The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Building Deep Networks on Grassmann Manifolds

Zhiwu Huang,[†] Jiqing Wu,[†] Luc Van Gool[†‡]
[†]Computer Vision Lab, ETH Zurich, Switzerland [‡]VISICS, KU Leuven, Belgium
{zhiwu.huang, jiqing.wu, vangool}@vision.ee.ethz.ch

Abstract

Learning representations on Grassmann manifolds is popular in quite a few visual recognition tasks. In order to enable deep learning on Grassmann manifolds, this paper proposes a deep network architecture by generalizing the Euclidean network paradigm to Grassmann manifolds. In particular, we design full rank mapping layers to transform input Grassmannian data to more desirable ones, exploit re-orthonormalization layers to normalize the resulting matrices, study projection pooling layers to reduce the model complexity in the Grassmannian context, and devise projection mapping layers to respect Grassmannian geometry and meanwhile achieve Euclidean forms for regular output layers. To train the Grassmann networks, we exploit a stochastic gradient descent setting on manifolds of the connection weights, and study a matrix generalization of backpropagation to update the structured data. The evaluations on three visual recognition tasks show that our Grassmann networks have clear advantages over existing Grassmann learning methods, and achieve results comparable with state-of-the-art approaches.

Introduction

This paper introduces a deep network architecture on Grassmannians, which are manifolds of linear subspaces. In many computer vision applications, linear subspaces have become a core representation. For example, for face verification (Huang et al. 2015b), emotion estimation (Liu et al. 2014b) and activity recognition (Cherian et al. 2017), the image sets of a single person are often modeled by low dimensional subspaces that are then compared on Grassmannians. Besides, for video classification, it is also very common to use autoregressive and moving average (ARMA) model (Vemulapalli, Pillai, and Chellappa 2013). The parameters of the ARMA model are known to be modeled with a high-dimensional linear subspace. For shape analysis, the widely-used affine and linear shape spaces for specific configurations can be also identified by points on the Grassmann manifold (Anirudh et al. 2017). Applications involving dynamic environments and autonomous agents often perform online visual learning by using subspace tracking techniques like incremental principal component analysis (PCA) to dynamically learn a better representational model as the appearance of the moving target (Turaga et al. 2011).

The popular applications of Grassmannian data motivate us to build a deep neural network architecture for Grassmannian representation learning. To this end, the new network architecture is designed to accept Grassmannian data directly as input, and learns new favorable Grassmannian representations that are able to improve the final visual recognition tasks. In other words, the new network aims to deeply learn Grassmannian features on their underlying Riemannian manifolds in an end-to-end learning architecture. In summary, two main contributions are made by this paper:

- We explore a novel deep network architecture in the context of Grassmann manifolds, where it has not been possible to apply deep neural networks. More generally, treating Grassmannian data in deep networks can be very valuable in a variety of machine learning applications.
- We generalize backpropagation to train the proposed network with deriving an connection weight update rule on a specific Riemannian manifold. Furthermore, we incorporate QR decomposition into backpropagation that might prove very useful in other applications since QR decomposition is a very common linear algebra operator.

Background

Grassmannian Geometry

A Grassmann manifold $Gr(q, D)$ is a $q(D-q)$ dimensional compact Riemannian manifold, which is the set of q-dimensional linear subspaces of the $\mathbb{R}^D$. Thus, each point on $Gr(q, D)$ is a linear subspace that is spanned by the related orthonormal basis matrix $\boldsymbol{X}$ of size $D \times q$ such that $\boldsymbol{X}^T\boldsymbol{X} = \boldsymbol{I}_q$, where $\boldsymbol{I}_q$ is the identity matrix of size $q \times q$.

One of the most popular approaches to represent linear subspaces and approximate the true Grassmannian geodesic distance is the projection mapping framework $\Phi(\boldsymbol{X}) = \boldsymbol{X}\boldsymbol{X}^T$ proposed by (Edelman, Arias, and Smith 1998). As the projection $\Phi(\boldsymbol{X})$ is a $D \times D$ symmetric matrix, a natural choice of inner product is $\langle \boldsymbol{X}_1, \boldsymbol{X}_2 \rangle_\Phi = tr(\Phi(\boldsymbol{X}_1)^T \Phi(\boldsymbol{X}_2))$. The inner product induces a distance metric named projection metric:

$$d_p(\boldsymbol{X}_1, \boldsymbol{X}_2) = 2^{-1/2} \|\boldsymbol{X}_1\boldsymbol{X}_1^T - \boldsymbol{X}_2\boldsymbol{X}_2^T\|_F. \qquad (1)$$

where $\|\cdot\|_F$ indicates the matrix Frobenius norm. As proved in (Harandi et al. 2013), the projection metric can approximate the true geodesic distance up to a scale of $\sqrt{2}$.

Grassmann Learning

To perform discriminant learning on Grassmann manifolds, many works (Hamm and Lee. 2008; Hamm and Lee 2009; Cetingul and Vidal 2009; Harandi et al. 2011; 2013; 2014) either adopt tangent space approximation of the underlying manifolds, or exploit positive definite kernel functions to embed the manifolds into reproducing kernel Hilbert spaces. In both of such two cases, any existing Euclidean techniques can then be applied to the embedded data, since Hilbert spaces respect Euclidean geometry as well. For example, (Hamm and Lee. 2008) first embeds the Grassmannian into a high dimensional Hilbert space, and then applies traditional Fisher analysis methods. Obviously, most of these methods are limited to the Mercer kernels, and hence are restricted to use only kernel-based classifiers. Moreover, their computational complexity increases steeply with the growing number of training samples.

More recently, a new learning scheme was proposed by (Huang et al. 2015b) to perform a geometry-aware dimensionality reduction from the original Grassmann manifold to another lower-dimensional, more discriminative Grassmann manifold. This could better preserve the original Riemannian data structure, which commonly leads to more favorable classification performances as studied in classical manifold learning. While (Huang et al. 2015b) has reached some success, it merely adopts a shallow learning scheme on Grassmann manifolds, which is still far away from the best solution for the problem of representation learning on nonlinear manifolds. Accordingly, this paper attempts to open up a possibility of deep learning on Grassmannians.

Manifold Network

By leveraging the paradigm of traditional neural networks, an increasing number of networks (Masci et al. 2015; Ionescu, Vantzos, and Sminchisescu 2015; Huang and Van Gool 2017) have been built over general manifold domains. For instance, (Masci et al. 2015) proposed a 'geodesic convolution' on local geodesic coordinate systems to extract local patches on the shape manifold for shape analysis. In particular, the method implements convolutions by sliding a window over the shape manifold, and local geodesic coordinates are employed instead of image patches. In (Huang and Van Gool 2017), to deeply learn appropriate features on the manifolds of symmetric positive definite (SPD) matrices, a deep network structure was developed with some spectral layers, which can be trained by a variant of backpropagation. Nevertheless, to the best of our knowledge, this is the first work that studies a deep network architecture on Grassmann manifolds.

Grassmann Network Architecture

In analogy to convolutional networks (ConvNets), the proposed Grassmann network (GrNet) also devises a Projection block containing fully connected convolution-like layers and normalization layers, named full rank mapping (FRMap) layers and re-orthonormalization (ReOrth) layers respectively. Inspired by the geometry-aware manifold learning idea (Huang et al. 2015b), the FRMap layers are proposed to firstly perform transformations on input orthonormal matrices of subspaces to generate new matrices by adopting a full rank mapping scheme. Then, the ReOrth layers are developed to normalize the output matrices of the FRMap layers so that they can keep the basic orthogonality. In other words, the normalized data become orthonormal matrices that reside on Stiefel manifold[1]. As well-studied in (Edelman, Arias, and Smith 1998; Hamm and Lee. 2008), the projection metric performing on orthonormal matrices can represent linear subspaces and respect the geometry of Grassmann manifolds, which is actually a quotient manifold of the Stiefel manifold. Accordingly, we develop projection mapping (ProjMap) layers to maintain the Grassmannian property of the resulting data. Meanwhile, the ProjMap layers are able to transfer the resulting Grassmannian data into Euclidean data, which enables the regular Euclidean layers such as softmax layers for classification. The ProjMap and softmax layers forms the Output block for GrNet. Additionally, since traditional pooling layers can reduce the network complexity, we also study projection pooling (ProjPooling) layers on the projection matrix form of the resulting orthonormal matrices. As it is non-trivial to perform pooling on non-Euclidean data directly, we develop a Pooling block to combine ProMap, ProjPooling and orthonormal mapping (OrthMap) layers, which respectively achieves Euclidean representation, performs pooling on resulting Euclidean data and transforms the results back to orthonormal data. The proposed GrNet structure is illustrated in Fig.1.

FRMap Layer

To learn compact and discriminative Grassmannian representation for better classification, we design the FRMap layers to first transform the input orthonormal matrices of subspaces to new matrices by a linear mapping function f_{fr} as

$$\boldsymbol{X}_k = f_{fr}^{(k)}(\boldsymbol{X}_{k-1}; \boldsymbol{W}_k) = \boldsymbol{W}_k \boldsymbol{X}_{k-1}, \tag{2}$$

where $\boldsymbol{X}_{k-1} \in Gr(q, d_{k-1})$ is the input of the k-th layer[2], $\boldsymbol{W}_k \in \mathbb{R}_*^{d_k \times d_{k-1}}, (d_k < d_{k-1})$ is the transformation matrix (connection weights) that is basically required to be a row full rank matrix, $\boldsymbol{X}_k \in \mathbb{R}^{d_k \times q}$ is the resulting matrix. Generally, the transformation result $\boldsymbol{W}_k \boldsymbol{X}_{k-1}$ is not an orthonormal basis matrix. To tackle this problem, we exploit a normalization strategy of QR decomposition in the following ReOrth layer. In addition, as classical deep networks, multiple projections $\{\boldsymbol{W}_k^1, \ldots, \boldsymbol{W}_k^m\}$ per FRMap layer can be applied on each input orthonormal matrix as well, where m is the number of transformation matrices.

As the weight space $\mathbb{R}_*^{d_k \times d_{k-1}}$ of full rank matrices on each FRMap layer is a non-compact Stiefel manifold where geodesic distance has no closed form. In view of gradient-steepest-descent learning by geodesic stepping for criterion optimization, it is necessary to have a closed form of the geodesic distance to derive natural gradients over a smooth

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

[2]For consistency, k is used to denote the relative index of each involved layer in the sequel.

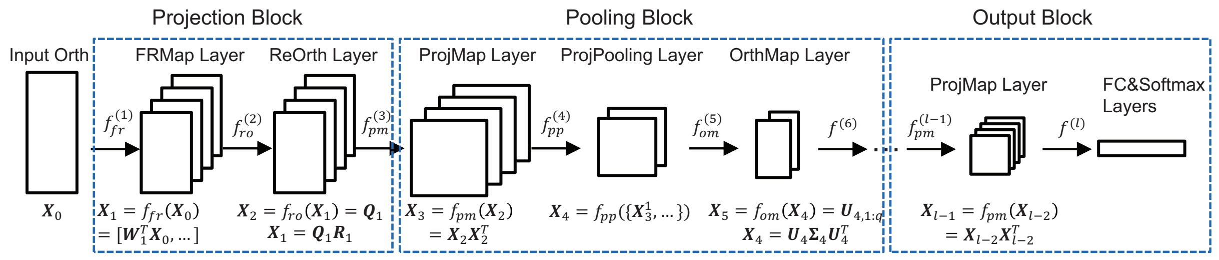
Projection Block
Pooling Block
Output Block
Input Orth
FRMap Layer
ReOrth Layer
ProjMap Layer
ProjPooling Layer
OrthMap Layer
ProjMap Layer
FC&Softmax Layers
$f_{fr}^{(1)}$
$f_{ro}^{(2)}$
$f_{pm}^{(3)}$
$f_{pp}^{(4)}$
$f_{om}^{(5)}$
$f^{(6)}$
$f_{pm}^{(l-1)}$
$f^{(l)}$
$\boldsymbol{X}_0$
$\boldsymbol{X}_1 = f_{fr}(\boldsymbol{X}_0) = [\boldsymbol{W}_1^T\boldsymbol{X}_0, \dots]$
$\boldsymbol{X}_2 = f_{ro}(\boldsymbol{X}_1) = \boldsymbol{Q}_1$
$\boldsymbol{X}_1 = \boldsymbol{Q}_1\boldsymbol{R}_1$
$\boldsymbol{X}_3 = f_{pm}(\boldsymbol{X}_2) = \boldsymbol{X}_2\boldsymbol{X}_2^T$
$\boldsymbol{X}_4 = f_{pp}(\{\boldsymbol{X}_3^1, \dots\})$
$\boldsymbol{X}_5 = f_{om}(\boldsymbol{X}_4) = \boldsymbol{U}_{4,1:q}$
$\boldsymbol{X}_4 = \boldsymbol{U}_4\boldsymbol{\Sigma}_4\boldsymbol{U}_4^T$
$\boldsymbol{X}_{l-1} = f_{pm}(\boldsymbol{X}_{l-2}) = \boldsymbol{X}_{l-2}\boldsymbol{X}_{l-2}^T$

Figure 1: Conceptual illustration of the proposed Grassmann Network (GrNet) architecture. The rectangles in blue represent three basic blocks, i.e., Projection, Pooling and Output blocks, respectively.

manifold. Hence, optimizing on the non-compact manifold directly is infeasible unless endowing the manifold with pseudo-Riemannian metrics. To handle this problem, one feasible solution is imposing orthogonality constraint on the weight matrix $\boldsymbol{W}_k$ so that it resides on a compact Stiefel manifold $St(d_k, d_{k-1})$. Obviously, such orthogonal solution space is smaller than the original solution space $\mathbb{R}_*^{n\times m}$, making the optimization theoretically yield suboptimal solution of $\boldsymbol{W}_k$. To achieve a more faithful solution, following (Huang et al. 2015b), we alternatively do the optimization over the manifolds $PSD(d_k, d_{k-1})$[3] of the conjugates $\boldsymbol{P}_k = \boldsymbol{W}_k^T\boldsymbol{W}_k$, which are actually positive semidefinite (PSD) matrices. As studied in (Bonnabel and Sepulchre 2009; Journee et al. 2010; Meyer, Bonnabel, and Sepulchre 2011) and the popular manopt toolbox[4], a PSD manifold is actually a quotient space, and thus optimizing on it actually pursues optimal full rank matrix $\boldsymbol{W}_k$ directly. In the sense, optimizing on PSD manifolds actually minimizes $f(\boldsymbol{W}_k)$ instead of $f(\boldsymbol{W}_k^T\boldsymbol{W}_k)$, where f denotes the involved layer's loss function that will be introduced in the next section.

ReOrth Layer

Inspired by (Kim, Kittler, and Cipolla 2007; Huang et al. 2015b) that use QR decomposition to transform a regular matrix to an orthonormal matrix, we design the ReOrth layers to perform QR decomposition on the input matrix $\boldsymbol{X}_{k-1}$

$$\boldsymbol{X}_{k-1} = \boldsymbol{Q}_{k-1}\boldsymbol{R}_{k-1}, \tag{3}$$

where $\boldsymbol{Q}_{k-1} \in \mathbb{R}^{d_{k-1}\times q}$ is the orthonormal matrix composed by the first q columns, and $\boldsymbol{R}_{k-1} \in \mathbb{R}^{q\times q}$ is the invertible upper-triangular matrix. Since $\boldsymbol{R}$ is invertible and $\boldsymbol{Q}$ is orthonormal, we can make $\boldsymbol{X}_k$ become an orthonormal basis matrix by normalizing $\boldsymbol{X}_{k-1}$ in the k-th layer as:

$$\boldsymbol{X}_k = f_{ro}^{(k)}(\boldsymbol{X}_{k-1}) = \boldsymbol{X}_{k-1}\boldsymbol{R}_{k-1}^{-1} = \boldsymbol{Q}_{k-1}. \tag{4}$$

In the context of ConvNets, (Cybenko 1989; Jarrett et al. 2009; Nair and Hinton 2010; Goodfellow et al. 2013; He et al. 2015) have presented various nonlinear activation functions, e.g., rectified linear units (ReLU), to improve discriminative performance. Accordingly, exploiting this kind of layers to introduce the non-linearity to the domain of the proposed GrNet is also necessary. In the light of this, to some extent, the function Eqn.4 also takes a role of performing a non-linear activation with the QR factorization.

ProjMap Layer

The ProjMap layer is designed to perform Grassmannian computation on the resulting orthonormal matrices. As well-studied in (Edelman, Arias, and Smith 1998; Hamm and Lee. 2008; Hamm and Lee 2009; Harandi et al. 2011; Huang et al. 2015b), the projection metric is one of the most popular Grassmannian metrics, and is able to endow the specific Riemannian manifold with an inner product structure so that the manifold is reduced to a flat space. In the flat space, classical Euclidean computations can be applied to the projection domain of orthonormal matrices. Formally, we apply the projection mapping (Edelman, Arias, and Smith 1998) to a orthonormal matrix $\boldsymbol{X}_{k-1}$ in the k-th layer as

$$\boldsymbol{X}_k = f_{pm}^{(k)}(\boldsymbol{X}_{k-1}) = \boldsymbol{X}_{k-1}\boldsymbol{X}_{k-1}^T. \tag{5}$$

As for other Riemannian computations on Grassmann manifolds, please refer to (Le 1991; Edelman, Arias, and Smith 1998; Srivastava and Klassen 2004; Absil, Mahony, and Sepulchre 2009; Helmke, Hüper, and Trumpf 2007).

ProjPooling Layer

It is known that classical pooling layers with max, min and mean pooling functions reduce the sizes of the representations to lower the model complexity, and therefore improve the regular ConvNets. Motivated by this, we also design pooling layers for the feature maps of Grassmannian data.

Without loss of generality, we here study mean pooling for the Grassmannian points. Actually, there exist some approaches (Absil, Mahony, and Sepulchre 2004; Dodge and Rousson 1999; Srivastava and Klassen 2002; Marrinan et al. 2014) to compute mean points on Grassmannians. Inspired by the idea (Srivastava and Klassen 2002) with keeping the balance between computational time and calculation accuracy, we propose three layers to implement mean pooling for Grassmannian data. In particular, the Grassmannian data are first mapped to the space of projection matrices by the ProjMap layer presented before. As the resulting m projection matrices $\{\boldsymbol{X}_{k-1}^i | 1 \le i \le m\}$ of size $d_{k-1} \times d_{k-1}$ are Euclidean, we then design a regular mean pooling layer for

[3] A PSD manifold $PSD(d_k, d_{k-1})$ is the set of d_k-rank positive semidefinite matrices of size d_{k-1} (Bonnabel and Sepulchre 2009; Journee et al. 2010).

[4] http://www.manopt.org/

them. Lastly, we devise an orthonormal mapping (OrthMap) layer to transform the mean projection matrix back to orthonormal data. Formally, the functions for the ProjPooling and OrthMap layers are respectively defined as

$$\boldsymbol{X}_k = f_{pp}^{(k)}(\{\hat{\boldsymbol{X}}_{k-1}^1, \ldots, \hat{\boldsymbol{X}}_{k-1}^n\}) = \frac{1}{n}\sum_i^n \hat{\boldsymbol{X}}_{k-1}^i, \quad (6)$$

$$\boldsymbol{X}_{k+1} = f_{om}^{(k)}(\boldsymbol{X}_k) = \boldsymbol{U}_{k-1,1:q}. \quad (7)$$

where $\boldsymbol{U}_{k-1,1:q}$ is the first q largest eigenvectors achieved by eigenvalue (EIG) decomposition on the input projection matrices $\boldsymbol{X}_k = \boldsymbol{U}_{k-1}\boldsymbol{\Sigma}_{k-1}\boldsymbol{U}_{k-1}^T$, and n is the number of instances $\hat{\boldsymbol{X}}_{k-1}^i$ for the pooling. The instances can be either n projection matrices or n entries within an square patch of size $\sqrt{n} \times \sqrt{n}$ located in one projection matrix. In other words, the first type of pooling is performed across the projection matrices (A-ProjPooling), while the second one is executed by sliding the mean filter over each square patch within one projection matrix (W-ProjPooling). As a result, A-ProjPooling with OrthMap finally yields $\frac{m}{n}$ orthonormal matrices of size $d_{k-1} \times q$, while W-ProjPooling with OrthMap outputs m orthonormal matrices of size $\frac{d_{k-1}}{\sqrt{n}} \times q$.

Output Layers

As shown in Fig.1, after applying the ProjMap layer, the outputs (i.e., the projection matrices) lie in Euclidean space, and thus can be converted into vector forms. Hence, on the top of the ProjMap layer, classical Euclidean network layers can be employed. For instance, the regular fully connected (FC) layer could be used after the ProjMap layer. The dimensionality of the filters in the FC layer is typically set to $d_k \times d_{k-1}$, where d_k and d_{k-1} are the class number and the dimensionality of the input vector forms respectively. Finally, the common softmax layer can be used for visual classification.

Training Grassmann Network

As most layers in the GrNet model are expressed with complex matrix factorization functions, they cannot be simply reduced to a constructed bottom-up from element-wise calculations. In other words, the matrix backpropgation (backprop) cannot be derived by using traditional matrix that treats element-wise operations in matrix form. As a result, simply using the traditional backprop will break down in the setting. To solve the problem, (Huang and Van Gool 2017; Ionescu, Vantzos, and Sminchisescu 2015) introduced manifold-valued connection weight update rule and matrix backprop respectively. Furthermore, the convergence of the stochastic gradient descent (SGD) algorithm on Riemannian manifolds has also been studied well in (Bottou 2010; Bonnabel 2013). Accordingly, we exploit the training procedure for the proposed GrNet upon these existing works.

To begin with, we represent the proposed GrNet model with a sequence of successive function compositions $f = f^{(l)} \circ f^{(l-1)} \circ f^{(l-2)} \ldots \circ f^{(2)} \circ f^{(1)}$ with a parameter tuple $\boldsymbol{W} = (\boldsymbol{W}_l, \boldsymbol{W}_{l-1}, \ldots, \boldsymbol{W}_1)$, where $f^{(k)}$ and $\boldsymbol{W}_k$ are the function and the weight matrix respectively for the k-th layer, and l is the number of layers. The loss of the k-th layer can be denoted by a function as $L^{(k)} = \ell \circ f^{(l)} \circ \ldots f^{(k)}$, where ℓ is the loss function for the last output layer.

Then, we recall the definition of the matrix backprop and its properties studied in (Ionescu, Vantzos, and Sminchisescu 2015). In particular, (Ionescu, Vantzos, and Sminchisescu 2015) exploits a function $\mathcal{F}$ to describe the variations of the upper layer variables with respect to the lower layer variables, i.e., $d\boldsymbol{X}_k = \mathcal{F}(d\boldsymbol{X}_{k-1})$. Consequently, a new version of the chain rule for the matrix backprop is defined as

$$\frac{\partial L^{(k)}(\boldsymbol{X}_{k-1}, y)}{\partial \boldsymbol{X}_{k-1}} = \mathcal{F}^*\left(\frac{\partial L^{(k+1)}(\boldsymbol{X}_k, y)}{\partial \boldsymbol{X}_k}\right), \quad (8)$$

where y is the desired output, $\boldsymbol{X}_k = f^{(k)}(\boldsymbol{X}_{k-1})$, $\mathcal{F}^*$ is a non-linear adjoint operator of $\mathcal{F}$, i.e., $a : \mathcal{F}(b) = \mathcal{F}^*(a) : b$, the matrix inner product $\boldsymbol{A} : \boldsymbol{B} = Tr(\boldsymbol{A}^T\boldsymbol{B})$.

In the sequel, we will detail the connection weight update on the specific PSD manifold and the matrix backprop process through some key layers in the context of the proposed GrNet. For simplicity, we uniformly let $\partial L^{(k)}(\boldsymbol{X}_{k-1}, y)$ be $\partial L^{(k)}$, $\boldsymbol{Q}_{k-1}$ be $\boldsymbol{Q}$, and $\boldsymbol{R}_{k-1}$ be $\boldsymbol{R}$ respectively.

FRMap Layer

For the FRMap layers, we propose a new way of updating the weights appeared in Eqn.2 by exploiting an SGD setting on PSD manifolds. As studied in (Absil, Mahony, and Sepulchre 2009), the steepest descent direction for the corresponding loss function $L^{(k)}(\boldsymbol{X}_{k-1}, y)$ with respect to $\boldsymbol{W}_k$ on one Riemannian manifold is the Riemannian gradient $\tilde{\nabla} L_{\boldsymbol{W}_k}^{(k)}$. In particular, following the standard optimization (Absil, Mahony, and Sepulchre 2009) on Riemannian manifolds, we first apply the parallel transport to transfer the Euclidean gradient in the tangent space at the current status of the weight $\boldsymbol{W}_k^t$ to the one in the tangent space at the next status $\boldsymbol{W}_k^{t+1}$. Then the resulting Euclidean gradient is subtracted to the normal component of the Euclidean gradient $\nabla L_{\boldsymbol{W}_k^t}^{(k)}$. After this operation, searching along the tangential direction yields the update in the tangent space of the PSD manifold. Finally, the resulting update is mapped back to the PSD manifold with a retraction operation Γ. For more details about the geometry of PSD manifolds and the retraction operation on Riemannian manifolds, the readers are referred to (Bonnabel and Sepulchre 2009; Journee et al. 2010; Meyer, Bonnabel, and Sepulchre 2011; Absil, Mahony, and Sepulchre 2009). Accordingly, the update of the current connection weight $\boldsymbol{W}_k^t$ on the PSD manifold adheres to the following form

$$\tilde{\nabla} L_{\boldsymbol{W}_k^t}^{(k)} = \nabla L_{\boldsymbol{W}_k^t}^{(k)} - \nabla L_{\boldsymbol{W}_k^t}^{(k)} (\boldsymbol{W}_k^t)^T \boldsymbol{W}_k^t, \quad (9)$$

$$\boldsymbol{W}_k^{t+1} = \Gamma(\boldsymbol{W}_k^t - \lambda \tilde{\nabla} L_{\boldsymbol{W}_k^t}^{(k)}), \quad (10)$$

where λ is the learning rate, $\nabla L_{\boldsymbol{W}_k^t}^{(k)} (\boldsymbol{W}_k^t)^T \boldsymbol{W}_k^t$ is the normal component of the Euclidean gradient $\nabla L_{\boldsymbol{W}_k^t}^{(k)}$. By employing the conventional backprop, $\nabla L_{\boldsymbol{W}_k^t}^{(k)}$ is computed by

$$\nabla L_{\boldsymbol{W}_k^t}^{(k)} = \frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k} \frac{\partial f^{(k)}(\boldsymbol{X}_{k-1})}{\partial \boldsymbol{W}_k^t} = \frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k} \boldsymbol{X}_{k-1}^T. \quad (11)$$

ReOrth Layer

Actually, the ReOrth layers involve QR decomposition Eqn.3 and the non-linear operation Eqn.4. Firstly, for Eqn.3 we introduce a virtual layer $k^{'}$, which receives $\boldsymbol{X}_{k-1}$ as input and produces a tuple $\boldsymbol{X}_{k'} = (\boldsymbol{Q}, \boldsymbol{R})$. Following (Ionescu, Vantzos, and Sminchisescu 2015) to handle the case of a tuple output, we apply the new chain rule Eqn.8 with the equations $a : \mathcal{F}(b) = \mathcal{F}^*(a) : b$ and $d\boldsymbol{X}_{k'} = \mathcal{F}(d\boldsymbol{X}_{k-1})$ to achieve the update rule for the structured data:

$$\begin{aligned}&\frac{\partial L^{(k)}}{\partial \boldsymbol{X}_{k-1}} : d\boldsymbol{X}_{k-1}\\ &= \mathcal{F}^*\left(\frac{\partial L^{(k^{'})}}{\partial \boldsymbol{Q}}\right) : d\boldsymbol{X}_{k-1} + \mathcal{F}^*\left(\frac{\partial L^{(k^{'})}}{\partial \boldsymbol{R}}\right) : d\boldsymbol{X}_{k-1}\\ &= \frac{\partial L^{(k^{'})}}{\partial \boldsymbol{Q}} : \mathcal{F}\left(d\boldsymbol{X}_{k-1}\right) + \frac{\partial L^{(k^{'})}}{\partial \boldsymbol{R}} : \mathcal{F}\left(d\boldsymbol{X}_{k-1}\right)\\ &= \frac{\partial L^{(k^{'})}}{\partial \boldsymbol{Q}} : d\boldsymbol{Q} + \frac{\partial L^{(k^{'})}}{\partial \boldsymbol{R}} : d\boldsymbol{R},\end{aligned} \tag{12}$$

where the two variations $d\boldsymbol{Q}$ and $d\boldsymbol{R}$ are derived by the variation of the QR operation $d\boldsymbol{X}_{k-1} = d\boldsymbol{Q}\boldsymbol{R} + \boldsymbol{Q}d\boldsymbol{R}$ as:

$$d\boldsymbol{Q} = \boldsymbol{S}d\boldsymbol{X}_{k-1}\boldsymbol{R}^{-1} + \boldsymbol{Q}(\boldsymbol{Q}^T d\boldsymbol{X}_{k-1}\boldsymbol{R}^{-1})_{asym}, \tag{13}$$

$$d\boldsymbol{R} = \boldsymbol{Q}^T d\boldsymbol{X}_{k-1} - (\boldsymbol{Q}^T d\boldsymbol{X}_{k-1}\boldsymbol{R}^{-1})_{asym}\boldsymbol{R}, \tag{14}$$

where $\boldsymbol{S} = \boldsymbol{I} - \boldsymbol{Q}\boldsymbol{Q}^T$, $\boldsymbol{I}$ is an identity matrix, $\boldsymbol{A}_{asym} = \boldsymbol{A}_{tril} - (\boldsymbol{A}_{tril})^T$, $\boldsymbol{A}_{tril}$ extracts the elements below the main diagonal of $\boldsymbol{A}$. For more details to derive Eqn.13 and Eqn.14, please refer to the part I of Appendix.

As derived by the part II of Appendix, plugging Eqn.13 and Eqn.14 into Eqn.12 achieves the partial derivatives of the loss functions for the ReOrth layers:

$$\begin{aligned}\frac{\partial L^{(k)}}{\partial \boldsymbol{X}_{k-1}} = &\left(\boldsymbol{S}^T \frac{\partial L^{(k^{'})}}{\partial \boldsymbol{Q}} + \boldsymbol{Q}\left(\boldsymbol{Q}^T \frac{\partial L^{(k^{'})}}{\partial \boldsymbol{Q}}\right)_{bsym}\right)(\boldsymbol{R}^{-1})^T\\ &+ \boldsymbol{Q}\left(\frac{\partial L^{(k^{'})}}{\partial \boldsymbol{R}} - \left(\frac{\partial L^{(k^{'})}}{\partial \boldsymbol{R}}\boldsymbol{R}^T\right)_{bsym}(\boldsymbol{R}^{-1})^T\right),\end{aligned} \tag{15}$$

where $\boldsymbol{A}_{bsym} = \boldsymbol{A}_{tril} - (\boldsymbol{A}^T)_{tril}$, $\boldsymbol{A}_{tril}$ extracts the elements below the main diagonal of $\boldsymbol{A}$. $\frac{\partial L^{(k^{'})}}{\partial \boldsymbol{Q}}$ and $\frac{\partial L^{(k^{'})}}{\partial \boldsymbol{R}}$ can then be obtained on the function Eqn.4 employed in the ReOrth layers. Specially, its variation becomes $d\boldsymbol{X}_k = d\boldsymbol{Q}$. Therefore, the involved partial derivatives with respect to $\boldsymbol{Q}$ and $\boldsymbol{R}$ are computed by $\frac{\partial L^{(k^{'})}}{\partial \boldsymbol{Q}} = \frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k}$ and $\frac{\partial L^{(k^{'})}}{\partial \boldsymbol{R}} = 0$.

OrthMap Layer

As presented before, the OrthMap layers involve eigenvalue (EIG) decomposition. Thus we adopt the proposition in (Ionescu, Vantzos, and Sminchisescu 2015) to calculate the partial derivatives for the EIG computation.

Proposition 1 *Let $\boldsymbol{X}_{k-1} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^T$ with $\boldsymbol{X}_{k-1} \in \mathbb{R}^{D\times D}$, such that $\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{I}$ and $\boldsymbol{\Sigma}$ owns a diagonal structure. The resulting partial derivative for the EIG layer $k^{'}$ is given by*

$$\begin{aligned}\frac{\partial L^{(k)}}{\partial \boldsymbol{X}_{k-1}} = &\boldsymbol{U}\left(\hat{\boldsymbol{K}}^T \circ \left(\boldsymbol{U}^T \frac{\partial L^{(k^{'})}}{\partial \boldsymbol{U}}\right)\right)\boldsymbol{U}^T\\ &+ \boldsymbol{U}\left(\frac{\partial L^{(k^{'})}}{\partial \boldsymbol{\Sigma}}\right)_{diag}\boldsymbol{U}^T.\end{aligned} \tag{16}$$

where $\hat{\boldsymbol{K}} = 1/(\sigma_i - \sigma_j), i \neq j; 0, i = j$ (σ_i is the diagonal element of $\boldsymbol{\Sigma}$), and the partial derivatives with respect to $\boldsymbol{\Sigma}$ and $\boldsymbol{U}$ in Eqn.7 for the OrthMap layers can be achieved by $\frac{\partial L^{(k^{'})}}{\partial \boldsymbol{U}} = [\frac{\partial L^{(k+1)}}{\partial \boldsymbol{X}_k} \quad \boldsymbol{0}]$ and $\frac{\partial L^{(k^{'})}}{\partial \boldsymbol{\Sigma}} = 0$, where $\boldsymbol{0}$ is the matrix of size $D \times (D - q)$ with all elements being zero.

Empirical Evaluation

We compare four groups of exiting methods to evaluate the proposed GrNet for three visual classification tasks: emotion recognition, action recognition and face verification.

Comparing methods: **1).** *General manifold learning methods:* Expressionlets on Spatio-Temporal Manifold (STM-ExpLet) (Liu et al. 2014a) and Riemannian Sparse Representaion combining with Manifold Learning on the manifold of SPD matrices (RSR-SPDML) (Harandi, Salzmann, and Hartley 2014); **2).** *Grassmann learning methods*: Discriminative Canonical Correlations (DCC) (Kim, Kittler, and Cipolla 2007), Grassmann Discriminant Analysis (GDA) (Hamm and Lee. 2008), Grassmannian Graph-Embedding Discriminant Analysis (GGDA) (Hamm and Lee 2009) and Projection Metric Learning (PML) (Huang et al. 2015b); **3).** *Regular convolutional networks*: VGGDeepFace (Parkhiand, Vedaldi, and Zisserman 2015) and Deep Second-order Pooling (DeepO2P) (Ionescu, Vantzos, and Sminchisescu 2015); **4).** *Manifold network*: Network on SPD manifolds (SPDNet) (Huang and Van Gool 2017) and DeepO2P that trains standard ConvNets with a manifold layer end-to-end

We use source codes of all the comparing methods from authors with tuning their parameters as in the original papers. For our GrNet, we build its architecture with using i Projection-Pooling block(s) (named as GrNet-iBlock) and one Output block, all of which are illustrated in Fig.1. The learning rate λ and the batch size are set to 0.01 and 30 respectively. The FRMap matrices are all initialized as random full rank matrices, and the number of them per layer is set to 16. For all the ProjPooling layers, the number n of the instances for pooling are fixed as 4. For training the GrNet, we just use an i7-2600K (3.40GHz) PC without any GPUs[5]. Note that, the readers can follow the DeepO2P method to implement an end-to-end learning of ConvNet+GrNet. Since the focus of this paper is on deep learning for Grassmannian inputs, we leave the study in future work.

[5]As the matrix factorizations are implemented well in CUDA, we will achieve the GPU version of our GrNet for speedups.

Emotion Recognition: We utilize the popular Acted Facial Expression in Wild (AFEW) (Dhall et al. 2014) dataset. The dataset contains 1,345 sequences of facial expressions acted by 330 actors in close to real world setting. The standard protocol designed by (Dhall et al. 2014) splits the dataset into three data sets, i.e., training, validation and test data sets. In the training and validation data sets, each video is classified into one of seven expressions, while the ground truth of the test set has not been released. As a result, we follow (Liu et al. 2014a; Huang and Van Gool 2017) to present the results on the validation set. As done in many works such as (Huang and Van Gool 2017) for augmenting the training data, we split the training videos to 1,747 small subvideos. For the evaluation, each facial frame is normalized to an image of size 20×20. Then, following (Liu et al. 2013; 2014b), we model sequences of facial expression with a set of linear subspaces of order 10, which span a Grassmann manifold $Gr(10, 400)$. In the task, the sizes of the GrNet-1Block weights are set to 400×100, while those of the GrNet-2Blocks are set to 400×300 and 150×100.

Action Recognition: We use the HDM05 database (Müller et al. 2007) that is one of the largest-scale skeleton-based human action datasets. The dataset consists of 2,337 sequences of 130 action classes, and provides 3D locations of 31 joints of the subjects. Following the protocol designed in (Huang and Van Gool 2017), we conduct 10 random evaluations, each of which randomly selected half of sequences for training and the rest for testing. For data augmentation, the training sequences are divided into around 18,000 small subsequences in each random evaluation. As done in (Harandi, Salzmann, and Hartley 2014; Huang and Van Gool 2017), we represent each sequence by a 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. Then, we apply SVD on the covariance descriptors to get linear subspaces of order 10, which form the data on a Grassmannian $Gr(10, 93)$. For our GrNet-1Block, the sizes of the connection weights are set to 93×60, while those of GrNet-2Blocks are fixed as $93 \times 80, 40 \times 30$ respectively.

Face Verification: We employ one standard dataset named Point-and-Shoot Challenge (PaSC) (Beveridge et al. 2013). For 256 subjects, it owns 1,401 videos taken by control cameras, and 1,401 videos from handheld cameras. For the dataset, (Beveridge et al. 2013) designs control and handheld face verification experiments. As done in (Beveridge, Zhang, and others 2015; Huang and Van Gool 2017), we use its 280 training videos and the COX data (Huang et al. 2015a) with 900 videos for training. Similarly, the training data are split to 12,529 small clips. To extract the state-of-the-art deep features, we perform the approach of (Parkhiand, Vedaldi, and Zisserman 2015) on the normalized face images of size 224×224. To speed up the training, we employ PCA to reduce the deep features to 400-dimensional ones. Following (Huang and Van Gool 2017), a SPD matrix of size 401×401 is computed by fusing covariance matrix and mean for each video. As done in (Huang et al. 2015b) on each video, we finally compute a linear subspace of order 10, which lies on $Gr(10, 401)$. We set the sizes of GrNet-1Block weights to 401×100, while setting those to 401×300 and 150×100 for GrNet-2Blocks.

Experimental Analysis

Table.1 presents the performances of the comparing methods for the three used datasets. The results show our GrNet with 2 blocks can outperform the existing general manifold learning, Grassmann learning methods and standard ConvNets (i.e., VGGDeepFace and DeepO2P). Particularly, on HDM05, we can observe that our GrNet outperforms the state-of-the-art Grassmann learning methods by a large margin (more than 11%). This verifies that the proposed GrNet yields great improvements when the training data is large enough. For PaSC, although the used softmax output layer in the GrNet does not suit the verification task well, we find that it still reaches the highest performances in the case of 2 blocks, which learns more favorable Grassmannian representation. As studied in existing state-of-the-art Grassmann learning methods, the GrNet without Riemannian computing (i.e., ProjMap) and without geometry-aware learning (i.e., ReOrth) perform very badly (17.62% and 26.15% respectively for AFEW). Besides, the consistent improvement of stacking more GrNet blocks verifies it can learn more discriminative Grassmannian representations and finally improve the classification performances.

By comparing one of the manifold networks DeepO2P that uses an end-to-end training of ConvNet with a single SPD manifold layer, our GrNets achieve better performances. In contrast, our GrNets fail to surpass the other manifold network SPDNets that use multiple SPD manifold layers. Nevertheless, our GrNets perform deep learning on a different type of manifolds with owning many considerable differences in both terms of application range and intrinsic properties, some of which are enumerated below.

1). The proposed FPMap layers learn full rank projections while the BiMap layers in SPDNets pursue bi-linear orthogonal projections. Besides, the GrNets exploit both single and multiple projections in each FRMap layer (i.e., S-FRMap, M-FRMap). The results in Fig.2 (a) show the benefit of using M-FRMap. Furthermore, it alternatively studies a new connection weight update rule on PSD manifolds rather than the one on Stiefel manifolds, whose performances (i.e., 32.13%, 57.25%, 80.15%, 71.28%) on the datasets are often worse.

2). The GrNets design brand new ReOrth, ProjMap and ProjPooling layers. Particularly, for the ReOrth layers, the exploration of the QR decomposition in backpropagation is a very important theoretical contribution. Besides, it devises pooling layer across and within projection matrices (A-ProjPooling, W-ProjPooling), both of which work on the top of the M-FRMap layers. As studied in Fig.2 (a), W-ProjPooling typically outperforms A-ProjPooling. The unfavorable performance of A-ProjPoolings would be caused by the extrinsic mean calulation on the Grassmannian and the weak relationship across multiple projection matrices.

Method	AFEW	HDM05	PaSC1	PaSC2
STM-ExpLet	31.73%	–	–	–
RSR-SPDML	30.12%	48.01%±3.38	–	–
DCC	25.78%	41.74%±1.92	75.83%	67.04%
GDA	29.11%	46.25%±2.71	71.38%	67.49%
GGDA	29.45%	46.87%±2.19	66.71%	68.41%
PML	28.98%	47.25%±2.78	73.45%	68.32%
VGGDeepFace	–	–	78.82%	68.24%
DeepO2P	28.54%	–	68.76%	60.14%
SPDNet	34.23%	61.45%±1.12	80.12%	72.83%
GrNet-0Block	25.34%	51.12%±3.55	68.52%	63.92%
GrNet-1Block	32.08%	57.73%±2.24	80.15%	72.51%
GrNet-2Blocks	34.23%	59.23%±1.78	80.52%	72.76%

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

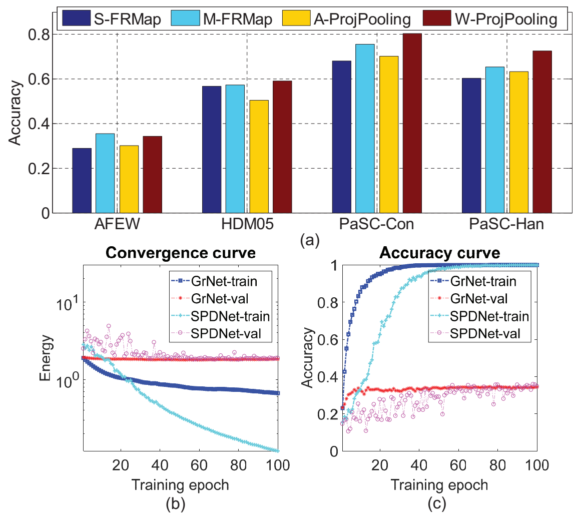

Figure 2: (a) Results of using single and multiple FRMap (S-FRMap, M-FRMap), ProjPoolings across or within projections (A-ProjPooling, W-ProjPooling) for the three used databases. (b) (c) Convergence and accuracy curves of SPDNet and the proposed GrNet for the AFEW.

3). Training GrNet-1Block per epoch costs about 10, 9 and 13 minutes respectively on the three datasets, while training SPDNet (w/o complex pooling) takes 2, 4 and 15 minutes. However, in theory, our GrNet actually runs much faster than the existing SPDNet when using the same setting. This is because the GrNet handles much lower-dimensional orthonormal matrices of size $d \times q$ (the order q is often set to 10), while the SPDNet treats SPD matrices of size $d \times d$.

4). Fig.2 (b)(c) show the GrNet can use much less epochs (than the SPDNet) to converge on AFEW, and the validation gets near 12% improvement after training. For larger datasets like HDM05, the improvement is even up to 40%.

Conclusion

This paper introduced the first network architecture to perform deep learning over Grassmann manifolds. Essentially, it is a natural exploration of convolutional networks to perform fully connected convolution, normalization, pooling and Remannian computing on Grassmannian data. In three typical visual classification evaluations, our Grassmann networks significantly outperformed existing Grassmann learning methods, and performed comparably with state-of-the-art methods. Directions for future work include extending the current GrNet to an end-to-end ConvNet+GrNet training system and applying it to other computer vision problems.
Acknowledgement: This work is supported by EU Framework Seven project ReMeDi (grant 610902).

Appendix

I. Regarding the gradient computation of QR decomposition, we first differentiate its implicit system

$$\boldsymbol{X}_{k-1} = \boldsymbol{QR}, \quad \boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}, \quad 0 = \boldsymbol{R}_{tril}, \tag{17}$$

where $\boldsymbol{R}_{tril}$ returns the elements below the main diagonal of $\boldsymbol{R}$, and we obtain

$$d\boldsymbol{X}_{k-1} = d\boldsymbol{Q}\boldsymbol{R} + \boldsymbol{Q}d\boldsymbol{R}, \quad d\boldsymbol{Q}^T\boldsymbol{Q} = -\boldsymbol{Q}^T d\boldsymbol{Q}. \tag{18}$$

Multiplying the first equation of Eqn.18 from the left with $\boldsymbol{Q}^T$ and the right with $\boldsymbol{R}^{-1}$ derives

$$d\boldsymbol{R} = \boldsymbol{Q}^T d\boldsymbol{X}_{k-1} - \boldsymbol{Q}^T d\boldsymbol{Q}\boldsymbol{R}, \tag{19}$$

$$d\boldsymbol{Q} = d\boldsymbol{X}\boldsymbol{R}^{-1} - \boldsymbol{Q}d\boldsymbol{R}\boldsymbol{R}^{-1}. \tag{20}$$

The multiplication of Eqn.19 from the right with the inverse of $\boldsymbol{R}$ yields the equation

$$0 = \boldsymbol{Q}^T d\boldsymbol{X}_{k-1}\boldsymbol{R}^{-1} - \boldsymbol{Q}^T d\boldsymbol{Q}\boldsymbol{R}\boldsymbol{R}^{-1} - d\boldsymbol{R}\boldsymbol{R}^{-1}. \tag{21}$$

As $(d\boldsymbol{R}\boldsymbol{R}^{-1})_{tril} = 0$, we further derive $(\boldsymbol{Q}^T d\boldsymbol{Q})_{tril} = (\boldsymbol{Q}^T d\boldsymbol{X}_{k-1}\boldsymbol{R}^{-1})_{tril}$. Since $\boldsymbol{Q}^T d\boldsymbol{Q} = (\boldsymbol{Q}^T d\boldsymbol{Q})_{tril} - ((\boldsymbol{Q}^T d\boldsymbol{Q})_{tril})^T$ is antisymmetric (see Eq.18) we have

$$\begin{aligned}\boldsymbol{Q}^T d\boldsymbol{Q} &= (\boldsymbol{Q}^T d\boldsymbol{X}_{k-1}\boldsymbol{R}^{-1})_{tril} - ((\boldsymbol{Q}^T d\boldsymbol{X}_{k-1}\boldsymbol{R}^{-1})_{tril})^T \\ &= (\boldsymbol{Q}^T d\boldsymbol{X}_{k-1}\boldsymbol{R}^{-1})_{asym}.\end{aligned} \tag{22}$$

Substituting Eqn.22 into Eqn.19 derives the gradient of QR decomposition w.r.t $\boldsymbol{R}$ as Eqn.14. By plugging Eqn.14 into Eqn.20, we can derive the gradient of the QR decomposition w.r.t $\boldsymbol{Q}$ as Eqn.13.

II. When plugging Eqn.13 and Eqn.14 into Eqn.12 to achieve Eqn.15, we employ the properties of matrix inner product $\boldsymbol{A} : \boldsymbol{B} = Tr(\boldsymbol{A}^T\boldsymbol{B})$, which were studied in (Ionescu, Vantzos, and Sminchisescu 2015), to derive the following equivalent equation

$$\boldsymbol{A} : \boldsymbol{B}_{asym} = \boldsymbol{A}_{bsym} : \boldsymbol{B}, \tag{23}$$

where $\boldsymbol{B}_{asym} = \boldsymbol{B}_{tril} - (\boldsymbol{B}_{tril})^T$, $\boldsymbol{A}_{bsym} = \boldsymbol{A}_{tril} - (\boldsymbol{A}^T)_{tril}$, $\boldsymbol{A}_{tril}$ extracts the elements below the main diagonal of $\boldsymbol{A}$.

References

Absil, P.-A.; Mahony, R.; and Sepulchre, R. 2004. Riemannian geometry of Grassmann manifolds with a view on algorithmic computation. *Acta Applicandae Mathematica* 80(2):199–220.

Absil, P.-A.; Mahony, R.; and Sepulchre, R. 2009. *Optimization algorithms on matrix manifolds*. Princeton University Press.

Anirudh, R.; Turaga, P.; Su, J.; and Srivastava, A. 2017. Elastic functional coding of Riemannian trajectories. *IEEE-TPAMI* 39(5):922–936.

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

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

Bonnabel, S., and Sepulchre, R. 2009. Riemannian metric and geometric mean for positive semidefinite matrices of fixed rank. *SIAM Journal on Matrix Analysis and Applications* 31(3):1055–1070.

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

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

Cetingul, H., and Vidal, R. 2009. Intrinsic mean shift for clustering on Stiefel and Grassmann manifolds. In *CVPR*.

Cherian, A.; Fernando, B.; Harandi, M.; and Gould, S. 2017. Generalized rank pooling for activity recognition. *CVPR*.

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*.

Dodge, Y., and Rousson, V. 1999. Multivariate L1 mean. *Metrika* 49(2):127–134.

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

Goodfellow, I.; Warde-Farley, D.; Mirza, M.; Courville, A.; and Bengio, Y. 2013. Maxout networks. In *ICML*.

Hamm, J., and Lee., D. D. 2008. Grassmann discriminant analysis: a unifying view on subspace-based learning. In *ICML*.

Hamm, J., and Lee, D. D. 2009. Extended Grassmann kernels for subspace-based learning. In *NIPS*.

Harandi, M.; Sanderson, C.; Shirazi, S.; and Lovell, B. C. 2011. Graph embedding discriminant analysis on Grassmannian manifolds for improved image set matching. In *CVPR*.

Harandi, M.; Sanderson, C.; Shen, C.; and Lovell, B. 2013. Dictionary learning and sparse coding on Grassmann manifolds: An extrinsic solution. In *ICCV*.

Harandi, M. T.; Salzmann, M.; Jayasumana, S.; Hartley, R.; and Li, H. 2014. Expanding the family of Grassmannian kernels: An embedding perspective. In *ECCV*.

Harandi, M. T.; Salzmann, M.; and Hartley, R. 2014. From manifold to manifold: Geometry-aware dimensionality reduction for SPD matrices. In *ECCV*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*.

Helmke, U.; Hüper, K.; and Trumpf, J. 2007. Newton's method on Grassmann manifolds. *arXiv preprint arXiv:0709.2205*.

Huang, Z., and Van Gool, L. 2017. A Riemannian network for SPD matrix learning. In *AAAI*.

Huang, Z.; Shan, S.; Wang, R.; Zhang, H.; Lao, S.; Kuerban, A.; and Chen, X. 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.; and Chen, X. 2015b. Projection metric learning on Grassmann manifold with application to video based face recognition. In *CVPR*.

Ionescu, C.; Vantzos, O.; and Sminchisescu, C. 2015. Training deep networks with structured layers by matrix backpropagation. *arXiv:1509.07838*.

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

Journee, M.; Bach, F.; Absil, P.-A.; and Sepulchre, R. 2010. Low-rank optimization on the cone of positive semidefinite matrices. *SIOPT*.

Kim, T.-K.; Kittler, J.; and Cipolla, R. 2007. Discriminative learning and recognition of image set classes using canonical correlations. *IEEE T-PAMI* 29(6):1005–1018.

Le, H. 1991. On geodesics in Euclidean shape spaces. *J. Lond. Math. Soc.* 360–372.

Liu, M.; Wang, R.; Huang, Z.; Shan, S.; and Chen, X. 2013. Partial least squares regression on Grassmannian manifold for emotion recognition. In *ICMI*.

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

Liu, M.; Wang, R.; Li, S.; Shan, S.; Huang, Z.; and Chen, X. 2014b. Combining multiple kernel methods on Riemannian manifold for emotion recognition in the wild. In *ICMI*.

Marrinan, T.; B.J., R.; Draper, B.; Kirby, M.; and Peterson, C. 2014. Finding the subspace mean or median to fit your need. In *CVPR*.

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

Meyer, G.; Bonnabel, S.; and Sepulchre, R. 2011. Regression on fixed-rank positive semidefinite matrices: a Riemannian approach. *JMLR* 12:593–625.

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*.

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

Srivastava, A., and Klassen, E. 2002. Monte carlo extrinsic estimators of manifold-valued parameters. *IEEE Transactions on Signal Processing* 50(2):299–308.

Srivastava, A., and Klassen, E. 2004. Bayesian and geometric subspace tracking. *Advances in Applied Probability* 43–56.

Turaga, P.; Veeraraghavan, A.; Srivastava, A.; and Chellappa, R. 2011. Statistical computations on Grassmann and Stiefel manifolds for image and video-based recognition. *IEEE-TPAMI* 33(11):2273–2286.

Vemulapalli, R.; Pillai, J.; and Chellappa, R. 2013. Kernel learning for extrinsic classification of manifold features. In *CVPR*.