Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

A Riemannian Network for SPD Matrix Learning

Zhiwu Huang, Luc Van Gool
Computer Vision Lab, ETH Zurich, Switzerland
{zhiwu.huang, vangool}@vision.ee.ethz.ch

Abstract

Symmetric Positive Definite (SPD) matrix learning methods have become popular in many image and video processing tasks, thanks to their ability to learn appropriate statistical representations while respecting Riemannian geometry of underlying SPD manifolds. In this paper we build a Riemannian network architecture to open up a new direction of SPD matrix non-linear learning in a deep model. In particular, we devise bilinear mapping layers to transform input SPD matrices to more desirable SPD matrices, exploit eigenvalue rectification layers to apply a non-linear activation function to the new SPD matrices, and design an eigenvalue logarithm layer to perform Riemannian computing on the resulting SPD matrices for regular output layers. For training the proposed deep network, we exploit a new backpropagation with a variant of stochastic gradient descent on Stiefel manifolds to update the structured connection weights and the involved SPD matrix data. We show through experiments that the proposed SPD matrix network can be simply trained and outperform existing SPD matrix learning and state-of-the-art methods in three typical visual classification tasks.

Introduction

Symmetric Positive Definite (SPD) matrices are often encountered and have made great success in a variety of areas. In medical imaging, they are commonly used in diffusion tensor magnetic resonance imaging (Pennec, Fillard, and Ayache 2006; Arsigny et al. 2007; Jayasumana et al. 2013). In visual recognition, SPD matrix data provide powerful statistical representations for images and videos. Examples include region covariance matrices for pedestrian detection (Tuzel, Porikli, and Meer 2006; 2008; Tosato et al. 2010), joint covariance descriptors for action recognition (Harandi, Salzmann, and Hartley 2014), image set covariance matrices for face recognition (Wang et al. 2012; Huang et al. 2014; 2015b) and second-order pooling for object classification (Ionescu, Vantzos, and Sminchisescu 2015).

As a consequence, there has been a growing need to carry out effective computations to interpolate, restore, and classify SPD matrices. However, the computations on SPD matrices often accompany with the challenge of their non-Euclidean data structure that underlies a Riemannian manifold (Pennec, Fillard, and Ayache 2006; Arsigny et al. 2007). Applying Euclidean geometry to SPD matrices directly often results in undesirable effects, such as the swelling of diffusion tensors (Pennec, Fillard, and Ayache 2006). To address this problem, (Pennec, Fillard, and Ayache 2006; Arsigny et al. 2007; Sra 2011) introduced Riemannian metrics, e.g., Log-Euclidean metric (Arsigny et al. 2007), to encode Riemannian geometry of SPD manifolds properly.

By employing these well-studied Riemannian metrics, existing SPD matrix learning approaches typically flatten SPD manifolds via tangent space approximation (Tuzel, Porikli, and Meer 2008; Tosato et al. 2010; Carreira et al. 2012; Fathy, Alavi, and Chellappa 2016), or map them into reproducing kernel Hilbert spaces (Harandi et al. 2012; Wang et al. 2012; Sanin et al. 2013; Quang, Biagio, and Murino 2014; Faraki, Harandi, and Porikli 2015; Zhang et al. 2015). To more faithfully respect the original Riemannian geometry, recent methods (Harandi, Salzmann, and Hartley 2014; Huang et al. 2015b) adopt a geometry-aware SPD matrix learning scheme to pursue a mapping from the original SPD manifold to another one with the same SPD structure. However, all the existing methods merely apply shallow learning, with which traditional methods are typically surpassed by recent popular deep learning methods in many contexts in artificial intelligence and visual recognition.

In the light of the successful paradigm of deep neural networks (e.g., (LeCun et al. 1998; Krizhevsky, Sutskever, and Hinton 2012)) to perform non-linear computations with effective backpropagation training algorithms, we devise a deep neural network architecture, that receives SPD matrices as inputs and preserves the SPD structure across layers, for SPD matrix non-linear learning. In other words, we aim to design a deep learning architecture to non-linearly learn desirable SPD matrices on Riemannian manifolds. In summary, this paper mainly brings three innovations:

- A novel Riemannian network architecture is introduced to open a new direction of SPD matrix deep non-linear learning on Riemannian manifolds.
- This work offers a paradigm of incorporating the Riemannian structures into deep network architectures for compressing both of the data space and the weight space.
- A new backpropagation is derived to train the proposed network with exploiting a stochastic gradient descent optimization algorithm on Stiefel manifolds.

Related Work

Deep neural networks have exhibited their great powers when the processed data own a Euclidean data structure. In many contexts, however, one may be faced with data defined in non-Euclidean domains. To tackle graph-structured data, (Bruna et al. 2014) presented a spectral formulation of convolutional networks by exploiting a notion of non shift-invariant convolution that depends on the analogy between the classical Fourier transform and the Laplace-Beltrami eigenbasis. Following (Bruna et al. 2014), a localized spectral network was proposed in (Boscaini et al. 2015) to non-Euclidean domains by generalizing the windowed Fourier transform to manifolds to extract the local behavior of some dense intrinsic descriptor. Similarly, (Masci et al. 2015) proposed a 'geodesic convolution' on non-Euclidean local geodesic system of coordinates to extract 'local patches' on shape manifolds. The convolutions in this approach were performed by sliding a window over the shape manifolds.

Stochastic gradient descent (SGD) has been the workhorse for optimizing deep neural networks. As an application of the chain rule, backpropagation is commonly employed to compute Euclidean gradients of objective functions, which is the key operation of SGD. Recently, the two works (Ionescu, Vantzos, and Sminchisescu 2015; Gao, Guo, and Wang 2016) extended backpropagation directly on matrices. For example, (Ionescu, Vantzos, and Sminchisescu 2015) formulated matrix backpropagation as a generalized chain rule mechanism for computing derivatives of composed matrix functions with respect to matrix inputs. Besides, the other family of network optimization algorithms exploits Riemannian gradients to handle weight space symmetries in neural networks. For instance, recent works (Bottou 2010; Bonnabel 2013; Ollivier 2013; Marceau-Caron and Ollivier 2016) developed several optimization algorithms by building Riemannian metrics on the activity and parameter space of neural networks, treated as Riemannian manifolds.

Riemannian SPD Matrix Network

Analogously to the well-known convolutional network (ConvNet), the proposed SPD matrix network (SPDNet) also designs fully connected convolution-like layers and rectified linear units (ReLU)-like layers, named bilinear mapping (BiMap) layers and eigenvalue rectification (ReEig) layers respectively. In particular, following the classical manifold learning theory that learning or even preserving the original data structure can benefit classification, the BiMap layers are designed to transform the input SPD matrices, that are usually covariance matrices derived from the data, to new SPD matrices with a bilinear mapping. As the classical ReLU layers, the proposed ReEig layers introduce a non-linearity to the SPDNet by rectifying the resulting SPD matrices with a non-linear function. Since SPD matrices reside on non-Euclidean manifolds, we have to devise an eigenvalue logarithm (LogEig) layer to carry out Riemannian computing on them to output their Euclidean forms for any regular output layers. The proposed Riemannian network is conceptually illustrated in Fig.1.

BiMap Layer

The primary function of the SPDNet is to generate more compact and discriminative SPD matrices. To this end, we design the BiMap layer to transform the input SPD matrices to new SPD matrices by a bilinear mapping f_b as

$$\boldsymbol{X}_k = f_b^{(k)}(\boldsymbol{X}_{k-1}; \boldsymbol{W}_k) = \boldsymbol{W}_k \boldsymbol{X}_{k-1} \boldsymbol{W}_k^T, \qquad (1)$$

where $\boldsymbol{X}_{k-1} \in Sym^+_{d_{k-1}}$ is the input SPD matrix of the k-th layer, $\boldsymbol{W}_k \in \mathbb{R}_*^{d_k \times d_{k-1}}, (d_k < d_{k-1})$ is the transformation matrix (connection weights), $\boldsymbol{X}_k \in \mathbb{R}^{d_k \times d_k}$ is the resulting matrix. Note that multiple bilinear mappings can be also performed on each input. To ensure the output $\boldsymbol{X}_k$ becomes a valid SPD matrix, the transformation matrix $\boldsymbol{W}_k$ is basically required to be a row full-rank matrix. By applying the BiMap layer, the inputs on the original SPD manifold $Sym^+_{d_{k-1}}$ are transformed to new ones which form another SPD manifold $Sym^+_{d_k}$. In other words, the data space on each BiMap layer corresponds to one SPD manifold.

Since the weight space $\mathbb{R}_*^{d_k \times d_{k-1}}$ of full-rank matrices is a non-compact Stiefel manifold where the distance function has no upper bound, directly optimizing on the manifold is infeasible. To handle this problem, one typical solution is to additionally assume the transformation matrix $\boldsymbol{W}_k$ to be orthogonal (semi-orthogonal more exactly here) so that they reside on a compact Stiefel manifold $St(d_k, d_{k-1})$ [1]. As a result, optimizing over the compact Stiefel manifolds can achieve optimal solutions of the transformation matrices.

ReEig Layer

In the context of ConvNets, (Jarrett et al. 2009; Nair and Hinton 2010) presented various rectified linear units (ReLU) (including the $\max(0, x)$ non-linearity) to improve discriminative performance. Hence, exploiting ReLU-like layers to introduce a non-linearity to the context of the SPDNet is also necessary. Inspired by the idea of the $\max(0, x)$ non-linearity, we devise a non-linear function f_r for the ReEig (k-th) layer to rectify the SPD matrices by tuning up their small positive eigenvalues:

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

where $\boldsymbol{U}_{k-1}$ and $\boldsymbol{\Sigma}_{k-1}$ are achieved by eigenvalue decomposition (EIG) $\boldsymbol{X}_{k-1} = \boldsymbol{U}_{k-1} \boldsymbol{\Sigma}_{k-1} \boldsymbol{U}_{k-1}^T$, ϵ is a rectification threshold, $\boldsymbol{I}$ is an identity matrix, $\max(\epsilon \boldsymbol{I}, \boldsymbol{\Sigma}_{k-1})$ is a diagonal matrix $\boldsymbol{A}$ with diagonal elements being defined as

$$\boldsymbol{A}(i,i) = \begin{cases} \boldsymbol{\Sigma}_{k-1}(i,i), & \boldsymbol{\Sigma}_{k-1}(i,i) > \epsilon, \\ \epsilon, & \boldsymbol{\Sigma}_{k-1}(i,i) \leq \epsilon. \end{cases} \qquad (3)$$

Intuitively, Eqn.2 prevents the input SPD matrices from being close to non-positive ones (while the ReLU yields sparsity). Nevertheless, it is not originally designed for regularization because the inputs are already non-singular after applying BiMap layers. In other words, we always set ϵ above the top-n smallest eigenvalue even when the eigenvalues of original SPD matrices are all much greater than zero.

[1] A compact Stiefel manifold $St(d_k, d_{k-1})$ is the set of d_k-dimensional orthonormal matrices of the $\mathbb{R}^{d_{k-1}}$.

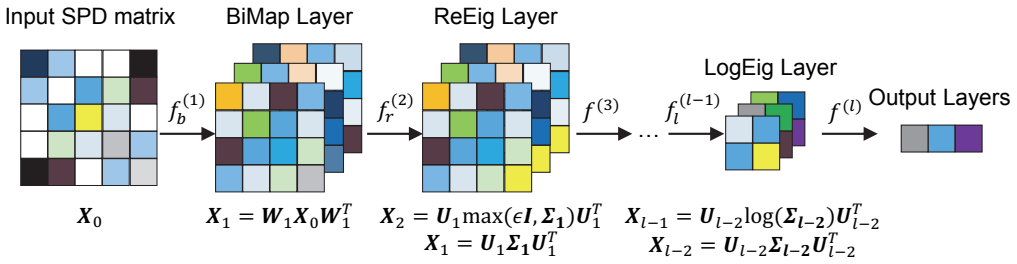

Figure 1: Conceptual illustration of the proposed SPD matrix network (SPDNet) architecture.

Besides, there also exist other feasible strategies to derive a non-linearity on the input SPD matrices. For example, the sigmoidal function (Cybenko 1989) could be considered to extend Eqn.2 to a different activation function. Due to the space limitation, we do not discuss this any further.

LogEig Layer

The LogEig layer is designed to perform Riemannian computing on the resulting SPD matrices for output layers with objective functions. As studied in (Arsigny et al. 2007), the Log-Euclidean Riemannian metric is able to endow the Riemannian manifold of SPD matrices with a Lie group structure so that the manifold is reduced to a flat space with the matrix logarithm operation $\log(\cdot)$ on the SPD matrices. In the flat space, classical Euclidean computations can be applied to the domain of SPD matrix logarithms. Formally, we employ the Riemannian computation (Arsigny et al. 2007) in the k-th layer to define the involved function f_l as

$$\boldsymbol{X}_k = f_l^{(k)}(\boldsymbol{X}_{k-1}) = \log(\boldsymbol{X}_{k-1}) = \boldsymbol{U}_{k-1} \log(\boldsymbol{\Sigma}_{k-1}) \boldsymbol{U}_{k-1}^T, \tag{4}$$

where $\boldsymbol{X}_{k-1} = \boldsymbol{U}_{k-1} \boldsymbol{\Sigma}_{k-1} \boldsymbol{U}_{k-1}^T$ is an EIG operation, $\log(\boldsymbol{\Sigma}_{k-1})$ is the diagonal matrix of eigenvalue logarithms.

For SPD manifolds, the Log-Euclidean Riemannian computation is particularly simple to use and avoids the high expense of other Riemannian computations (Pennec, Fillard, and Ayache 2006; Sra 2011), while preserving favorable theoretical properties. As for other Riemannian computations on SPD manifolds, please refer to (Pennec, Fillard, and Ayache 2006; Sra 2011) for more studies on their properties.

Other Layers

After applying the LogEig layer, the vector forms of the outputs can be fed into classical Euclidean network layers. For example, the Euclidean fully connected (FC) layer could be inserted after the LogEig layer. The dimensionality of the filters in the FC layer is set to $d_k \times d_{k-1}$, where d_k and d_{k-1} are the class number and the dimensionality of the vector forms of the input matrices respectively. The final output layer for visual recognition tasks could be a softmax layer used in the context of Euclidean networks.

In addition, the pooling layers and the normalization layers are also important to improve regular Euclidean ConvNets. For the SPDNet, the pooling on SPD matrices can be first carried out on their matrix logarithms, and then transform them back to SPD matrices by employing the matrix exponential map $\exp(\cdot)$ in the Riemannian framework (Pennec, Fillard, and Ayache 2006; Arsigny et al. 2007). Similarly, the normalization procedure on SPD matrices could be first to calculate the mean and variance of their matrix logarithms, and then normalize them with their mean and variance as done in (Ioffe and Szegedy 2015).

Riemannian Matrix Backpropagation

The model of the proposed SPDNet can be written as a series of successive function compositions $f = f^{(l)} \circ f^{(l-1)} \ldots \circ f^{(1)}$ with a parameter tuple $\boldsymbol{W} = (\boldsymbol{W}_l, \boldsymbol{W}_{l-1}, \ldots, \boldsymbol{W}_1)$, where $f^{(k)}$ is the function for the k-th layer, $\boldsymbol{W}_k$ is the weight parameter of the k-th layer and l is the number of layers. The loss of the k-th layer could be denoted by a function as $L^{(k)} = \ell \circ f^{(l)} \circ \ldots f^{(k)}$, where ℓ is the loss function for the final output layer.

Training deep networks often uses stochastic gradient descent (SGD) algorithms. The key operation of one classical SGD algorithm is to compute the gradient of the objective function, which is obtained by an application of the chain rule known as backpropagation (backprop). For the k-th layer, the gradients of the weight $\boldsymbol{W}_k$ and the data $\boldsymbol{X}_{k-1}$ can be respectively computed by backprop as

$$\frac{\partial L^{(k)}(\boldsymbol{X}_{k-1}, y)}{\partial \boldsymbol{W}_k} = \frac{\partial L^{(k+1)}(\boldsymbol{X}_k, y)}{\partial \boldsymbol{X}_k} \frac{\partial f^{(k)}(\boldsymbol{X}_{k-1})}{\partial \boldsymbol{W}_k}, \tag{5}$$

$$\frac{\partial L^{(k)}(\boldsymbol{X}_{k-1}, y)}{\partial \boldsymbol{X}_{k-1}} = \frac{\partial L^{(k+1)}(\boldsymbol{X}_k, y)}{\partial \boldsymbol{X}_k} \frac{\partial f^{(k)}(\boldsymbol{X}_{k-1})}{\partial \boldsymbol{X}_{k-1}}, \tag{6}$$

where y is the output, $\boldsymbol{X}_k = f^{(k)}(\boldsymbol{X}_{k-1})$. Eqn.5 is the gradient for updating $\boldsymbol{W}_k$, while Eqn.6 is to compute the gradients of the involved data in the layers below. For simplicity, we often replace $\partial L^{(k)}(\boldsymbol{X}_{k-1}, y)$ with $\partial L^{(k)}$ in the sequel.

There exist two key issues for generalizing backprop to the context of the proposed Riemannian network for SPD matrices. The first one is updating the weights in the BiMap layers. As we force the weights to be on Stiefel manifolds, merely using Eqn.5 to compute their Euclidean gradients rather than Riemannian gradients in the procedure of backprop cannot yield valid orthogonal weights. While the gradients of the SPD matrices in the BiMap layers can be calculated by Eqn.6 as usual, computing those with EIG decomposition in the layers of ReEig and LogEig has not been well-solved by the traditional backprop. Thus, it is the second key issue for training the proposed network.

To tackle the first issue, we propose a new way of updating the weights defined in Eqn.1 for the BiMap layers by exploiting an SGD setting on Stiefel manifolds. The steepest descent direction for the corresponding loss function $L^{(k)}(\boldsymbol{X}_{k-1}, y)$ with respect to $\boldsymbol{W}_k$ on the Stiefel manifold is the Riemannian gradient $\tilde{\nabla} L^{(k)}_{\boldsymbol{W}_k}$. To obtain it, the normal component of the Euclidean gradient $\nabla L^{(k)}_{\boldsymbol{W}_k}$ is subtracted to generate the tangential component to the Stiefel manifold. Searching along the tangential direction takes the update in the tangent space of the Stiefel manifold. Then, such the update is mapped back to the Stiefel manifold with a retraction operation. For more details about the Stiefel geometry and retraction, readers are referred to (Edelman, Arias, and Smith 1998) and (Absil, Mahony, and Sepulchre 2008) (Page 45-48, 59). Formally, an update of the current weight $\boldsymbol{W}^t_k$ on the Stiefel manifold respects the form

$$\tilde{\nabla} L^{(k)}_{\boldsymbol{W}^t_k} = \nabla L^{(k)}_{\boldsymbol{W}^t_k} - \nabla L^{(k)}_{\boldsymbol{W}^t_k} (\boldsymbol{W}^t_k)^T \boldsymbol{W}^t_k, \tag{7}$$

$$\boldsymbol{W}^{t+1}_k = \Gamma(\boldsymbol{W}^t_k - \lambda \tilde{\nabla} L^{(k)}_{\boldsymbol{W}^t_k}), \tag{8}$$

where Γ is the retraction operation, λ is the learning rate, $\nabla L^{(k)}_{\boldsymbol{W}_k} (\boldsymbol{W}^t_k)^T \boldsymbol{W}^t_k$ is the normal component of the Euclidean gradient that can be computed by using Eqn.5 as

$$\nabla L^{(k)}_{\boldsymbol{W}^t_k} = 2 \frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k} \boldsymbol{W}^t_k \boldsymbol{X}_{k-1}. \tag{9}$$

As for the second issue, we exploit the matrix generalization of backprop studied in (Ionescu, Vantzos, and Sminchisescu 2015) to compute the gradients of the involved SPD matrices in the ReEig and LogEig layers. In particular, let $\mathcal{F}$ be a function describing the variations of the upper layer variables with respect to the lower layer variables, i.e., $d\boldsymbol{X}_k = \mathcal{F}(d\boldsymbol{X}_{k-1})$. With the function $\mathcal{F}$, a new version of the chain rule Eqn.6 for the matrix backprop is defined as

$$\frac{\partial L^{(k)}(\boldsymbol{X}_{k-1}, y)}{\partial \boldsymbol{X}_{k-1}} = \mathcal{F}^* \left(\frac{\partial L^{(k+1)}(\boldsymbol{X}_k, y)}{\partial \boldsymbol{X}_k} \right), \tag{10}$$

where $\mathcal{F}^*$ is a non-linear adjoint operator of $\mathcal{F}$, i.e., $\boldsymbol{B} : \mathcal{F}(\boldsymbol{C}) = \mathcal{F}^*(\boldsymbol{B}) : \boldsymbol{C}$, the operator : is the matrix inner product with the property $\boldsymbol{B} : \boldsymbol{C} = Tr(\boldsymbol{B}^T \boldsymbol{C})$.

Actually, both of the two functions Eqn.2 and Eqn.4 for the ReEig and LogEig layers involve the EIG operation $\boldsymbol{X}_{k-1} = \boldsymbol{U}_{k-1} \boldsymbol{\Sigma}_{k-1} \boldsymbol{U}^T_{k-1}$ (note that, to increase the readability, we drop the layer indexes for $\boldsymbol{U}_{k-1}$ and $\boldsymbol{\Sigma}_{k-1}$ in the sequel). Hence, we introduce a virtual layer ($k^{'}$ layer) for the EIG operation. Applying the new chain rule Eqn.10 and its properties, the update rule for the data $\boldsymbol{X}_{k-1}$ is derived as

$$\begin{aligned}
&\frac{\partial L^{(k)}}{\partial \boldsymbol{X}_{k-1}} : d\boldsymbol{X}_{k-1} \\
&= \mathcal{F}^* \left(\frac{\partial L^{(k')}}{\partial \boldsymbol{U}} \right) : d\boldsymbol{X}_{k-1} + \mathcal{F}^* \left(\frac{\partial L^{(k')}}{\partial \boldsymbol{\Sigma}} \right) : d\boldsymbol{X}_{k-1} \\
&= \frac{\partial L^{(k')}}{\partial \boldsymbol{U}} : \mathcal{F}\left(d\boldsymbol{X}_{k-1}\right) + \frac{\partial L^{(k')}}{\partial \boldsymbol{\Sigma}} : \mathcal{F}\left(d\boldsymbol{X}_{k-1}\right) \\
&= \frac{\partial L^{(k')}}{\partial \boldsymbol{U}} : d\boldsymbol{U} + \frac{\partial L^{(k')}}{\partial \boldsymbol{\Sigma}} : d\boldsymbol{\Sigma},
\end{aligned} \tag{11}$$

where the two variations $d\boldsymbol{U}$ and $d\boldsymbol{\Sigma}$ are derived by the variation of the EIG operation $d\boldsymbol{X}_{k-1} = d\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^T + \boldsymbol{U}d\boldsymbol{\Sigma}\boldsymbol{U}^T + \boldsymbol{U}\boldsymbol{\Sigma}d\boldsymbol{U}^T$ as:

$$d\boldsymbol{U} = 2\boldsymbol{U}(\boldsymbol{P}^T \circ (\boldsymbol{U}^T d\boldsymbol{X}_{k-1} \boldsymbol{U})_{sym}), \tag{12}$$

$$d\boldsymbol{\Sigma} = (\boldsymbol{U}^T d\boldsymbol{X}_{k-1} \boldsymbol{U})_{diag}, \tag{13}$$

where $\circ$ is the Hadamard product, $\boldsymbol{D}_{sym} = \frac{1}{2}(\boldsymbol{D} + \boldsymbol{D}^T)$, $\boldsymbol{D}_{diag}$ is $\boldsymbol{D}$ with all off-diagonal elements being 0 (note that we also use these two denotations in the following), $\boldsymbol{P}$ is calculated by operating on the eigenvalues σ in $\boldsymbol{\Sigma}$:

$$\boldsymbol{P}(i,j) = \begin{cases} \frac{1}{\sigma_i - \sigma_j}, & i \neq j, \\ 0, & i = j. \end{cases} \tag{14}$$

For more details to derive Eqn.12 and Eqn.13, please refer to (Ionescu, Vantzos, and Sminchisescu 2015). Plugging Eqn.12 and Eqn.13 into Eqn.11 and using the properties of the matrix inner product : can derive the partial derivatives of the loss functions for the ReEig and LogEig layers:

$$\begin{aligned}
\frac{\partial L^{(k)}}{\partial \boldsymbol{X}_{k-1}} = 2\boldsymbol{U} &\left(\boldsymbol{P}^T \circ \left(\boldsymbol{U}^T \frac{\partial L^{(k')}}{\partial \boldsymbol{U}} \right)_{sym} \right) \boldsymbol{U}^T \\
&+ \boldsymbol{U} \left(\frac{\partial L^{(k')}}{\partial \boldsymbol{\Sigma}} \right)_{diag} \boldsymbol{U}^T,
\end{aligned} \tag{15}$$

where $\frac{\partial L^{(k')}}{\partial \boldsymbol{U}}$ and $\frac{\partial L^{(k')}}{\partial \boldsymbol{\Sigma}}$ can be obtained with the same derivation strategy used in Eqn.11. For the function Eqn.2 employed in the ReEig layers, its variation becomes $d\boldsymbol{X}_k = 2(d\boldsymbol{U} \max(\epsilon \boldsymbol{I}, \boldsymbol{\Sigma}) \boldsymbol{U}^T)_{sym} + (\boldsymbol{U}\boldsymbol{Q}d\boldsymbol{\Sigma}\boldsymbol{U}^T)_{sym}$, and these two partial derivatives can be computed by

$$\frac{\partial L^{(k')}}{\partial \boldsymbol{U}} = 2 \left(\frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k} \right)_{sym} \boldsymbol{U} \max(\epsilon \boldsymbol{I}, \boldsymbol{\Sigma}), \tag{16}$$

$$\frac{\partial L^{(k')}}{\partial \boldsymbol{\Sigma}} = \boldsymbol{Q}\boldsymbol{U}^T \left(\frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k} \right)_{sym} \boldsymbol{U}, \tag{17}$$

where $\max(\epsilon \boldsymbol{I}, \boldsymbol{\Sigma})$ is defined in Eqn.3, and $\boldsymbol{Q}$ is the gradient of $\max(\epsilon \boldsymbol{I}, \boldsymbol{\Sigma})$ with diagonal elements being defined as

$$\boldsymbol{Q}(i,i) = \begin{cases} 1, & \boldsymbol{\Sigma}(i,i) > \epsilon, \\ 0, & \boldsymbol{\Sigma}(i,i) \leq \epsilon. \end{cases} \tag{18}$$

For the function Eqn.4 used in the LogEig layers, its variation is $d\boldsymbol{X}_k = 2(d\boldsymbol{U} \log(\boldsymbol{\Sigma}) \boldsymbol{U}^T)_{sym} + (\boldsymbol{U}\boldsymbol{\Sigma}^{-1} d\boldsymbol{\Sigma}\boldsymbol{U}^T)_{sym}$. Then we calculate the following two partial derivatives:

$$\frac{\partial L^{(k')}}{\partial \boldsymbol{U}} = 2 \left(\frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k} \right)_{sym} \boldsymbol{U} \log(\boldsymbol{\Sigma}). \tag{19}$$

$$\frac{\partial L^{(k')}}{\partial \boldsymbol{\Sigma}} = \boldsymbol{\Sigma}^{-1} \boldsymbol{U}^T \left(\frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k} \right)_{sym} \boldsymbol{U}, \tag{20}$$

By mainly employing Eqn.7–Eqn.9 and Eqn.15–Eqn.20, the Riemannian matrix backprop for training the SPDNet can be realized. The convergence analysis of the used SGD algorithm on Riemannian manifolds follows the developments in (Bottou 2010; Bonnabel 2013).

Discussion

Although the two works (Harandi, Salzmann, and Hartley 2014; Huang et al. 2015b) have studied the geometry-aware map to preserve SPD structure, our SPDNet exploits more general (i.e., Stiefel manifold) setting for the map, and sets it up in the context of deep learning. Furthermore, we also introduce a non-linearity during learning SPD matrices in the network. Thus, the main theoretical advantage of the proposed SPDNet over the two works is its ability to perform deep learning and non-linear learning mechanisms.

While (Ionescu, Vantzos, and Sminchisescu 2015) introduced a covariance pooling layer into ConvNets starting from images, our SPDNet works on SPD matrices directly and exploits multiple layers tailored for SPD matrix deep learning. From another point of the view, the proposed SPDNet can be built on the top of the network (Ionescu, Vantzos, and Sminchisescu 2015) for deeper SPD matrix learning that starts from images. Moreover, we must claim that our SPDNet is still useful while (Ionescu, Vantzos, and Sminchisescu 2015) will totally break down when the processed data are not covariance matrices for images.

Experiments

We evaluate the proposed SPDNet for three popular visual classification tasks including emotion recognition, action recognition and face verification, where SPD matrix representations have achieved great successes. The comparing state-of-the-art SPD matrix learning methods are Covariance Discriminative Learning (CDL) (Wang et al. 2012), Log-Euclidean Metric Learning (LEML) (Huang et al. 2015b) and SPD Manifold Learning (SPDML) (Harandi, Salzmann, and Hartley 2014) that uses affine-invariant metric (AIM) (Pennec, Fillard, and Ayache 2006) and stein divergence (Sra 2011). The Riemannian Sparse Representation (RSR) (Harandi et al. 2012) for SPD matrices is also evaluated. Besides, we measure the deep second-order pooling (DeepO2P) network (Ionescu, Vantzos, and Sminchisescu 2015) which introduces a covariance pooling layer into typical ConvNets. For all of them, we use their source codes from authors with tuning their parameters according to the original works. For our SPDNet, we study 4 configurations, i.e., SPDNet-0BiRe/1BiRe/2BiRe/3BiRe, where iBiRe means using i blocks of BiMap/ReEig. For example, the structure of SPDNet-3BiRe is $\boldsymbol{X}_0 \rightarrow f_b^{(1)} \rightarrow f_r^{(2)} \rightarrow f_b^{(3)} \rightarrow f_r^{(4)} \rightarrow f_b^{(5)} \rightarrow f_l^{(6)} \rightarrow f_f^{(7)} \rightarrow f_s^{(8)}$, where f_b, f_r, f_l, f_f, f_s indicate the BiMap, ReEig, LogEig, FC and softmax log-loss layers respectively. The learning rate λ is fixed as 10^{-2}, the batch size is set to 30, the weights are initialized as random semi-orthogonal matrices, and the rectification threshold ϵ is set to 10^{-4}. For training the SPDNet, we just use an i7-2600K (3.40GHz) PC without any GPUs.

Emotion Recognition

We use the popular Acted Facial Expression in Wild (AFEW) (Dhall et al. 2014) dataset for emotion recognition. The AFEW database collects 1,345 videos of facial expressions of actors in movies with close-to-real-world scenarios. The database is divided into training, validation and test data 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 (Liu et al. 2014) to report the results on the validation set. To augment the training data, we segment the training videos into 1,747 small clips. For the evaluation, each facial frame is normalized to an image of size 20×20. Then, following (Wang et al. 2012), we compute the covariance matrix of size 400×400 to represent each video.

On the AFEW database, the dimensionalities of the SPDNet-3BiRe transformation matrices are set to 400×200, $200 \times 100, 100 \times 50$ respectively. Training the SPDNet-3BiRe per epoch (500 epoches in total) takes around 2 minutes(m) on this dataset. As show in Table.1, we report the performances of the competing methods including the state-of-the-art method (STM-ExpLet (Liu et al. 2014)) on this database. It shows our proposed SPDNet-3BiRe achieves several improvements over the state-of-the-art methods although the training data is small.

In addition, we study the behaviors of the proposed SPDNet with different settings. First, we evaluate our SPDNet without using the LogEig layer. Its extremely low accuracy 21.49% shows the layer for Riemannian computing is necessary. Second, we study the case of learning directly on Log-Euclidean forms of original SPD matrices, i.e., SPDNet-0BiRe. The performance of SPDNet-0BiRe is 26.32%, which is clearly outperformed by the deeper SPDNets (e.g., SPDNet-3BiRe). This justifies the importance of using the SPD layers. Besides, SPDNets also clearly outperform DeepO2P that inserts one LogEig-like layer into a standard ConvNet architecture. This somehow validates the improvements are from the contribution of the BiMap and ReEig layers rather than deeper architectures. Third, to study the gains of using multiple BiRe blocks, we compare SPDNet-1BiRe and SPDNet-2BiRe that feed the LogEig layer with SPD matrices of the same size as set in SPDNet-3BiRe. As reported in Table.1, the performance of our SPDNet is improved when stacking more BiRes. Fourth, we also study the power of the designed ReEig layers in Fig.2 (a). The accuracies of the three different rectification threshold settings $\epsilon = 10^{-4}, 5 \times 10^{-5}, 0$ are $34.23\%, 33.15\%, 32.35\%$ respectively, that verifies the success of introducing the non-linearity. Lastly, we also show the convergence behavior of our SPDNet in Fig.2 (b), which suggests it can converge well after hundreds of epoches.

Action Recognition

We handle the problem of skeleton-based human action recognition using the HDM05 database (Müller et al. 2007), which is one of the largest-scale datasets for the problem.

On the HDM05 dataset, we conduct 10 random evaluations, in each of which half of sequences are randomly selected for training, and the rest are used for testing. Note that the work (Harandi, Salzmann, and Hartley 2014) only recognizes 14 motion classes while the protocol designed by us is to identify 130 action classes and thus be more challenging. For data augmentation, the training sequences are divided into around 18,000 small sequences in each random evaluation. As done in (Harandi, Salzmann, and Hartley 2014), we

Method	AFEW	HDM05	PaSC1	PaSC2
STM-ExpLet	31.73%	–	–	–
RSR-SPDML	30.12%	48.01%±3.38	–	–
DeepO2P	28.54%	–	68.76%	60.14%
HERML-DeLF	–	–	58.0%	59.0%
VGGDeepFace	–	–	78.82%	68.24%
CDL	31.81%	41.74%±1.92	78.29%	70.41%
LEML	25.13%	46.87%±2.19	66.53%	58.34%
SPDML-AIM	26.72%	47.25%±2.78	65.47%	59.03%
SPDML-Stein	24.55%	46.21%±2.65	61.63%	56.67%
RSR	27.49%	41.12%±2.53	–	–
SPDNet-0BiRe	26.32%	48.12%±3.15	68.52%	63.92%
SPDNet-1BiRe	29.12%	55.26%±2.37	71.75%	65.81%
SPDNet-2BiRe	31.54%	59.13%±1.78	76.23%	69.64%
SPDNet-3BiRe	**34.23%**	**61.45%±1.12**	**80.12%**	**72.83%**

Table 1: The results for the AFEW, HDM05 and PaSC datasets. PaSC1/PaSC2 are the control/handheld testings.

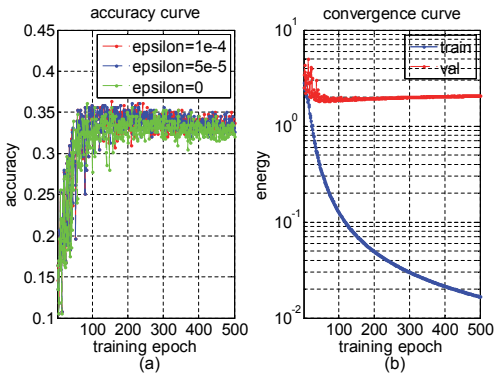

Figure 2: (a) Accuracy curve of the proposed SPDNet-3BiRe at different rectification threshold ϵ values, and (b) its convergence curve at $\epsilon = 10^{-4}$ for the AFEW dataset.

represent each sequence by a joint covariance descriptor of size 93×93, which is computed by the second order statistics of the 3D coordinates of the 31 joints in each frame.

For our SPDNet-3BiRe, the sizes of the transformation matrices are set to $93 \times 70, 70 \times 50, 50 \times 30$ respectively, and its training time at each of 500 epoches is about 4m on average. Table.1 summarizes the results of the comparative algorithms and of the state-of-the-art method (RSR-SPDML) (Harandi, Salzmann, and Hartley 2014) on this dataset. As DeepO2P (Ionescu, Vantzos, and Sminchisescu 2015) is merely for image based visual classification tasks, we do not evaluate it in the 3D skeleton based action recognition task. We find that our SPDNet-3BiRe outperforms the state-of-the-art shallow SPD matrix learning methods by a large margin (more than 13%). This shows that the proposed non-linear deep learning scheme on SPD matrices leads to great improvements when the training data is large enough.

The studies on without using LogEig layers and different configurations of BiRe blocks are executed as the way of the last evaluation. The performance of the case of without using LogEig layers is 4.89%, again validating the importance of the Riemannian computing layers. Besides, as seen from Table.1, the same conclusions as before for different settings of BiRe blocks can be drew on this database.

Face Verification

For face verification, we employ the Point-and-Shoot Challenge (PaSC) database (Beveridge et al. 2013), which is very challenge and widely-used for verifying faces in videos. It includes 1,401 videos taken by control cameras and 1,401 videos captured by handheld cameras for 265 people. In addition, it also contains 280 videos for training.

On the PaSC database, there are control and handheld face verification tasks, both of which are to verify a claimed identity in the query video by comparing with the associated target video. As done in (Beveridge, Zhang, and others 2015), we also use the training data (COX) (Huang et al. 2015a) with 900 videos. Similar to the last two experiments, the whole training data are also augmented to 12,529 small video clips. For evaluation, we use the approach of (Parkhi-and, Vedaldi, and Zisserman 2015) to extract state-of-the-art deep face features on the normalized face images of size 224×224. To speed up the training, we employ PCA to finally get 400-dimensional features. As done in (Huang et al. 2015b), we compute an SPD matrix of size 401×401 for each video to fuse its data covariance matrix and mean.

For the evaluation, we configure the sizes of the SPDNet-3BiRe weights to $401 \times 200, 200 \times 100, 100 \times 50$ respectively. The time for training the SPDNet-3BiRe at each of 100 epoches is around 15m. Table.1 compares the accuracies of the different methods including the state-of-the-art methods (HERML-DeLF (Beveridge, Zhang, and others 2015) and VGGDeepFace (Parkhiand, Vedaldi, and Zisserman 2015)) on the PaSC database. Since the RSR method is designed for recognition tasks rather than verification tasks, we do not report its results. Although the used softmax output layer in our SPDNet is not favorable for the verification tasks, we find that it still achieves the highest performances.

Finally, we can also obtain the same conclusions as before for the studies on different configurations of the proposed SPDNet as observed from the results on the PaSC dataset.

Conclusion

We proposed a novel deep Riemannian network architecture for opening up a possibility of SPD matrix non-linear learning. To train the SPD network, we exploited a new back-propagation with an SGD setting on Stiefel manifolds. The evaluations on three visual classification tasks studied the effectiveness of the proposed network for SPD matrix learning. As future work, we plan to explore more layers, e.g., parallel BiMap layers, pooling layers and normalization layers, to improve the Riemannian network. For further deepening the architecture, we would build the SPD network on the top of existing convolutional networks such as (Ionescu, Vantzos, and Sminchisescu 2015) that start from images. In addition, other interesting directions would be to extend this work for a general Riemannian manifold or to compress traditional network architectures into more compact ones with the proposed SPD or orthogonal constraints.

Acknowledgments

The authors gratefully acknowledge support by EU Framework Seven project ReMeDi (grant 610902).

References

Absil, P.-A.; Mahony, R.; and Sepulchre, R. 2008. *Optimization algorithms on matrix manifolds*. Princeton University Press.

Arsigny, V.; Fillard, P.; Pennec, X.; and Ayache, N. 2007. Geometric means in a novel vector space structure on symmetric positive-definite matrices. *SIAM journal on Matrix Analysis and Applications* 29(1):328–347.

Beveridge, J.; Phillips, J.; Bolme, D.; Draper, B.; Givens, G.; Lui, Y.; Teli, M.; Zhang, H.; Scruggs, W.; Bowyer, K.; Flynn, P.; and Cheng, S. 2013. The challenge of face recognition from digital point-and-shoot cameras. In *BTAS*.

Beveridge, J.; Zhang, H.; et al. 2015. Report on the FG 2015 video person recognition evaluation. In *FG*.

Bonnabel, S. 2013. Stochastic gradient descent on Riemannian manifolds. *IEEE T-AC* 58(9):2217–2229.

Boscaini, D.; Masci, J.; Melzi, S.; Bronstein, M.; Castellani, U.; and Vandergheynst, P. 2015. Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks. In *Computer Graphics Forum*, volume 34, 13–23.

Bottou, L. 2010. Large-scale machine learning with stochastic gradient descent. In *COMPSTAT*.

Bruna, J.; Zaremba, W.; Szlam, A.; and LeCun, Y. 2014. Spectral networks and locally connected networks on graphs. In *ICLR*.

Carreira, J.; Caseiro, R.; Caseiro, J.; and Sminchisescu, C. 2012. Semantic segmentation with second-order pooling. In *ECCV*.

Cybenko, G. 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems* 2(4):303–314.

Dhall, A.; Goecke, R.; Joshi, J.; Sikka, K.; and Gedeon, T. 2014. Emotion recognition in the wild challenge 2014: Baseline, data and protocol. In *ICMI*.

Edelman, A.; Arias, T. A.; and Smith, S. T. 1998. The geometry of algorithms with orthogonality constraints. *SIAM journal on Matrix Analysis and Applications* 20(2):303–353.

Faraki, M.; Harandi, M. T.; and Porikli, F. 2015. Approximate infinite-dimensional region covariance descriptors for image classification. In *ICASSP*.

Fathy, M. E.; Alavi, A.; and Chellappa, R. 2016. Discriminative Log-Euclidean feature learning for sparse representation-based recognition of faces from videos. In *IJCAI*.

Gao, J.; Guo, Y.; and Wang, Z. 2016. Matrix neural networks. *arXiv preprint arXiv:1601.03805v1*.

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 Hartley, R. 2014. From manifold to manifold: Geometry-aware dimensionality reduction for SPD matrices. In *ECCV*.

Huang, Z.; Wang, R.; Shan, S.; and Chen, X. 2014. Learning Euclidean-to-Riemannian metric for point-to-set classification. In *CVPR*.

Huang, Z.; Shan, S.; Wang, R.; Zhang, H.; Lao, S.; Kuerban, A.; and X.Chen. 2015a. A benchmark and comparative study of video-based face recognition on COX face database. *IEEE T-IP* 24(12):5967–5981.

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

Ioffe, S., and Szegedy, C. 2015. Batch Normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

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

Jarrett, K.; Kavukcuoglu, K.; Ranzato, M.; and LeCun, Y. 2009. What is the best multi-stage architecture for object recognition? In *ICCV*.

Jayasumana, S.; Hartley, R.; Salzmann, M.; Li, H.; and Harandi, M. T. 2013. Kernel methods on the Riemannian manifold of symmetric positive definite matrices. In *CVPR*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*.

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

Marceau-Caron, G., and Ollivier, Y. 2016. Practical Riemannian neural networks. *arXiv preprint arXiv:1602.08007*.

Masci, J.; Boscaini, D.; Bronstein, M.; and Vandergheynst, P. 2015. Geodesic convolutional neural networks on Riemannian manifolds. In *ICCV Workshops*.

Müller, M.; Röder, T.; Clausen, M.; Eberhardt, B.; Krüger, B.; and Weber, A. 2007. Documentation: Mocap database HDM05. *Tech. Rep. CG-2007-2*.

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

Ollivier, Y. 2013. Riemannian metrics for neural networks I: Feedforward networks. *arXiv preprint arXiv:1303.0818*.

Parkhiand, O.; Vedaldi, A.; and Zisserman, A. 2015. Deep face recognition. In *BMVC*.

Pennec, X.; Fillard, P.; and Ayache, N. 2006. A Riemannian framework for tensor computing. *IJCV* 66(1):41–66.

Quang, M.; Biagio, M.; and Murino, V. 2014. Log-Hilbert-Schmidt metric between positive definite operators on Hilbert spaces. In *NIPS*.

Sanin, A.; Sanderson, C.; Harandi, M. T.; and Lovell, B. C. 2013. Spatio-temporal covariance descriptors for action and gesture recognition. In *WACV Workshop*.

Sra, S. 2011. Positive definite matrices and the s-divergence. *arXiv preprint arXiv:1110.1773*.

Tosato, D.; Farenzena, M.; Cristani, M.; Spera, M.; and Murino, V. 2010. Multi-class classification on Riemannian manifolds for video surveillance. In *ECCV*.

Tuzel, O.; Porikli, F.; and Meer, P. 2006. Region covariance: A fast descriptor for detection and classification. In *ECCV*.

Tuzel, O.; Porikli, F.; and Meer, P. 2008. Pedestrian detection via classification on Riemannian manifolds. *IEEE T-PAMI* 30(10):1713–1727.

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

Zhang, S.; Kasiviswanathan, S.; Yuen, P. C.; and Harandi, M. T. 2015. Online dictionary learning on symmetric positive definite manifolds with vision applications. In *AAAI*.