Learning Neural Bag-of-Matrix-Summarization with Riemannian Network

Hong Liu[†*], Jie Li[†*], Yongjian Wu[‡], Rongrong Ji[†♮],
[†]Fujian Key Laboratory of Sensing and Computing for Smart City,
School of Information Science and Engineering, Xiamen University, China
[‡]Tencent Youtu Lab, Tencent Technology (Shanghai) Co.,Ltd, China
lynnliu.xmu@gmail.com, lijie32@stu.xmu.edu.cn, littlekenwu@tencent.com, rrji@xmu.edu.cn
[*]Contributed Equally, [♮]Corresponding Author

Abstract

Symmetric positive defined (SPD) matrix has attracted increasing research focus in image/video analysis, which merits in capturing the Riemannian geometry in its structured 2D feature representation. However, computation in the vector space on SPD matrices cannot capture the geometric properties, which corrupts the classification performance. To this end, Riemannian based deep network has become a promising solution for SPD matrix classification, because of its excellence in performing non-linear learning over SPD matrix. Besides, Riemannian metric learning typically adopts a kNN classifier that cannot be extended to large-scale datasets, which limits its application in many time-efficient scenarios. In this paper, we propose a Bag-of-Matrix-Summarization (BoMS) method to be combined with Riemannian network, which handles the above issues towards highly efficient and scalable SPD feature representation. Our key innovation lies in the idea of summarizing data in a Riemannian geometric space instead of the vector space. First, the whole training set is compressed with a small number of matrix features to ensure high scalability. Second, given such a compressed set, a constant-length vector representation is extracted by efficiently measuring the distribution variations between the summarized data and the latent feature of the Riemannian network. Finally, the proposed BoMS descriptor is integrated into the Riemannian network, upon which the whole framework is end-to-end trained via matrix back-propagation. Experiments on four different classification tasks demonstrate the superior performance of the proposed method over the state-of-the-art methods.

Introduction

Symmetric Positive Defined (SPD) matrix has been recently popular for feature representation in various computer vision and artificial intelligence applications, *e.g.*, image classification (Fathy and Chellappa 2017; Wang, Li, and Zhang 2017), action recognition (Guo, Ishwar, and Konrad 2013; Huang et al. 2017a), and brain computer interface (BCI) data analysis (Lotte et al. 2018). Existing works in feature representation with SPD matrix can be categorized into either using covariance matrix (Wang et al. 2012) or using Gaussian distribution matrix (Wang et al. 2015b). The former is to preserve the second-order statistics of a set of vectors, while the latter targets at capturing the overall probability of data variations. By preserving such non-Euclidean geometric properties, SPD matrices can measure nearness on a specific Riemannian manifold (Sra 2012; Harandi, Salzmann, and Baktashmotlagh) rather than in the Euclidean space, which considers the geodesic distance between two points on the Riemannian manifold. As observed in (Arsigny et al. 2007), by using SPD matrices, the Riemannian space eliminates the large swelling effect that is previously existed in the Euclidean space, which brings significant advantages in handling various problems suiting for non-Euclidean metrics. However, directly applying traditional machine learning algorithms with Euclidean geometry to SPD metrics often results in poor performance (Huang et al. 2017b). To overcome this problem, the Riemannian metric based learning model has received increasing focus, which can directly conduct non-linear learning by fed into the SPD matrix representations (Pennec, Fillard, and Ayache 2006; Wang et al. 2015a; Yger and Sugiyama 2015).

Recently, deep learning methods have received much attention in visual feature representation. However, most schemes consider solely first-order statistics using traditional neural networks. More recently, the second-order statistics, such as covariance, are further considered to construct better regional descriptors to solve challenging problems like fine-grained visual recognition (Lin, RoyChowdhury, and Maji 2015; Lin and Maji 2017). These works do not use dimensionality reduction layers to obtain effective second-order statistics, which instead directly use the multiple fully-connected (FC) layers following the matrix vectorization processing of the SPD matrices. However, such non-Euclidean representation implies the lack of familiar properties such as global parameterization, common coordinates, vector space structure, and shift-invariance. Consequently, basic operations like FC layers cannot be well defined on the non-Euclidean domains (Bronstein et al. 2017). Moreover, such vanilla deep structure will destroy the Riemannian goemetry and corrupt the classification results, as demonstrated in our experiments.

To overcome these problems, Riemannian SPD Matrix Network (SPDNet) (Huang and Van Gool 2017) is proposed, which receives SPD matrices as inputs. It aims to preserve the SPD structure across layers to be non-linearly mapped into latent space and then does the task like classification on

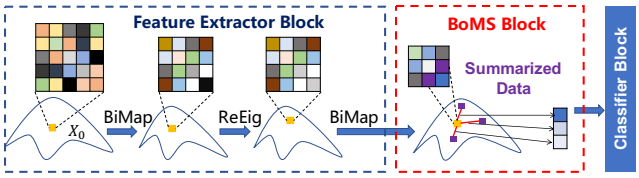


Figure 1: The framework of our proposed Bag-of-Matrix Summarization layer with SPDNet. (Best view in color)

this latent space. In particular, SPDNet is composed of two traditional layers (*i.e.*, the Fully-connected Layer and Softmax Layer) and three newly defined layers (*i.e.*, the BiMap Layer, ReEig Layer, and LogEig Layer). A special matrix back-propagation method with stochastic gradient descend (SGD) was proposed to train the deep SPDNet in an end-to-end way. Note that, the LogEig layer in SPDNet is the key to transform the geometric space to the Euclidean space, which can then be plugged by the traditional neural layers. This layer needs to compute the matrix logarithms that dramatically increase the computational cost, while its moments do not have closed forms (Cherian et al. 2013). Moreover, the existing works (Guo, Ishwar, and Konrad 2013; Anirudh et al. 2017) typically adopt a flatten vector representation towards tangent approximation or rolling maps, and then uses SVM or kNN classifier to learn features in the resulting flattened space. These shallow learning schemes have led to suboptimal solutions on the specific nonlinear manifolds, which also often require significantly more time to conduct online predictions with complex calculation. However, to preserve the original geometric relation that is captured by such a matrix structure, we argue that a better feature representation, *e.g.*, from a statistical perspective, is needed for various real-world applications.

To handle this issue, we propose to embed SPD matrices by using a Bag-of-Visual-Word (BoVW) model under a metric learning framework. Recently, BoVW has been integrated into the convolutional neural networks to perform image classification and simultaneously compress the model (Passalis and Tefas 2017). However, due to the SPD constraints, directly using BoVW is intractable, which serves as the first problem need to be tackled in this paper. Although recent works in (Sivalingam et al. 2015; Cherian et al. 2017) have extended the dictionary learning to SPD matrix representation with encoding model, the integration of BoVW into deep Riemannian networks remains an open problem due to the difficulty in optimization, which serves as the second problem.

To solve the above two problems, we propose a novel Bag-of-Matrix-Summarization (BoMS) model, which can be efficiently inferred, be well scaled up to large dataset, and can significantly improve the classification accuracy. The BoMS is based on supervised learning that is designed to learn nonlinear transformations to preserve the neighbor structure with the labeled data. It simultaneously learns an extremely small set of codewords, procedure of which can be seen as a nonlinear Riemannian metric learning, *i.e.*, a summarized version of the low-dimensional SPD matrices. The summarized data can be seen as the codewords, each of which contains the corresponding semantic information like objective labels. Then, the output of BiMap layer in SPDNet can be embedded into a vector representation, each dimension of which captures the respective divergence of the codeword output. Finally, the data summarization and feature learning are learned jointly through matrix back-propagation with stochastic gradient descent (SGD) (Ionescu, Vantzos, and Sminchisescu 2015), which ensures our model to be scalable to large training sets. In particular, BoMS acts as a trainable encoding layer, which can be plugged between the BiMap layer and the FC layer to replace the LogEig Layer in the original SPDNet. A sequence of BiMap and ReEig are further used to construct the feature extractor, forming a low-dimensional SPD matrix input to the corresponding classification/recognition model.

We term the proposed scheme BoMS+SPDNet, the framework of which is shown in Fig.1. Quantitatively, we compare the proposed model against various state-of-the-art SPD matrix based classification methods on four benchmarks, *i.e.*, AFEW, HDM05, YTC, and BCI. Experiments demonstrate that the proposed BoMS+SPDNet outperforms the existing classification methods in terms of both accuracy and efficiency. The rest of this paper is organized as follows: In Sec.2, we briefly overview the SPDNet, which serves as basis to perform SPD matrix classification. Sec.3 describes the proposed BoMS+SPDNet and the experiments are shown in Sec.4. Finally, we conclude this paper in Sec.5.

Preliminaries of SPDNet

We first briefly present the SPDNet (Huang and Van Gool 2017), which serves as the basis for the proposed BoMS model. SPDNet is the first deep Riemannian network with four different layers, *i.e.*, BiMap Layer, ReEig Layer, LogEig Layer, and other layers. It receives SPD matrices as inputs, preserves the Riemannian manifold structure across layers, and non-linearly maps an input matrix into a vector representation. Let $\mathbf{X}_{k-1} \in \mathrm{Sym}^+_{d_{k-1}}$ be the SPD matrix of the k-th layer, $\mathbf{W}_k \in \mathbb{R}^{d_k \times d_{k-1}}$ ($d_k < d_{k-1}$) be the transformation matrix in the k-th layer, and $\mathbf{X}_k \in \mathbb{R}^{d_k \times d_k}$ be the resulting matrix in the k-th layer, where $\mathrm{Sym}^+_{d_{k-1}}$ is the space of SPD real $d_{k-1} \times d_{k-1}$ matrices.

The **BiMap Layer** is similar to the linear transformation layer in auto-encoder, which transforms the input SPD matrices to the low-dimensional SPD matrices by a bilinear mapping f_b as:

$$\mathbf{X}_k = f_b^{(k)}(\mathbf{X}_k; \mathbf{X}_{k-1}) = \mathbf{W}_k \mathbf{X}_{k-1} \mathbf{W}_k^T. \quad (1)$$

To make the output $\mathbf{X}_k$ remain a SPD matrix, the transformation $\mathbf{W}_k$ should be constrained to a raw full-rank matrix.

The **ReEig Layer** is to utilize non-linear activation to improve discrimination, which is similar to the ReLU layer in the convolutional neural networks (ConvNets) (Nair and Hinton 2010). As a consequence, ReEig Layer is devised to a non-linear function f_r, which rectifies the SPD matrices by tuning up their small positive eigenvalues:

$$\mathbf{X}_k = f_r^{(k)}(\mathbf{X}_{k-1}) = \mathbf{U}_{k-1} \max(\epsilon \mathbf{I}, \mathbf{\Sigma}_{k-1}) \mathbf{U}_{k-1}^T, \quad (2)$$

where $\max(\cdot,\cdot)$ is the maximum function, $\mathbf{U}_{k-1}$ and $\mathbf{\Sigma}_{k-1}$ are learned by eigenvalue decomposition of $\mathbf{X}_{k-1} = \mathbf{U}_{k-1}\mathbf{\Sigma}_{k-1}\mathbf{U}_{k-1}^T$, ϵ is a threshold parameter, and $\mathbf{I}$ is the identity matrix.

The **LogEig Layer** is similar to the Log-Euclidean Riemannian metric learning (Huang et al. 2015), in which the matrix logarithm operation $\log()$ is done on the SPD matrices, and then the resulting matrix is flatted to a vector representation. As a result, the classical Euclidean computations can be applied to the logarithms of SPD matrix. Formally, the layer can be defined as a function f_l as:

$$\mathbf{X}_k = f_l^{(k)}(\mathbf{X}_{k-1}) = \log(\mathbf{X}_{k-1}) = \mathbf{U}_{k-1}\log(\mathbf{\Sigma}_{k-1})\mathbf{U}_{k-1}^T, \tag{3}$$

where $\mathbf{X}_{k-1} = \mathbf{U}_{k-1}\mathbf{\Sigma}_{k-1}\mathbf{U}_{k-1}^T$ is the eigenvalue decomposition.

Finally, the **Other Layers** are composed of a sequence of neural blocks in the traditional neural networks, *i.e.*, Fully Connected (FC) layer and Softmax layers.FC layer is inserted after the LogEig Layer, which is set to be a projection matrix $\mathbf{W}_{fc} \in \mathbb{R}^{d_k \times d_{k-1}}$ where d_k is the class number and d_{k-1} is the dimension of the output of LogEig Layer. The final output for classification can be a Softmax layers. To learn this SPDNet, inspired by the matrix BP (Ionescu, Vantzos, and Sminchisescu 2015), the back-propagation (BP) with an SGD setting on Stiefel manifolds was proposed, which makes the SPDNet trainable in an end-to-end setting.

The Proposed Method

As illustrated in Fig.1, the proposed BoMS+SPDNet is composed of three blocks: a) a Feature Extraction Layer block (composed of several BiMap Layers and ReEig Layers), b) a BoMS Layer Block, and c) a Classification Layer Block. We depict the details as below:

Feature Extraction Layer Block

As shown in the left part of Fig.1, feature extraction is the fundamental block of the proposed method, which aims to extract low dimensional SPD matrix that serves as the input for the subsequent Riemannian feature learning process. Inspired by the removal of fully-connected layers in the recent deep feature extractor (Passalis and Tefas 2017), for the i-th SPD feature extractor, we remove the LogEig Layer and the Other Layers in SPDNet, the rest of which is composed into a block sequence of BiMap Layer and ReEig Layer. In our feature extractor, we use a similar architecture of SPDNet with three BiMap layers $f_b^{(k)}$ and two ReEig layers $f_r^{(k)}$, the exemplar structure of which is $\mathbf{X}_0 \rightarrow f_b^1 \rightarrow f_r^2 \rightarrow f_b^3 \rightarrow f_r^4 \rightarrow \cdots \rightarrow f_b^l \rightarrow \mathbf{Z}$. The output of the last BiMap Layer, denoted by the l-th layer, is used to feature extractor and is subsequently fed into the BoMS block. Without loss of generality, we define the output of the feature extractor as a function $F(\mathbf{X}_0) = \mathbf{Z}$[1], which is subsequently fed into the proposed BoMS block.

[1]Note that, the output $\mathbf{Z}$ is still SPD matrix, which can still hold the Riemannian geometric properties.

BoMS Layer Block

Similar to the BoVW model, the proposed BoMS layer is an encoding layer that captures the statistics of the feature matrix $\mathbf{Z}$. The goal of learning the BoMS layer is two-fold: a) learn a dictionary set $\mathcal{B} = \{\mathbf{B}_1, \mathbf{B}_2, ..., \mathbf{B}_m\}$ with m SPD matrices, where each dictionary $\mathbf{B}_i \in \mathrm{Sym}_d^+$, b) learn an accumulating scalar on each dictionary atom to best represent the SPD matrix feature $\mathbf{Z}$ for classification,

We denote $\mathbf{Z}_i \in \mathrm{Sym}_d^+$ as the i-th resulting feature through the feature extractor with input SPD matrix $\mathbf{X}^i$, which can be collected as a set of N matrix features as $\mathcal{Z} = \{\mathbf{Z}_1, \mathbf{Z}_2, ..., \mathbf{Z}_N\}$ with associated labels $y_i \in \mathcal{Y} = \{y_1, y_2, ..., y_N\}$. BoMS aims to output a fixed-length vector representation $\mathbf{v}$ based on the dictionary set $\mathcal{B}$. The proposed layer can be viewed as a unified processing layer, whose output is sent to a subsequent classifier. The output of the BoVW can be defined by a nonlinear function f_p as:

$$\mathbf{v}_i = f_p(\mathbf{Z}_i) = \left[D(\mathbf{Z}_i, \mathbf{B}_1), ..., D(\mathbf{Z}_i, \mathbf{B}_m)\right]^T \in \mathbb{R}^m, \tag{4}$$

where function $D(\cdot,\cdot)$ is the Riemannian metric to measure two given SPD matrices.

Therefore, the key issue is how to define such a dictionary set $\mathcal{B}$, which is typically achieved by k-means clustering. However, the input features here are in a matrix style, therefore the traditional clustering algorithms are not workable. On one hand, some methods have been proposed to cluster the SPD matrix features (Cherian, Morellas, and Papanikolopoulos 2016), they are unsuitable to be integrated into the deep learning architecture, in which the dictionary updating is separated with the feature learning, leading to a suboptimal learning. On the other hand, the clustering algorithm is unsupervised and cannot utilize the label information to improve the quality of the dictionaries.

To solve this problem, inspired by the data summarization technology (Kusner et al. 2014), the goal of our dictionary learning is to find a set of summarized samples $\hat{\mathcal{Z}} = \{\hat{\mathbf{Z}}_1, ..., \hat{\mathbf{Z}}_m\}$ ($m \ll N$) with labels $\hat{\mathcal{Y}} = \{\hat{y}_1, ..., \hat{y}_m\}$ to replace the dictionary $\mathcal{B}$, so that the original training data $\mathcal{Z}$ and $\mathcal{Y}$ can be best approximated via k-nearest neighbors. Different from the traditional BoVW, the summarization set $\hat{\mathcal{Z}}$ needs to be learned from the whole dataset, and the summarized label set $\hat{\mathcal{Y}}$ should have the same proportional distribution to the original label set[2]. Note that, the data summarization can be viewed as a special case of supervised dictionary learning, which aims to correctly classify as many training inputs as possible in the deep matrix space.

However, two issues of data summarization need to be further solved: 1) the learned metric should maximize the margin between different classes. 2) all the summarized data with the same labels maybe converge into a single point, as we target at maximizing the classification accuracy. To solve the first problem, we propose a margin-based loss function

[2]That is, if one category accounts for 60% of the original label collection, it should occupies the same percentage in the summarization collection.

for the matrix summarization learning, which is defined as:

$$L^1_{ms}(\mathcal{Z},\hat{\mathcal{Z}}) = \sum_{i=1}^{N}\sum_{j,k=1}^{m}\big[0,\alpha - D(\mathbf{Z}_i,\hat{\mathbf{Z}}_j) + D(\mathbf{Z}_i,\hat{\mathbf{Z}}_k)\big],$$
$$s.t.\ y_i = y_j \ \text{ and } \ y_i \neq y_k \tag{5}$$

For the second problem, the summarized data with the same label should be dissociated, so that such samples can best present the diversity of the training data. To this end, the target is to maximize the pair-wise distance between two input SPD matrices with the same label, that is:

$$L^2_{ms}(\hat{\mathcal{Z}}) = \sum_{i=1}^{m}\sum_{j=1}^{m}\delta(y_i,y_j)D(\hat{\mathbf{Z}}_i,\hat{\mathbf{Z}}_j), \tag{6}$$

where $\delta(y_i,y_j)=1$ if $y_i=y_j$, and 0 otherwise. Then, the final objective function of the proposed data summarization is to combine Eq.5 and Eq.6 as follows:

$$L_{ms}(\mathcal{Z},\hat{\mathcal{Z}}) = \lambda_1 L^1_{ms}(\mathcal{Z},\hat{\mathcal{Z}}) - \lambda_2 L^2_{ms}(\hat{\mathcal{Z}}), \tag{7}$$

where λ_1 and λ_2 are two tradeoff parameters to control the weights between two functions Eq.5 and Eq.6. As a result, the proposed Bag-of-Matrix-Summarization (BoMS) layer can be defined as:

$$\mathbf{v}_i = \Big[D\big(F(\mathbf{X}^i),\hat{\mathbf{Z}}_1\big),...,D\big(F(\mathbf{X}^i),\hat{\mathbf{Z}}_m\big)\Big]^T. \tag{8}$$

Distance Metric

To learn better vector representation for the corresponding SPD matrix, the key issue is to define an appropriated distance metric $D(\cdot,\cdot)$ in Eq.8. To this end, we introduce three representative distance metric below.

Inspired by the study in (Arsigny et al. 2006), we first use the *Log-Euclidean metric (LEM)* to define the distance function $D(\cdot,\cdot)$ in Eq.4, which aims at exploiting the Lie group structure under the typical matrix exponential and logarithm operators. Such that, the Riemannian distance between two SPD matrices is defined by LEM, *i.e.*,

$$D(\mathbf{Z}_i,\mathbf{Z}_j) = \|\log(\mathbf{Z}_i) - \log(\mathbf{Z}_j)\|, \tag{9}$$

where $\|\cdot\|$ is the Frobenius norm of the matrix.

However, as mentioned in (Cherian et al. 2013), the flattening of the manifold in LEM often leads to less accurate distance computations and therefore affect the performance. To solve this problem, an intuitive method is to use metric learning method to reduce such computation error, which serves as our second distance measure, *i.e.*,

$$D(\mathbf{Z}_i,\mathbf{Z}_j) = \big\|\mathbf{W}\mathrm{vec}\big(\log(\mathbf{Z}_i)\big) - \mathbf{W}\mathrm{vec}\big(\log(\mathbf{Z}_j)\big)\big\|_2, \tag{10}$$

where $\mathrm{vec}(\cdot)$ is the matrix vectorization processing.

On the other hand, LEM is interpreted as an Euclidean distance between the matrices mapped in the tangent space at the identity, which implies a deformation. Therefore, a more natural measure should be considered to hold the Riemannian geometry. To this end, we consider an effective and efficiency divergence measure, termed Jensen-Bregman LogDet Divergence (JBLD), which redefines Eq.4 as:

$$D(\mathbf{Z}_i,\mathbf{Z}_j) = \log|(\mathbf{Z}_i+\mathbf{Z}_j)/2| - 0.5*\log|\mathbf{Z}_i\mathbf{Z}_j|, \tag{11}$$

where $|\cdot|$ denotes the determinant.

Classification Block

The final block is to perform the classification, which is formulated via the following objective function:

$$L_c(\mathcal{Z},\mathcal{W}) = \sum\nolimits_{i=1}^{N} f(\mathbf{v}_i,y_i;\mathcal{W}), \tag{12}$$

where the function $f(\cdot)$ with parameter set $\mathcal{W}$ aims at learning the classifier on $\mathbf{v}_i$ according to the provided class labels y_i, and $\mathbf{v}_i$ is the resulting vector representation from the proposed BoMS layer. There are several choices for the definition of f, we resort to the cross-entropy loss function with FC layer and softmax layer.

At last, combing L_{ms} and L_c together, we get the final loss function for SPD classification as:

$$L = L_{ms}\big(F(\mathcal{X}),\hat{\mathcal{Z}}\big) + L_c\Big(f_p\big(F(\mathcal{X})\big),\mathcal{W}\Big). \tag{13}$$

Learning with BoMS

For the SPD matrix based classification, the proposed BoMS can be directly integrated into the SPDNet, which can be written as a composition of sequentially connected functions with the input SPD matrix $\mathbf{X}$ and the output predicted class label. To train such a deep network, one can use the matrix back-propagation (Ionescu, Vantzos, and Sminchisescu 2015) together with stochastic gradient descent. Fortunately, the gradients of the parameters in the FC layer and Softmax layer can be easily calculated in the traditional ways, as these layers lie in the Euclidean space. The major problem here is that the matrix must hold the SPD constraint in the BiMap layer, the ReEig layer, and the proposed BoMS layer.

For the gradients of function F in Eq.8 containing BiMap layer and ReEig layer, similar to that in SPDNet, we use a customized updating on Stiefel manifolds. For the proposed BoMS layer, the gradients of the corresponding parameters can be from two information flows: One is the gradients from the classification block, and the other is from the loss of the data summarization in Eq.7. Specifically, there are three components that need to be updated: the layer parameters $\mathcal{Q}$, the summarized data $\hat{\mathbf{Z}}$, and the gradient that is propagated to the feature extraction block F.

As for each summarized $\hat{\mathbf{Z}}_i$ and function F, the updating schemes are achieved by the following chain rule:

$$\frac{\partial L}{\partial \hat{\mathbf{Z}}_i} = \Big(\frac{\partial L^1_{ms}}{\partial f_p} - \frac{\partial L^2_{ms}}{\partial f_p} + \frac{\partial L_c}{\partial \mathbf{v}^i}\circ\frac{\partial \mathbf{v}^i}{\partial f_p}\Big)\circ\frac{\partial f_p}{\partial \hat{\mathbf{Z}}_i}, \tag{14}$$

$$\frac{\partial L}{\partial F} = \Big(\frac{\partial L^1_{ms}}{\partial f_p} + \sum_{i=1}^{m}\big(\frac{\partial L_c}{\partial \mathbf{v}^i}\circ\frac{\partial \mathbf{v}^i}{\partial f_p}\big)\Big)\circ\frac{\partial f_p}{\partial F}, \tag{15}$$

where $\mathbf{v}^i = f_D\big(F(\mathbf{X}),\hat{\mathbf{Z}}_i\big)$ is the i-th dimension in the BoMS feature.

Due to the different distance measures from Eq.9 to Eq.11, we use different updating schemes to calculate the gradients. For Eq.9, the function f_p can be replaced by f_l in Eq.3, and the calculating gradients $\frac{\partial f_p}{\partial \mathbf{Z}_i}$ and $\frac{\partial f_p}{\partial F}$ are with the same updating rules in SPDNet. Comparing to Eq.9, Eq.10 has an additional parameters $\mathbf{W}$ whose updating scheme is:

$$\frac{\partial L}{\partial \mathbf{W}} = \frac{\partial L^1_{ms}}{\partial \mathbf{W}} - \frac{\partial L^2_{ms}}{\partial \mathbf{W}} + \sum_{i=1}^{m}\Big(\frac{\partial L_c}{\partial f_p}\circ\frac{\partial f_p}{\partial \mathbf{W}}\Big). \tag{16}$$

For Eq.11, the gradient is $\partial D/\partial \mathbf{Z}_i = (\mathbf{Z}_i + \mathbf{Z}_j)^{-1} - 0.5 * \mathbf{Z}_i^{-1}$. Therefore, the gradients of $\dot{\mathbf{Z}}_i$ and F can be easily calculated as similar to the LogEig layer.

Discussion

The proposed feature learning layer can better present the SPD matrix as a vector to do the classification task. We now show the relationship between the BoMS layer and the LogEig layer: If we replace the summarized data $\hat{\mathcal{Z}}$ in Eq.9 to the Identity matrix, the loss function Eq.7 can be reduced to a scalar constant. As a result, when $\hat{\mathcal{Z}}$ is the identity matrix, the proposed BoMS will degenerate into the original LogEig layer. Therefore, the LogEig layer can be viewed as a special case of our BoMS. Moreover, the summarized data are generated from supervised dictionary learning, which not only influents the previous feature network with the predefined metric, but also helps learn a better classifier. Adding these two points into Riemannian network can further improve the classification accuracy, whose quantitative evidences will be shown in our experiments. Finally, the proposed BoMS is more flexible and more scalable, where the distance function can be replaced with better metrics to reflect the Riemannian property, as also demonstrated in our experiments.

Experiments

In this section, we evaluate our BoMS model on SPDNet to the state-of-the-art SPD matrix based classification methods on four different tasks, *i.e.,* emotion recognition, action recognition, face recognition and brain computer interface.

1) **Emotion Recognition**. We use Acted Facial Expression in the Wild (*AFEW*) dataset, which collects $1,345$ video sequences of facial expressions acted by 330 actors in movies. This dataset has been divided into training, validation, and test sets, where each video is classified into one of seven expressions. Since the ground truth of the test set has not been released, we follow the setting in (Liu et al. 2014; Huang and Van Gool 2017) to evaluate the performance on the validation set. To augment the training set, we also segment the training videos into $1,747$ small clips. And each facial frame is normalized to an image of size 20×20. Then, we compute the covariance matrix feature of size 400×400.

2) **Action Recognition**. We evaluate our model on the task of skeleton-based human action recognition using the *HDM05* dataset, which is a large-scale dataset for SPD matrix based representation. This dataset contains $2,337$ sequences of 130 action classes, which provides 3D locations of 31 joints of the subjects. To preprocess this dataset, we divide the training sequences set to around $18,000$ small subsequences. Then we represent each sequence by a covariance descriptor of size 93×93, which is calculated by a second order statistics of the 3D coordinates for the 31 joints in each frame.

3) **Face Recognition**. We use the YouTube Celebrities (TYC) dataset to perform video face recognition, which contains $1,910$ video clips of 47 subjects collected from YouTube, and most of the clips contain hundreds of frames. The dataset is randomly split into a training set and a testing set, with a splitting ratio of $1:2$. Each face image in a video is cropped into a 20×20 intensity image and is then histogram-equalized to eliminate lighting effects. We extract the set-based covariance matrix for each video sequence in this dataset, the matrix size of which is 401×401.

4) **BCI classification**. We further evaluate the classification performance on the BCI Competition IV dataset 2a (*BCI*) [3], which is a 22-electrode EEG motor-imagery dataset. It consists of 9 subjects and 2 sessions, each subject of which has 288 four-second trials of imagined movements. To preprocess this dataset, we train the models on each subject on the first session, and test on its corresponding subject on the second session. We report the average precision results among 9 subjects. As the similar preprocessing to (Schirrmeister et al. 2017), for each channel, all the EED data are first band-pass filtered with a bandwidth of $4-38$Hz, and electrode-wise exponential moving standardization is then performed to compute exponential moving means and variances, both of which are used to standardize the continuous data. As a result, each EEG signal is represented by a 22×22 SPD matrix.

Compared Methods. We mainly compared to six state-of-the-art SPD matrix learning methods: Covariance Discriminative Learning (*CDL*) (Wang et al. 2012), Log-Euclidean Metric Learning (*LEML*) (Huang and Van Gool 2017), SPD Manifold Learning (*SPDML*) (Harandi, Salzmann, and Hartley 2017) that uses affine-invariant metric (*SPDML-AIM*) (Pennec, Fillard, and Ayache 2006) and Stein divergence (*SPDML-Stein*) (Sra 2016), Riemannian Sparse Representation (*RSR*) (Harandi et al. 2012), Matrix-Square Root Normalization (*MSRN*) (Lin and Maji 2017), and Riemannian Network (*SPDNet*) (Huang and Van Gool 2017). For the above methods, we use the source codes kindly provided by the authors and tune the parameters according to the original settings. For SPDNet, we use the settings with the best performance from the original work, which uses three blocks of BiMap/ReEig layer for AFEW and HDM05, and uses one BiMap layer and one ReEig layer for BCI competition. To verify the efficiency of Riemannian-based network, we also compare to the vanilla neural network, which is composed of multiple fully-connected layers and one softmax layer. For MSRN, we use the best structure in (Lin and Maji 2017), which contains matrix-square root, element-wise signed square-root normalization, FC layer, and softmax layer. In details of vanilla NN and MSRN, three FC layers are used on AFEW and HDM05, and one FC layer is used on BCI.

The Proposed Riemannian Network. BoMS mainly contains three types of measures, named BoMS-1, BoMS-2, and BoMS-3 according to Eq.9 to Eq.11, respectively. To further validate the proposed model, we also compare to the abbreviated versions based on the proposed BoMS-2: We first delete the metric learning part in Eq.5, termed BoMS$-$ML, which considers the BoMS as the traditional BoVW model, as similar to (Passalis and Tefas 2017). We then delete the loss taking account of the data divergence in Eq.6, while preserving the metric learning loss. We name this method as BoMS$-$D. Finally, we delete all the part of

[3] http://www.bbci.de/competition/iv/

Table 1: The results for the AFEW, HDM05, and BCI_4_2A datasets. Note that the baseline results on AFEW and HDM05 are cited from (Huang and Van Gool 2017), which we have also reproduced. All accuracy rates are averaged number. The last column shows the testing time for different methods, which is conducted on all testing set in AFEW.

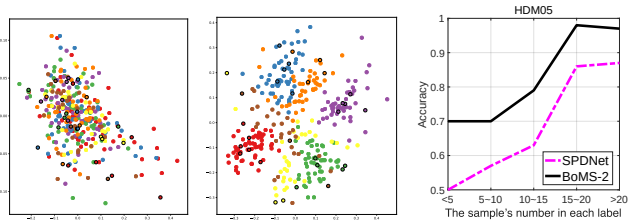
Method	AFEW	HDM05	BCI	YTC	Times
CDL	31.81%	41.74%	45.02%	83.99%	2243s
LEML	25.13%	46.87%	40.39%	81.93%	1823s
SPDML-AIM	26.72%	47.25%	57.72%	80.49%	5366s
SPDML-Stein	24.55%	46.21%	53.47%	74.52%	1849s
RSR	27.49%	41.12%	45.49%	81.8%	4841s
MSRN	N/A	59.92%	47.72%	-	N/A
SPDNet	34.23%	61.45%	55.59%	89.01%	6.29 s
BoMS-1	35.04%	71.03%	61.65%	-	6.39 s
BoMS-2	**38.81%**	**71.79%**	**65.20%**	**89.81%**	6.49 s
BoMS-3	36.93%	70.42%	62.23%	-	11.53 s
BoMS-ML	34.23%	68.84%	61.11%	83.04%	-
BoMS-D	36.93%	70.80%	65.08%	89.04%	-
BoMS-A	32.88%	68.61%	60.42%	80.02%	-
Vanilla NN	N/A	49.29%	49.88%	-	N/A

data summarization in Eq.7 to evaluate the importance of BoMS model, named BoMS-A.

The Setting of Network Architecture. The architecture of the proposed BoMS+SPDNet is $\mathbf{X} \rightarrow F \rightarrow f_p \rightarrow f_r \rightarrow f_s \rightarrow \hat{y}$, where F, f_p, f_r, f_s and $\hat{y}$ indicate the feature extractor, BoMS, FC, softmax, and approximated label. For the first two datasets, we use the similar architecture of SPDNet with three BiMap layers $f_b^{(k)}$ and two ReEig layers $f_r^{(k)}$, the exemplar structure of which is $\mathbf{X}_0 \rightarrow f_b^1 \rightarrow f_r^2 \rightarrow f_b^3 \rightarrow f_r^4 \rightarrow f_b^5 \rightarrow \mathbf{Z}$. The parameters in AFEW are set to 400×200, 200×100, and 100×50, respectively. The parameters on HDM05 are set to 93×70, 70×50, and 50×30, respectively. And the parameters on YTC are set to 401×200, 200×100, and 100×50 , respectively. For the BCI dataset, we verify the performance with just one BiMap Layer and one ReEig Layer as the feature extractor, whose parameters are set to 22×15.

Parameter Setting. We implement our Riemannian Network with BoMS using Pytorch on a single PC with Dual Core I7-3421 and 128G memory. We use the stochastic gradient descent to update the network parameters, and the learning rate is set to 1×10^{-3} with 5×10^{-4} weight decay. The batch size is set to 30, the weights are initialized as random semi-orthogonal matrices, as similar to the SPDNet. As described before, the summarized data is uniformly sampled according to the label ratio. For the unbalanced label rations, we have at least one data for each category. For all three benchmarks, the scale of the summarized data set is selected based on the randomly sampled validation set. In all our experiments, we empirically set $\lambda_1 = 1.2$ and $\lambda_2 = 0.7$ according to parameter's tuning.

Results and Analysis. As shown in Tab.1, we report the proposed model with three different distance measures, which have 10.8%, 12.8%, and 10.1% accuracy improvements when compared to the second best methods respectively, such as SPDNet and SPDML-AIM. It is worth to note that, Riemannian based neural networks, *e.g.,* both SPDNet and ours, not only achieve competitive results, but also use less time for online prediction. When comparing with Vanilla NN and MSRN that have been widely used in bilinear models (Lin, RoyChowdhury, and Maji 2015; Lin and Maji 2017), our proposed models have average 35.3% accuracy improvements, respectively.



(a) Original data. (b) Summarization. (c) Acc vs. Label.

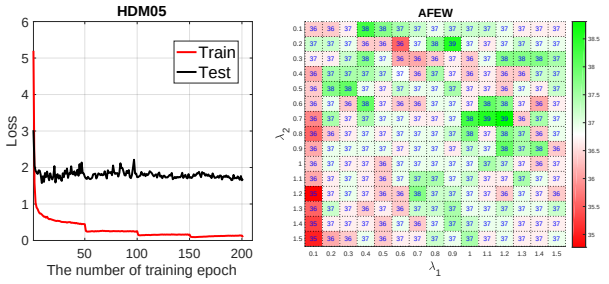
Figure 2: The subfigure (a-b) show the visualization of the summarized data on AFEW dataset . Each color point represents a category in the dataset. The figure (a) shows the initialized distribution, and the right (b) shows the distribution after training. The black circle point presents the summarized data. The subfigure (c) shows the detailed evaluation for imbalanced label dataset, HDM05. (Best view in color)

As the result in Tab.1, we report the testing time for different models. All the traditional Riemannian-based models require significantly more time to conduct online predictions, for which the k-NN classifier requires pairwise distance computation and comparison. Since the scale of SPD feature is 400×400 in AFEW, the vectorized dimension of such matrix is too high and make the training of both vanilla NN and MSRN intractable to achieve. Therefore, their accuracy and testing time are not reported here.

Moreover, HDM05 is an imbalance label dataset. We calculate the variance of precision for each label, and the average variance score of SPDNet is 0.1017 and that of BoMS-2 is 0.0935. More details experimental results are given in Fig.2 (c). As a conclusion, our work still achieves the best statistical results, which demonstrates that BoMS is more robust in the imbalance classification task.

In addition, our proposed method BoMS-1 calculates the LEM between features and summarized set multiple times. However, compared to SPDNet, but the method is still efficient. It is worth to note that, the testing time of BoMS-1 is faster than that of BoMS-3, which is due to the framework we used that can handle batch data directly. When we test BoMS-1 using the same calculation way as BoMS-3, the testing time is $15.55s$, which is slower than BoMS-3. As a result, the Riemannian based NNs are all effective and efficiency for online testing, which verifies the importance of the Riemannian-based NNs to solve the SPD matrix input.

Comparing BoMS-1 with SPDNet, although BoMS-1 also uses Log-Euclidean metric in calculation, the accuracies on all three benchmarks are better than SPDNet. To analyze, each dimension in vector $\mathbf{v}$ represents the probability of SPD feature belonging to the corresponding category,



(a) Convergence curve. (b) λ_1 vs. λ_2.

Figure 3: Convergence curve on the representative HDM05 dataset, and parameters' analysis on the representative AFEW dataset. (Best view in color)

where a small value in the i-th dimension means the feature is close to the category of the i-th summarized data. On the other hand, the summarized data can be defined as the importance samples in their corresponding categories, which leads to a higher information entropy for each dimension in the vector representation. When ignoring this part in Eq.7, the performance decreases significantly on all three benchmarks. It demonstrates that the data summarization is very important for improving the classification accuracy, which can be viewed as a supervised pooling layer. Besides, comparing to SPDNet, BoMS-A is always better and competitive, which demonstrates our argumentation that a better statistical feature representation is needed.

To demonstrate the divergence among the summarized data, we first plot the summarized data by class before or after training, as shown in Fig.2 (a-b). Before training, the distribution of data features was chaos and difficult to be distinguished between categories. After training, category information can be easily separated. Besides, the summarized data (black circle points) is not aggregated to the center of the category, but is relatively dispersed within the category.

However, the Log-Euclidean metric has some inherent disadvantages. We therefore propose two other solutions to further improve the accuracy, such as BoMS-2 and BoMS-3. When metric learning is introduced into the Log-Euclidean metric, the performance is the best. Although BoMS-3 achieves the second best performance in all three benchmarks, its training is very efficient. It is worth to note that, the accuracy of Jensen-Bregman LogDet Divergence can be also improved with metric learning.

The results on three tasks show that the proposed BoMS has superior classification accuracy. The convergence curve of our network is shown in Fig.3(a), which suggests that our Riemannian network can converge quickly. In addition, we analyze the validity of different parts of the proposed model, by which we have found that the representative model BoMS-2 achieves the best result. Consequently, we evaluate the classification results by simultaneously tuning the parameters λ_1 and λ_2 on the validation set for all three datasets. And the results on representative AFEW are shown in Fig.3 (b). We have found that the best accuracy is achieved when empirically setting $\lambda_1 = 1.2$ and $\lambda_2 = 0.7$, which is consistent in all datasets.

Table 2: The training time of each epoch (Times), compression ratio (CR.), and accuracy (ACC.) with different scale of Summarized dataset, where the results is conducted on the representative AFEW dataset.

# $\hat{\mathcal{Z}}$	11	18	25	31	38	46	53
Times (s)	43.7	47.0	51.1	53.5	55.9	63.2	67.9
CR. (%)	0.62	1.03	1.43	1.78	2.18	2.63	3.03
Acc. (%)	35.3	36.4	36.4	38.8	35.9	34.8	34.5

Comparing to BoMS-D that sets $\lambda_2 = 0$, adding divergence can bring certain performance improvement, which indicates that the divergence should be considered in the data summarization. When we delete the triplet metric learning part, BoMS-ML shows a significant performance decrease comparing to BoMS-2. To analyze, as mentioned before, LEM often leads to less accurate distance measure, where metric learning needs to be considered to reduce such loss. The same phenomenon also appears in the comparison of BoMS-1 and BoMS-3.

The number m reflects the scale of the summarized set, whose relation to the classification accuracy is shown in Tab.2. The results show that either large or small number will lead to poor performance. A small number means the representative information contained in the summary data is limited, which therefore cannot conclude sufficient training data. In contrast, a large number means more unrelated data is combined to bring more noise. Moreover, Tab.2 shows the training time with the increasing number of summarized data. The results show that a large number needs longer training time. So, the suitable size not only improves the performance of classification, but also improves the training efficiency. According to the results, this number is set to 31, 258, 210 and 32 for AFEW, HDM05, YTC and BCI according to the accuracies on the validation set, respectively.

Conclusion

This paper has proposed a Bag-of-Matrix-Summarization method, which is combined with SPDNet towards SPD matrix based classification. The proposed BoMS addresses the Riemannian codebook learning and Riemannian NNs' optimization issues in the existing approaches, which is based on the idea of summarizing data via a metric learning scheme to compress the whole training data by a predefined feature set. Then, the low-dimensional SPD matrix through Riemannian network is quantized into the predefined matrix summarization bins. Finally, a constant length vector representation is extracted for each SPD matrix by calculating the divergence of data feature and matrix summarization. The proposed method can be integrated into the Riemannian network, and the whole framework can be end-to-end trained via the regular matrix back-propagation. The experiments on three benchmarks demonstrate that the proposed method has outperformed all existing state-of-the-arts in SPD matrix classification. In our future works, we mainly consider to integrate other divergence for Riemannian geometry, such $\alpha-\beta$ divergence, *etc*.

Acknowledge

This work is supported by the National Key R&D Program (No. 2017YFC0113000, and No. 2016YFB1001503), Nature Science Foundation of China (No. U1705262, No. 61772443, and No. 61572410), Post Doctoral Innovative Talent Support Program under Grant BX201600094, China Post-Doctoral Science Foundation under Grant 2017M612134, Scientific Research Project of National Language Committee of China (Grant No. YB135-49), and Nature Science Foundation of Fujian Province, China (No. 2017J01125 and No. 2018J01106).

References

Anirudh, R.; Turaga, P.; Su, J.; and Srivastava, A. 2017. Elastic functional coding of riemannian trajectories. *TPAMI*.

Arsigny, V.; Fillard, P.; Pennec, X.; and Ayache, N. 2006. Log-euclidean metrics for fast and simple calculus on diffusion tensors. *Magnetic resonance in medicine*.

Arsigny, V.; Fillard, P.; Pennec, X.; and Ayache, N. 2007. Geometric means in a novel vector space structure on symmetric positive-definite matrices. *SIAM JMAA*.

Bronstein, M. M.; Bruna, J.; LeCun, Y.; Szlam, A.; and Vandergheynst, P. 2017. Geometric deep learning: going beyond euclidean data. *SPM*.

Cherian, A.; Sra, S.; Banerjee, A.; and Papanikolopoulos, N. 2013. Jensen-bregman logdet divergence with application to efficient similarity search for covariance matrices. *TPAMI*.

Cherian, A.; Stanitsas, P.; Harandi, M.; Morellas, V.; and Papanikolopoulos, N. 2017. Learning discriminative α-β divergences for positive definite matrices. In *ICCV*.

Cherian, A.; Morellas, V.; and Papanikolopoulos, N. 2016. Bayesian nonparametric clustering for positive definite matrices. *TPAMI*.

Fathy, M. E., and Chellappa, R. 2017. Image set classification using sparse bayesian regression. In *WACV*.

Guo, K.; Ishwar, P.; and Konrad, J. 2013. Action recognition from video using feature covariance matrices. *TIP*.

Harandi, M. T.; Sanderson, C.; Hartley, R.; and Lovell, B. C. 2012. Sparse coding and dictionary learning for symmetric positive definite matrices: A kernel approach. In *ECCV*.

Harandi, M. T.; Salzmann, M.; and Baktashmotlagh, M. Beyond gauss: Image-set matching on the riemannian manifold of pdfs. *ICCV*.

Harandi, M.; Salzmann, M.; and Hartley, R. 2017. Dimensionality reduction on spd manifolds: The emergence of geometry-aware methods. *TPAMI*.

Huang, Z., and Van Gool, L. J. 2017. A riemannian network for spd matrix learning. In *AAAI*.

Huang, Z.; Wang, R.; Shan, S.; Li, X.; and Chen, X. 2015. Log-euclidean metric learning on symmetric positive definite manifold with application to image set classification. In *ICML*.

Huang, Z.; Wan, C.; Probst, T.; and Van Gool, L. 2017a. Deep learning on lie groups for skeleton-based action recognition. In *CVPR*.

Huang, Z.; Wang, R.; Li, X.; Liu, W.; Shan, S.; Van Gool, L.; and Chen, X. 2017b. Geometry-aware similarity learning on spd manifolds for visual recognition. *TCSVT*.

Ionescu, C.; Vantzos, O.; and Sminchisescu, C. 2015. Matrix backpropagation for deep networks with structured layers. In *ICCV*.

Kusner, M.; Tyree, S.; Weinberger, K.; and Agrawal, K. 2014. Stochastic neighbor compression. In *ICML*.

Lin, T.-Y., and Maji, S. 2017. Improved bilinear pooling with cnns. In *BMVC*.

Lin, T.-Y.; RoyChowdhury, A.; and Maji, S. 2015. Bilinear cnn models for fine-grained visual recognition. In *ICCV*.

Liu, M.; Shan, S.; Wang, R.; and Chen, X. 2014. Learning expressionlets on spatio-temporal manifold for dynamic facial expression recognition. In *CVPR*.

Lotte, F.; Bougrain, L.; Cichocki, A.; Clerc, M.; Congedo, M.; Rakotomamonjy, A.; and Yger, F. 2018. A review of classification algorithms for eeg-based brain–computer interfaces: a 10 year update. *Journal of neural engineering*.

Nair, V., and Hinton, G. E. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*.

Passalis, N., and Tefas, A. 2017. Learning bag-of-features pooling for deep convolutional neural networks. In *ICCV*.

Pennec, X.; Fillard, P.; and Ayache, N. 2006. A riemannian framework for tensor computing. *IJCV*.

Schirrmeister, R. T.; Springenberg, J. T.; Fiederer, L. D. J.; Glasstetter, M.; Eggensperger, K.; Tangermann, M.; Hutter, F.; Burgard, W.; and Ball, T. 2017. Deep learning with convolutional neural networks for eeg decoding and visualization. *Human brain mapping*.

Sivalingam, R.; Boley, D.; Morellas, V.; and Papanikolopoulos, N. 2015. Tensor dictionary learning for positive definite matrices. *TIP*.

Sra, S. 2012. A new metric on the manifold of kernel matrices with application to matrix geometric means. In *NIPS*.

Sra, S. 2016. Positive definite matrices and the s-divergence. *AMS*.

Wang, R.; Guo, H.; Davis, L. S.; and Dai, Q. 2012. Covariance discriminative learning: A natural and efficient approach to image set classification. In *CVPR*.

Wang, L.; Zhang, J.; Zhou, L.; Tang, C.; and Li, W. 2015a. Beyond covariance: Feature representation with nonlinear kernel matrices. In *ICCV*.

Wang, W.; Wang, R.; Huang, Z.; Shan, S.; and Chen, X. 2015b. Discriminant analysis on riemannian manifold of gaussian distributions for face recognition with image sets. In *CVPR*.

Wang, Q.; Li, P.; and Zhang, L. 2017. G2denet: Global gaussian distribution embedding network and its application to visual recognition. In *CVPR*.

Yger, F., and Sugiyama, M. 2015. Supervised logeuclidean metric learning for symmetric positive definite matrices. *arXiv*.