is also referred to as model uncertainty.

UQ is a key factor for an accurate application of ML and DL methods. An estimation of uncertainties can increase the confidence of results provided by these methods. To the best of our knowledge, bayesian approximation and ensemble learning techniques are two types of uncertainty quantification methods being widely-used. In this respect, researchers have suggested different UQ methods and their performance has been evaluated in a variety of practical applications, *e.g.*, computer vision (Abdar, Pourpanah et al., 2021; Kendall & Gal, 2017) and medical image analysis (Abdar et al., 2021, 2021; Abdar, Samami et al., 2021), etc. Placing a prior distribution, *e.g.*, $\mathbf{W} \sim \mathcal{N}(0, I)$, on the neural network weights defines a Bayesian Neural Network (BNN). Actually, BNNs are easy to formulate, but difficult to perform inference in. The underlying reason is that the marginal probability $p(\mathbf{Y}|\mathbf{X})$, required to evaluate the posterior $p(\mathbf{W}|\mathbf{X}, \mathbf{Y})$ cannot be computed analytically. Existing evidences (Abdar, Samami et al., 2021; Gal & Ghahramani, 2016) show that optimizing neural networks by dropout (as a standard regularization operation) and L2-regularization can be equivalent to a type of variational inference in a Bayesian model. This inference is done by training a model with dropout before every weight layer, followed by also performing dropout at test stage to sample from the approximate posterior (stochastic forward passes, referred to as Monte Carlo dropout). As BNNs are robust to overfitting problems and can be trained on both small and large-scale datasets (Abdar, Pourpanah et al., 2021; Kendall & Gal, 2017), a larger number of Monte Carlo (MC) dropout-based methods have been proposed. To name a

few, Abdar, Fahami et al. (2021) incorporate an uncertainty-aware module using MC dropout into the proposed BinAry Residual feature Fusion (BARF) model for medical image classification. Besides, the authors jointly use MC and standard dropouts for the sake of optimizing the performance of the designed fusion model. Experimental results on four medical datasets have demonstrated its effectiveness with increased certainty in the obtained results. As a simple and scalable alternative to BNNs, the deep ensemble (DE) approach (Rahaman & Thiery, 2020) computes uncertainty by collecting various predictions realized by several separate deterministic models, whose parameters are initialized to random starting points, trained independently. It has the merits of easy to run, parallelizable, and can generate more precise estimations of uncertainty. Inspired by this, Abdar, Salari et al. (2021) propose a powerful deep learning-based feature fusion model called *UncertaintyFuseNet* for COVID-19 detection. Considering the extremely high sensitivity and variability of the COVID-19 virus, the authors inject an Ensemble Monte Carlo (EMC) dropout module into the suggested framework to perform UQ, with the purpose of enhancing the reliability of detection. Experimental results certify that the designed fusion model not only exhibits strong robustness to data noise, but also produces excellent detection results for the unknown data. Although deep neural networks often provide reasonable predictions for disease diagnosis, they typically focus on improving accuracy, regardless of the uncertainty in estimating the decision-making process. To improve the performance of computer-aided diagnostic systems, a novel hybrid dynamic bayesian deep learning architecture (Abdar, Samami et al., 2021) has been established to implement a three-phase uncertainty estimation for the prevention of overconfidence in skin cancer prediction. The designed model enables the use of different UQ techniques and different deep neural networks in distinct classification stage, so that the most appropriate elements can be selected at each phase and adjusting them according to the given dataset. To realize an accurate prediction model, the DE and EMC dropout methods are incorporated into the suggested framework using the three-way decision (TWD) theory, followed by a bayesian optimization technique to tune the hyperparameters of all the involved deep learning models. Extensive experiments made on two well-known skin cancer datasets demonstrate its effectiveness in medical image analysis. However, the methods of DE and MC dropout require multiple forward passes and are therefore computationally expensive. This will lead to problems when using these models in situations that require real-time decision making. Taking this into account, van Amersfoort et al. (2021) design a single forward pass uncertainty model named Deterministic Uncertainty Estimation (DUE) that constructs upon Deep Kernel Learning (DKL) to tackle its limitation in feature collapse. To be specific, the authors propose to constrain DKL's feature extractor to approximately preserve distances through a bi-Lipschitz constraint, leading to a feature space favorable to DKL. Experimental results justify that the designed DUE model outperforms the previous DKL and other singleforward pass uncertainty methods. There also exist other UQ studies related to the applications of traditional ML and DL methods, such as Markov chain Monte Carlo (MCMC), Bayesian Active Learning (BAL), and Variational autoencoders (VAEs). Due to space limitation, we do not review them in this article. For details, please kindly refer to Abdar, Pourpanah et al. (2021).

Although research on uncertainty quantification in Euclidean deep neural networks has achieved considerable success, there are almost no works to perform UQ in the context of Riemannian neural networks to the best of our knowledge. We think the main reasons come from two aspects: (1) the development of Riemannian neural networks is still at the early stage; (2) performing pattern analysis over the Riemannian manifolds requires specific mathematical computations, characterizing the non-Euclidian property of the involved data points. Since the existing SPD manifold neural networks are all single-channel (the ReEig and LogEig layers of SPDNets involve the SVD operation, which makes the GPU acceleration inapplicable (Brooks et al., 2019). The SPD manifold neural network with multiple channels is time-consuming due to the need for successive SVD computations), as an exploration, the MC dropout operation is applied only to the spatial domain of the proposed model to perform UQ. Besides, the MC dropout operation cannot be used in conjunction with the BiMap operation. The underlying reason is that the dropout operation will inevitably bring about a big distortion to the eigenvalue space of the SPD matrix, making some of the pivotal structural information of the input data be destroyed. Moreover, we have observed that the classification performance and convergence behavior of the proposed model are very poor under the above-mentioned case. As a consequence, on the MDSD dataset (as an example), the MC dropout layer only appears at the tail of the FC layer in the proposed model with a dropout rate of 0.1, and the MC sampling times is set to 10 at the test stage. The baseline SPDNet (Huang & Van Gool, 2017) also utilizes this experimental setting to carry out UQ in the experiments. The experimental results of the different competitors on the MDSD dataset are reported in Table 11, where '*i*MCD' denotes that the MC dropout operation is included in the SPD manifold neural network. From Table 11, it can be noted that the classification scores of SPDNetRP-*i*MCD are higher than those of SPDNet-*i*MCD on the MDSD dataset, again justifying the efficacy of the proposed approach in improving the image set classification performance of the baseline model. Nevertheless, the classification ability of SPDNetRP-*i*MCD and SPDNet-*i*MCD is inferior to that of the most of the methods in Table 10. This reveals that it is not trivial to prolongate the conventional UQ methods to the domain of Riemannian neural networks. In future, more studies will be made to explore how to generalize the paradigm of the existing UQ methods to the scenario of Riemannian deep learning from both theoretical and engineering practice perspectives to obtain a more trustworthy classification system.

6. Application to skeleton-based human action recognition with unmanned aerial vehicles

Thanks to the flexibility and capability of unmanned aerial vehicles (UAVs) in long range tracking and providing unique viewpoints, obvious resolution variations, and significant camera movements, the UAV-based human behavior understanding has becoming increasingly significant in many application domains, *e.g.*, pose estimation, person re-identification, and attribute recognition. Compared to the videos collected by common ground cameras, the video sequences captured by UAVs generally present a wide range of diversity, due to the fast motion and continuously changing attitudes and heights of the UAVs during flight. These factors make the UAV-based computer vision tasks rather challenging. In this section, the UAV-Human dataset (Li, Liu, Zhang, Ni, Wang, & Li, 2021) is applied to further evaluate the practicability of the proposed method in skeleton-based human pose recognition.

The UAV-Human dataset is a large-scale benchmark for UAV-based human behavior understanding. It contains a total of 67,428 annotated video sequences for action recognition, in which 22,476 annotated video clips belonging to 155 action categories are designated for the task of human pose estimation. As each action frame is labeled by 17 major body joints using 3D coordinates (illustrated in Fig. 6), the practice introduced in Chen et al. (2021) is exploited to convert each frame into a 51-dimensional feature vector. Besides, the number of frames selected for computation in each video is 305. In this way, each video sequence can be

Table 11
Comparison (%) on the MDSD dataset under different validation metrics.

Methods	Precision_{mi}	Recall_{mi}	Specificity_{mi}	F1-Score
SPDNet-*i*MCD (Huang & Van Gool, 2017)	25.64	25.64	93.80	25.33
SPDNetRP-*i*MCD	**30.77**	**30.77**	**94.23**	**30.99**

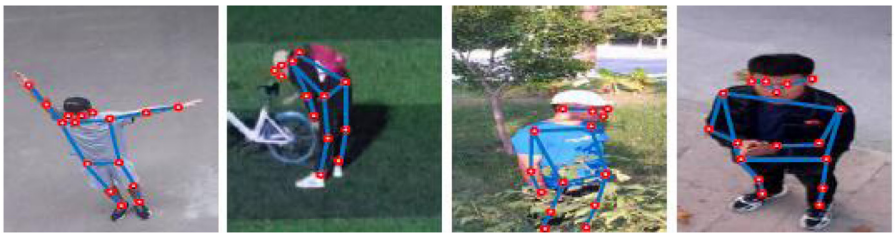


Fig. 6. Some instances of the UAV-Human dataset.

viewed as an image set matrix of size 51×305. Considering that these exist some actions that performed by two persons in this dataset, the PCA technique is applied to transform the 102-dimensional feature vectors into 51-dimensional ones by preserving 99% energy of the data. In this scenario, a number of 22,476 SPD matrices of size 51×51 can be computed for video representation. On the UAV-Human dataset, the seventy-thirty-ratio (STR) protocol is used to build the gallery and probes from the randomly picked 16,723 SPD matrices. In the experiments, the sizes of the transformation matrices of the BiMap layers in the suggested network are set to 51×43, 43×37, and 34×31, respectively. In addition, the values of the learning rate λ, batch size B, rectification threshold ϵ, trade-off parameter γ, and the size of the projection matrix of the Riemannian pooling layer of the designed model are respectively configured as 0.01, 40, 1e–5, 0.2, and 37×34.

In this part, the following representative Riemannian learning-based image set classification methods are chosen for comparison, including PML (Huang et al., 2015b), LEML (Huang, Wang, Shan, Li et al., 2015), SPDML (Harandi et al., 2018), HRGEML (Chen et al., 2021), GEMKML (Wang et al., 2020), GrNet (Huang et al., 2018), SPDNet (Huang & Van Gool, 2017), and SymNet (Wang, Wu, Kittler, 2021). Additionally, a neural architecture search algorithm named DARTS (Liu, Simonyan, & Yang, 2019) is also selected for better evaluation. The experimental results of the different methods on the UAV-Human dataset are listed in Table 12. Note that the classification scores of these competitors are obtained by running their open-source codes, and are tuned to the optimal currently. For a fair comparison, the logarithm maps of SPD matrices are treated as the input data of DARTS. From Table 12, we can draw the following five observations. Firstly, the classification accuracy of HRGEML is significantly higher than that of PML, LEML, and SPDML on this dataset. The fundamental reason is that HRGEML performs image set modeling from the multi-manifold perspective, which can capture richer (complementary) statistical information to facilitate the learning of a discriminative subspace compared to the methods based on a single geometric model. Secondly, the classification performance of SymNet and GEMKML is superior to that of PML, LEML, and SPDML, further demonstrating that the integration of metric learning and lightweight Riemannian neural networks can extract more discriminative features than the shallow learning algorithms. Since DARTS exploits the gradient-based optimization mechanism to dynamically search the optimal network structure, it is qualified to obtain good classification performance, and this has also been verified in Table 12. The effectiveness of DARTS inspires us to extend the paradigm of Euclidean neural architecture search to the context of Riemannian manifolds in the future. Another interesting finding from Table 12 is that the classification performance of GEMKML and SymNet is respectively inferior to that of GrNet and SPDNet, further illustrating the superiority of end-to-end Riemannian deep learning, especially on large-scale datasets. Finally, the classification scores of SPDNetR and SPDNetP are respectively 0.44% and 0.32% higher than that of the baseline SPDNet on this dataset. This again justifies the efficacy of the two metric learning-based Riemannian operation modules, *i.e.*, RBR and Riemannian pooling, in guiding the network to learn a more discriminative manifold-to-manifold transformation mapping. Moreover, the combined use of the two modules makes further improvement of SPD manifold neural networks in terms of the classification ability, again confirming their complementary role in alleviating the intra-class diversity and inter-class ambiguity of geometric representations.

7. Conclusion

In this paper, two algorithms of Riemannian batch regularization and Riemannian pooling are developed for the SPD manifold neural networks, realizing their application to image set classification. The Riemannian batch regularization module is used to supervise the network to yield compact and efficient feature manifolds by capturing, encoding, and processing the intra- and inter-class data variability information. The Riemannian pooling module focuses on dimensionality reduction in an unsupervised manner using geometric computations on SPD matrices, namely Riemannian barycenter, metric learning, and Riemannian optimization. Experimental comparisons and ablation studies on five benchmarking datasets confirm that: (1) the proposed SPDNetRP is an eligible candidate in promoting the performance of image set classification, even with limited data; (2) the two algorithms are complementary to each other in enhancing the discriminability of the baseline model.

However, it can be noted that the overall classification ability of the proposed method and the existing Riemannian learning approaches on the complicated AFEW and MDSD datasets is still at a relatively low level. One of the main reasons is that the structured information embodied in the input SPD matrices cannot adequately characterize the complex data variations in such datasets. Considering that the SPD matrices are directly computed from the raw image set data without containing some preprocessings for the images themselves, a prospective countermeasure is to jointly perform image feature learning and SPD matrix learning within a designed network for the sake of relieving the negative impact of the implicit data variability information on the image set classification performance. As the temporal information plays an important role in describing the dynamic changes of objects in video sequences, and the spatiotemporal fluctuations of data sequences represented by the SPD matrices are coarse-grained, another possible direction for future work is to integrate

Table 12
Accuracy comparison (%) on the UAV-Human dataset.

Methods	PML (Huang et al., 2015b)	LEML (Huang, Wang, Shan, Li et al., 2015)	SPDML (Harandi et al., 2018)	HRGEML (Chen et al., 2021)
Acc.	10.66	21.83	22.69	36.10
Methods	GEMKML (Wang et al., 2020)	GrNet (Huang et al., 2018)	DARTS (Liu et al., 2019)	SymNet (Wang, Wu, Kittler, 2021)
Acc.	34.67	35.23	36.13	35.89
Methods	SPDNet (Huang & Van Gool, 2017)	**SPDNetR**	**SPDNetP**	**SPDNetRP**
Acc.	41.27	41.71	41.59	**42.15**

a spatiotemporal SPD aggregation model into the designed architecture to conduct coarse-to-fine feature modeling. Although the experimental results mentioned above show that the proposed algorithm has a good convergence behavior, it is difficult for us to provide a theoretical proof about it currently. Therefore, in the future, theoretical studies and explorations on the convergence of Riemannian neural networks will be carried out to guide the design of more robust network architectures.

Actually, there exist some theoretically supported Riemannian computation paradigms that cannot be effectively applied to the practical scenarios, such as the M-LPE method studied in this paper. As analyzed above, the fundamental reason is that the surjective and continuously differentiable mean pooling function ρ is not learnable, thus cannot infer an optimal manifold to manifold submersion g. Hence, more efforts will be made on the Riemannian pooling functions in the future, unifying the deep Riemannian techniques theoretically and experimentally. Learning from the information theory (Lovrić, Min-Oo, & Ruh, 2000) that a d-dimensional Single Gaussian Model (SGM) determined as: $G(\boldsymbol{S}) = \mathcal{N}(\boldsymbol{S}|\widetilde{\boldsymbol{m}}, \widetilde{\boldsymbol{C}})$ ($\widetilde{\boldsymbol{m}}$ and $\widetilde{\boldsymbol{C}}$ respectively denote the mean and variance of a given image set $\boldsymbol{S}$) can be uniquely identified as a $(d+1)$-by-$(d+1)$ SPD matrix $\acute{\boldsymbol{X}}$ in the Gaussian embedded space $\mathcal{S}_{++}^{d+1}$. In addition to the aforementioned future works, the generalization of the paradigms of the SPD manifold neural networks to the domain of Gaussian embedded Riemannian manifolds will also be studied in the future.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (62020106012, U1836218, 61672265, 62106089 62006097), the 111 Project of Ministry of Education of China (B12018), the Natural Science Foundation of Jiangsu Province, China (BK20200593), the Postgraduate Research & Practice Innovation Program of Jiangsu Province, China (KYCX21-2006), the UK EPSRC EP/N00 7743/1, MURI/EPSRC/DSTL, UK EP/R018456/1 grants, and the National Key Research and Development Program of China (2017YFC1601800).

References

Abdar, M., Fahami, M. A., Chakrabarti, S., Khosravi, A., Pławiak, P., Acharya, U. R., et al. (2021). Barf: A new direct and cross-based binary residual feature fusion with uncertainty-aware module for medical image classification. *Information Sciences*, 353–378.

Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., et al. (2021). A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 243–297.

Abdar, M., Salari, S., Qahremani, S., Lam, H.-K., Karray, F., Hussain, S., et al. (2021). Uncertaintyfusenet: Robust uncertainty-aware hierarchical feature fusion with ensemble monte carlo dropout for covid-19 detection. arXiv preprint arXiv:2105.08590.

Abdar, M., Samami, M., Mahmoodabad, S. D., Doan, T., Mazoure, B., Hashemifesharaki, R., et al. (2021). Uncertainty quantification in skin cancer classification using three-way decision-based bayesian deep learning. *Computers in Biology and Medicine*, Article 104418.

Absil, P.-A., Mahony, R., & Sepulchre, R. (2009). *Optimization Algorithms on Matrix Manifolds*. Princeton University Press.

van Amersfoort, J., Smith, L., Jesson, A., Key, O., & Gal, Y. (2021). On feature collapse and deep kernel learning for single forward pass uncertainty. arXiv preprint arXiv:2102.11409.

Arsigny, V., Fillard, P., Pennec, X., & Ayache, N. (2007). Geometric means in a novel vector space structure on symmetric positive-definite matrices. *SIAM Journal of Mathematical Analysis*, 328–347.

Baudat, G., & Anouar, F. (2000). Generalized discriminant analysis using a kernel approach. *Neural Computation*, 2385–2404.

Boumal, N., Mishra, B., Absil, P.-A., & Sepulchre, R. (2014). Manopt, a matlab toolbox for optimization on manifolds. *Journal of Machine Learning Research*, 1455–1459.

Brooks, D., Schwander, O., Barbaresco, F., Schneider, J.-Y., & Cord, M. (2019). Riemannian batch normalization for spd neural networks. arXiv preprint arXiv:1909.02414.

Chen, Z., Xu, T., Wu, X.-J., Wang, R., & Kittler, J. (2021). Hybrid riemannian graph-embedding metric learning for image set classification. *IEEE Transactions on Big Data*, 1.

Cheng, H., Yap, K.-H., & Wen, B. (2021). Reconciliation of statistical and spatial sparsity for robust image and image-set classification. arXiv preprint arXiv:2106.00256.

Cheng, G., Zhou, P., & Han, J. (2017). Duplex metric learning for image set classification. *IEEE Transactions on Image Processing*, 281–292.

Dhall, A., Goecke, R., Joshi, J., Sikka, K., & Gedeon, T. (2014). Emotion recognition in the wild challenge 2014: Baseline, data and protocol. In *ICMI* (pp. 461–466).

Du, Y., Wang, W., & Wang, L. (2015). Hierarchical recurrent neural network for skeleton based action recognition. In *CVPR* (pp. 1110–1118).

Edelman, A., Arias, T. A., & Smith, S. T. (1998). The geometry of algorithms with orthogonality constraints. *SIAM Journal of Mathematical Analysis*, 303–353.

Feichtenhofer, C., Pinz, A., & Zisserman, A. (2016). Convolutional two-stream network fusion for video action recognition. In *CVPR* (pp. 1933–1941).

Gal, Y., & Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning* (pp. 1050–1059).

Gao, Z., Wu, Y., Harandi, M., & Jia, Y. (2019). A robust distance measure for similarity-based classification on the spd manifold. *IEEE Transactions on Neural Networks and Learning Systems*, 1–15.

Garcia-Hernando, G., & Kim, T.-K. (2017). Transition forests: Learning discriminative temporal transitions for action recognition and detection. In *CVPR* (pp. 432–440).

Garcia-Hernando, G., Yuan, S., Baek, S., & Kim, T.-K. (2018). First-person hand action benchmark with rgb-d videos and 3d hand pose annotations. In *CVPR* (pp. 409–419).

Harandi, M., & Salzmann, M. (2015). Riemannian coding and dictionary learning: Kernels to the rescue. In *CVPR* (pp. 3926–3935).

Harandi, M., Salzmann, M., & Hartley, R. (2017). Joint dimensionality reduction and metric learning: A geometric take. In *ICML* (pp. 1404–1413).

Harandi, M., Salzmann, M., & Hartley, R. (2018). Dimensionality reduction on spd manifolds: The emergence of geometry-aware methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 48–62.

Harandi, M., Sanderson, C., Hartley, R., & Lovell, B. C. (2012). Sparse coding and dictionary learning for symmetric positive definite matrices: A kernel approach. In *ECCV* (pp. 216–229).

Hu, J.-F., Zheng, W.-S., Lai, J., & Zhang, J. (2015). Jointly learning heterogeneous features for rgb-d activity recognition. In *CVPR* (pp. 5344–5352).

Huang, Z., & Van Gool, L. (2017). A riemannian network for spd matrix learning. In *AAAI* (pp. 2036–2042).

Huang, Z., Wang, R., Shan, S., & Chen, X. (2015a). Face recognition on large-scale video in the wild with hybrid euclidean-and-riemannian metric learning. *Pattern Recognition*, 3113–3124.

Huang, Z., Wang, R., Shan, S., & Chen, X. (2015b). Projection metric learning on grassmann manifold with application to video based face recognition. In *CVPR* (pp. 140–149).

Huang, Z., Wang, R., Shan, S., Li, X., & Chen, X. (2015). Log-euclidean metric learning on symmetric positive definite manifold with application to image set classification. In *ICML* (pp. 720–729).
Huang, Z., Wu, J., & Van Gool, L. (2018). Building deep networks on grassmann manifolds. In *AAAI* (pp. 1137–1145).
Ionescu, C., Vantzos, O., & Sminchisescu, C. (2015). Training deep networks with structured layers by matrix backpropagation. arXiv preprint arXiv:1509.07838.
Kendall, A., & Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision? arXiv preprint arXiv:1703.04977.
Kim, T.-K., & Cipolla, R. (2008). Canonical correlation analysis of video volume tensors for action categorization and detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1415–1428.
Kim, T. S., & Reiter, A. (2017). Interpretable 3d human action analysis with temporal convolutional networks. In *CVPRW* (pp. 1623–1631).
Li, T., Liu, J., Zhang, W., Ni, Y., Wang, W., & Li, Z. (2021). Uav-human: A large benchmark for human behavior understanding with unmanned aerial vehicles. In *CVPR* (pp. 16266–16275).
Liu, H., Simonyan, K., & Yang, Y. (2019). Arts: Differentiable architecture search. In *ICLR*.
Lohit, S., Wang, Q., & Turaga, P. (2019). Temporal transformer networks: Joint learning of invariant and discriminative time warping. In *CVPR* (pp. 12426–12435).
Lovrić, M., Min-Oo, M., & Ruh, E. A. (2000). Multivariate normal distributions parametrized as a riemannian symmetric space. *Journal of Multivariate Analysis*, 36–48.
Lu, J., Liong, V. E., & Zhou, J. (2017). Simultaneous local binary feature learning and encoding for homogeneous and heterogeneous face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1979–1993.
Lu, J., Wang, G., Deng, W., Moulin, P., & Zhou, J. (2015). Multi-manifold deep metric learning for image set classification. In *CVPR* (pp. 1137–1145).
Lu, J., Wang, G., & Moulin, P. (2013). Image set classification using holistic multiple order statistics features and localized multi-kernel metric learning. In *ICCV* (pp. 329–336).
Nadeem, U., Shah, S. A. A., Bennamoun, M., Togneri, R., & Sohel, F. (0000). Real time surveillance for low resolution and limited-data scenarios: an image set classification approach, arXiv preprint arXiv:1803.09470.
Nguyen, X. S., Brun, L., Lézoray, O., & Bougleux, S. (2019). A neural network based on spd manifold learning for skeleton-based hand gesture recognition. In *CVPR* (pp. 12036–12045).
Pennec, X., Fillard, P., & Ayache, N. (2006). A riemannian framework for tensor computing. *International Journal of Computer Vision*, 41–66.
Rahaman, R., & Thiery, A. H. (2020). Uncertainty quantification and deep ensembles. arXiv preprint arXiv:2007.08792.
Rahmani, H., & Mian, A. (2016). 3D action recognition from novel viewpoints. *CVPR* (pp. 1506–1515).
Rao, Y., Lu, J., & Zhou, J. (2019). Learning discriminative aggregation network for video-based face recognition and person re-identification. *International Journal of Computer Vision*, 701–718.
Shroff, N., Turaga, P., & Chellappa, R. (2010). Moving vistas: Exploiting motion for describing scenes. In *CVPR* (pp. 1911–1918).
Sun, H., Zhen, X., Zheng, Y., Yang, G., Yin, Y., & Li, S. (2017). Learning deep match kernels for image-set classification. In *CVPR* (pp. 3307–3316).
Tang, F., Feng, H., Tino, P., Si, B., & Ji, D. (2021). Probabilistic learning vector quantization on manifold of symmetric positive definite matrices. *Neural Networks*, 105–118.
Tekin, B., Bogo, F., & Pollefeys, M. (2019). H+o: Unified egocentric recognition of 3d hand-object poses and interactions. In *CVPR* (pp. 4511–4520).
Vemulapalli, R., Arrate, F., & Chellappa, R. (2014). Human action recognition by representing 3d skeletons as points in a lie group. In *CVPR* (pp. 588–595).
Vemulapalli, R., Pillai, J. K., & Chellappa, R. (2013). Kernel learning for extrinsic classification of manifold features. In *CVPR* (pp. 1782–1789).
Wang, R., Guo, H., Davis, L. S., & Dai, Q. (2012). Covariance discriminative learning: A natural and efficient approach to image set classification. In *CVPR* (pp. 2496–2503).
Wang, R., Wu, X.-J., & Kittler, J. (2020). Graph embedding multi-kernel metric learning for image set classification with grassmann manifold-valued features. *IEEE Trans. Multimedia*, 228–242.
Wang, R., Wu, X.-J., & Kittler, J. (2021). Symnet: A simple symmetric positive definite manifold deep learning method for image set classification. *IEEE Transactions on Neural Networks and Learning Systems*, 1–15.
Wang, R., Wu, X.-J., Liu, Z., & Kittler, J. (2021). Geometry-aware graph embedding projection metric learning for image set classification. *IEEE Transactions on Cognitive and Developmental Systems*, 1.
Yan, W., Sun, Q., Sun, H., & Li, Y. (2020). Semi-supervised learning framework based on statistical analysis for image set classification. *Pattern Recognition*, Article 107500.
Yan, S., Xiong, Y., & Lin, D. (2018). Spatial temporal graph convolutional networks for skeleton-based action recognition. In *AAAI*.
Zhang, X., Wang, Y., Gou, M., Sznaier, M., & Camps, O. (2016). Efficient temporal sequence comparison and classification using gram matrix embeddings on a riemannian manifold. In *CVPR* (pp. 4498–4507).
Zhang, J., Wang, L., Zhou, L., & Li, W. (2015). Learning discriminative stein kernel for spd matrices and its applications. *IEEE Transactions on Neural Networks and Learning Systems*, 1020–1033.
Zhang, G., Yang, J., Zheng, Y., Luo, Z., & Zhang, J. (2021). Optimal discriminative feature and dictionary learning for image set classification. *Information Sciences*, 498–513.
Zhang, T., Zheng, W., Cui, Z., Zong, Y., Li, C., Zhou, X., et al. (2020). Deep manifold-to-manifold transforming network for skeleton-based action recognition. *IEEE Transactions on Multimedia*, 2926–2937.
Zhou, L., Wang, L., Zhang, J., Shi, Y., & Gao, Y. (2017). *Revisiting Metric Learning for Spd Matrix Based Visual Representation* (pp. 3241–3249).