Learning Invariant Riemannian Geometric Representations Using Deep Nets

Suhas Lohit
Arizona State University
slohit@asu.edu

Pavan Turaga
Arizona State University
pturaga@asu.edu

Abstract

Non-Euclidean constraints are inherent in many kinds of data in computer vision and machine learning, typically as a result of specific invariance requirements that need to be respected during high-level inference. Often, these geometric constraints can be expressed in the language of Riemannian geometry, where conventional vector space machine learning does not apply directly. The central question this paper deals with is: How does one train deep neural nets whose final outputs are elements on a Riemannian manifold? *To answer this, we propose a general framework for manifold-aware training of deep neural networks – we utilize tangent spaces and exponential maps in order to convert the proposed problem into a form that allows us to bring current advances in deep learning to bear upon this problem. We describe two specific applications to demonstrate this approach: prediction of probability distributions for multi-class image classification, and prediction of illumination-invariant subspaces from a single face-image via regression on the Grassmannian. These applications show the generality of the proposed framework, and result in improved performance over baselines that ignore the geometry of the output space. In addition to solving this specific problem, we believe this paper opens new lines of enquiry centered on the implications of Riemannian geometry on deep architectures.*

1. Introduction

Many applications in computer vision employ data that are naturally represented on manifolds [32]. Shapes that are invariant to affine transforms [5] and linear dynamical systems [42] can be represented as points on the Grassmannian. In diffusion tensor imaging, each ”pixel” of the ”image” is a symmetric positive definite (SPD) matrix and the space of SPD matrices forms a manifold [36]. Lie groups like $SO(3)$ and $SE(3)$ are used to represent human skeletons [43, 44]. Predicting probability density functions is another area of interest, applicable to multi-class classification and bag-of-words models [30], and saliency prediction [25].

Several years of research has presented us with various tools for statistics, and thereby machine learning approaches to be deployed when the objects of interest have manifold-valued domains (c.f. [40]). In deep learning, it is usually the case that data samples are viewed as elements of vector spaces. Any additional structure that the data may possess is left to be learned through the training examples. However, recently, there has been interest in employing deep learning techniques for non-Euclidean inputs as well: [7] including graph-structured data [8, 20, 34] and 3D shapes viewed as Riemannian manifolds [33]. Also, deep networks that preserve the input geometry at each layer have been studied for inference problems, e.g., for symmetric positive definite matrices [22], Lie groups [23] and points on the Stiefel manifold [24]. Another recent work considers weight matrices which are constrained to be orthogonal, i.e., points of the Stiefel manifold, and propose a generalized version of backpropagation [19]. These works do not consider output variables with geometric constraints.

In contrast to the above, instead of enforcing geometry at the inputs, our goal is to design a general framework to extend neural network architectures where output variables (or deeper feature maps) lie on manifolds of known geometry, typically due to certain invariance requirements. We do not assume the inputs themselves have known geometric structure and employ standard back-propagation for training. Equivalently, one may consider this approach as trying to estimate a mapping from an input $\mathbf{x} \in \mathbb{R}^N$ to a manifold-valued point $\mathbf{m} \in \mathcal{M}$ i.e., $f : \mathbb{R}^N \rightarrow \mathcal{M}$, using a neural network, where $\mathbf{m}$ is the desired output.

That is, this paper provides a framework for regression that is applicable to predicting manifold-valued data and at the same time is able to leverage the power of neural nets for feature learning, using standard backpropagation for unconstrained optimization. In this paper, we focus on two manifolds that are of wide interest in computer vision – the hypersphere and the Grassmannian. We describe the applications next.

Face $\rightarrow$ Illumination Subspace as regression on the Grassmannian: The *illumination subspace* of a human face is a popular example from computer vision where for a particular subject, the set of all face images of that subject under all illumination conditions can be shown to lie close to a low dimensional subspace [17]. These illumination subspaces are represented as points on the Grassmannian (or Stiefel, depending on application) manifold. Several applications such as robust face recognition have been proposed using this approach. In this work, in order to demonstrate how deep networks can be employed to map to Grassmannian-valued data, we consider the problem of estimating the illumination subspace from a single input image of a subject under unknown illumination. We refer to this application as Face$\rightarrow$Illumination Subspace (F2IS).

Multi-class classification as regression on the unit hypersphere: Classification problems in deep learning use the softmax layer to map arbitrary vectors to the space of probability distributions. However, more formally, probability distributions can be easily mapped to the unit hypersphere, under a square-root parametrization [38] inspired by the Fisher-Rao metric used in information geometry. Thus, multi-class classfication can be posed as regression to a hypersphere. Indeed, there has been work recently that consider *spherical-loss functions* which use the Euclidean loss on unit-norm output vectors of a network [45, 11]. In this work, we propose loss-functions for the classification problem based on the geometry of the sphere.

Main contributions: In this paper, we address the training of neural networks using standard backpropagation to output elements that lie on Riemannian manifolds. To this end, we propose two frameworks in this paper:

(1) We discuss how to map to simpler manifolds like the hypersphere directly using a combination of geodesic loss functions as well as differentiable constraint satisfaction layers such as the normalization layer in the case of the hypersphere.

(2) We also propose a more general framework that is applicable to Riemannian manifolds that may not have closed-form expressions for the geodesic distance or when the constraints are hard to encode as a layer in the neural network. In this framework, the network maps to the tangent space of the manifold and then the exponential map is employed to find the desired point on the manifold.

We carry out experiments for the applications described above in order to evaluate the proposed frameworks and show that geometry-aware frameworks result in improved performance compared to baselines that do not take output geometry into account.

2. Related work

We will now point to some related work that also examine the problem of predicting outputs with geometric structure using neural networks. Byravan and Fox [9] and Clark et al. [10] design deep networks to output $SE(3)$ transformations. The set of transformations $SE(3)$ is a group which also possesses manifold structure, i.e., a Lie group. It is not straightforward to predict elements on $SE(3)$ since it involves predicting a matrix constrained to be orthogonal. Instead, the authors map to the Lie algebra $se(3)$ which is a linear space. We note that the Lie algebra is nothing but the tangent space of $SE(3)$ at the identity transformation and can be considered a particular case of the general formulation presented in this paper. Huang et al. [23] use the logarithm map to map *feature maps* on $SE(3)$ to $se(3)$ before using regular layers for action recognition. However, the logarithm map is implemented within the network, since for $SE(3)$, this function is simple and differentiable. In contrast, in this work, we require the network *output* to be manifold-valued and thus do not impose any geometry requirements at the input or for the feature maps. This also means that a suitable loss function needs to be defined, taking into account, the structure of the manifold of interest.

In a more traditional learning setting, there has been work using *geodesic regression*, a generalization of linear regression, on Riemannian manifolds [15, 14, 37, 21, 26], where a geodesic curve is computed such that the average distance (on the manifold) from the data points to the curve is minimized. This involves computing gradients on the manifold. Recent work has also included non-linear regression on Riemannian manifolds [4, 3]. Here, the non-linearity is provided by a pre-defined kernel function and the mapping algorithm solves an optimization problem iteratively. Our work is a non-iterative deep-learning based approach to the problem described in Banerjee et al. [4] as regression with the independent variable in $\mathbb{R}^N$ and the dependent variable lying on a manifold $\mathcal{M}$. That is, unlike these works, the mapping $f : \mathbb{R}^N \rightarrow \mathcal{M}$ in our case is a hierarchical non-linear function, learned directly from data without any hand-crafted feature extraction, and the required mapping is achieved by a simple feed forward pass through the trained network.

All neural nets in the paper are trained and tested using Tensorflow [1], making use of its automatic differentiation capability. Before we discuss the contributions of this work, if required, please refer supplementary material for definitions of some important terms from differential geometry.

3. Two approaches for deep manifold-aware prediction

We propose two ways of predicting manifold-valued data using neural networks with standard backpropagation. See

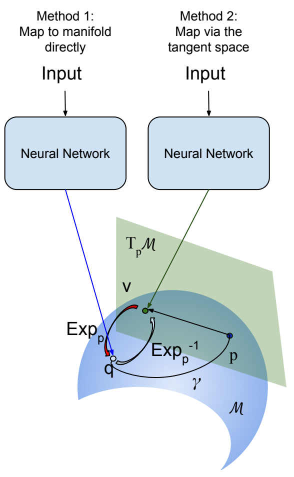


Figure 1. This figure illustrates the two approaches presented in this paper for training neural networks to predict manifold-valued data. It also explains some basic concepts from differential geometry visually. $\mathcal{M}$ is a manifold, $T_P\mathcal{M}$ is the tangent space at $p \in \mathcal{M}$. p is called the pole of the tangent space. The curve connected p and $q \in \mathcal{M}$ is the geodesic curve γ. v is a point on $T_p\mathcal{M}$ such that the exponential map $\exp_p(v) = \gamma(1)$ and the logarithm map $\exp_p^{-1}(q) = v$.

Figure 1 for visual illustration and important notation.

Mapping to the manifold via geodesic-loss functions: In this case, the network directly maps input vectors to elements on the manifold $\mathcal{M}$ and is required to learn the manifold constraints from the data. If we represent the neural network as a mapping $\texttt{NN}$, we have $\texttt{NN}$: $\mathbb{R}^N \rightarrow \mathcal{M}$. Firstly, unlike simple manifolds like the sphere, manifolds in general do not have a differentiable closed-form expression, that are also efficiently computable, for the geodesic distance function that can be used as a loss function for the neural network. Although one can still resort to using a differentiable loss function such as the Euclidean distance, this approach is not mathematically correct and does not yield the right estimate for distance on the manifold. Secondly, the network output has to satisfy the manifold constraints. In the case of the sphere, it is simple to enforce the unit-norm constraint at the output layer of the neural network using a differentiable normalization layer. It is however less clear how to map to more complicated manifolds such as the Stiefel and Grassmann manifolds where the points are usually represented by tall-thin orthonormal matrices. That is, in addition to the unit-norm constraints, orthogonality constraints between all pairs of columns in the matrix need to be enforced. The Grassmann manifold, presents a more difficult challenge, since each point in this space is an equivalence class of points on the Stiefel manifold that are orthogonal transforms of each other. As we will see later, the data representation that respects this equivalence (projection matrix) does not admit a feasible way for a neural network to map to this manifold directly.

Mapping via the tangent space – toward a general framework: This is a more general formulation that is applicable to all the manifolds of interest. Here, the network first maps to a vector on the tangent space constructed at a suitable pole $p \in \mathcal{M}$, which forms the intermediate output. Once the network outputs the required tangent, the exponential map ($\exp_p$) is employed to find the corresponding point on the manifold. Mathematically, we decompose the desired function $f : \mathbb{R}^N \rightarrow \mathcal{M}$ as $f = \exp_p \circ \texttt{ NN}$ and $\texttt{NN}$: $\mathbb{R}^N \rightarrow T_p\mathcal{M}$. Intuitively, since the tangent space is a vector space that encodes geometric constraints implicitly, it is attractive here, as neural networks have been shown to be effective for estimating vector-valued data. We note that an assumption is implicit in this framework: all the data points of interest on the manifold are much closer to p than the cut-locus of the manifold and in this case, the distance on the tangent space serves as a good approximation to the geodesic distance. This is the same assumption that goes into currently successful approaches for statistical computing methods on manifolds [36]. In practice, we find this assumption is respected in our applications as well.

4. Deep regression on the Grassmannian for F2IS

Face $\rightarrow$ Illumination Subspace: We will now describe an ill-posed inverse problem from computer vision that serves as our canonical application to illustrate prediction on the Grassmann manifold using a neural network. It is well known that the set of images of a human face in frontal pose under all illuminations lies close to a low-dimensional subspace, known as the illumination subspace [17, 13]. If we compute the eigenvectors of this set of images for different subjects using PCA, we observe that the top 5 principal components (PCs) capture nearly 90% of the variance. More importantly, for this paper, an obvious pattern can be observed between the subject under consideration and the PCs of the illumination subspace. Firstly, the identity of the subject can be easily determined from the PCs. Secondly, as noted by Hallinan [17], we observe that the illumination patterns of top 5 principal components are the same across subjects only up to certain permutations and sign flips. [1] Using the terminology in [17], we can interpret the visualizations of the top 5 PC's as a face under the following respective illuminations: *frontal lighting, side lighting, lighting from above/below, extreme side lighting and lighting from a corner*. The 1^{st} and 2^{nd} PC's have eigenvalues in a similar range and sometimes exchange places depending on the subject. The 3^{rd} PC, corresponding to eigenvalue is

[1]It is clear that for an eigenvector $\mathbf{e}$, $-\mathbf{e}$ is also an eigenvector.

always at the same place. The 4th and 5th PCs have eigenvalues in a similar range and can interchange places for a few subjects.

The illumination subspace refers to the linear *span* of these eigenvectors, and is a point on the Grassmannian. **When we represent the subspace by its projection matrix representation, the representation becomes invariant to both sign flips and permutations** (in fact, invariant to the full set of right orthogonal transforms).

In this paper, as an example of predicting points on Grassmann manifold, we define the following ill-posed inverse problem: given a human face in frontal pose under an unknown illumination, output the corresponding illumination subspace. We will refer to this problem as the "Face $\rightarrow$ Illumination Subspace" problem or F2IS. In our experiments, we consider the illumination subspace to be of dimension $d = 3, 4$,or 5.

Geometry of the Grassmannian: The Grassmann manifold, denoted by $\mathcal{G}_{n,d}$, is a matrix manifold and is the set of d-dimensional subspaces in $\mathbb{R}^n$. To represent a point on $\mathcal{G}_{n,d}$, we can use an orthonormal matrix, $\mathbf{U} \in \mathbb{R}^{n\times d}$ ($\mathbf{U}^T\mathbf{U} = \mathbf{I}_n$), to represent the equivalence class of points in $\mathbb{R}^{n\times d}$, such that, two points are equivalent if their columns span the same d-dimensional subspace. That is, $\mathcal{G}_{n,d} = \{[\mathbf{U}]\}$, where $[\mathbf{U}] = \{\mathbf{UQ} | \mathbf{U}^T\mathbf{U} = \mathbf{I}, \mathbf{Q} \text{ is orthogonal}\}$. In order to uniquely represent the equivalence class $[\mathbf{U}] \in \mathcal{G}_{n,d}$, we use its projection matrix representation $\mathbf{P} = \mathbf{UU}^T \in \mathbb{R}^{n\times n}$, where $\mathbf{U}$ is some point in the equivalence class. $\mathbf{UU}^T$ contains $\frac{n(n+1)}{2}$ unique entries as it is a symmetric matrix. Clearly, for any other point in the same equivalence class $\mathbf{UQ}$, its projection matrix representation is $(\mathbf{UQ})(\mathbf{UQ})^T = \mathbf{UU}^T$, as required. **Thus, the space of all rank d projection matrices of size $n \times n$, $\mathcal{P}_n$ is diffeomorphic to $\mathcal{G}_{n,d}$.** The identity element of $\mathcal{P}_n$ is given by $\mathbf{I}_{\mathcal{P}_n} = diag(\mathbf{I}_d, \mathbf{0}_{n-d})$, where $\mathbf{0}_{n-d}$ is the matrix of zeros of size $(n-d) \times (n-d)$. In order to find the exponential and logarithm maps for $\mathcal{G}_{n,d}$, we will view $\mathcal{G}_{n,d}$ as a quotient space of the orthogonal group, $\mathcal{G}_{n,d} = \mathcal{O}_n/(\mathcal{O}_{n-d} \times \mathcal{O}_d)$. The Riemannian metric in this case is the standard inner product [12] and thus, the distance function induced on the tangent space is the Euclidean distance function. Using this formulation, given any point $\mathbf{P} = \mathbf{UU}^T \in \mathcal{P}_n$, a geodesic of $\mathcal{P}_n$ at $\mathbf{I}_{\mathcal{P}_n}$ passing through $\mathbf{P}$ at $t = 0$, is a particular geodesic $\alpha(t)$ of $\mathcal{O}(n)$ completely specified by a skew-symmetric $\mathbf{X} \in \mathbb{R}^{n\times n}$: $\alpha(t) = \exp_m(t\mathbf{X}) I_P \exp_m(-t\mathbf{X})$, where $\exp_m(.)$ is the matrix exponential and $\mathbf{P} = \alpha(1)$, such that $\mathbf{X}$ belongs to the set M given by $M = \left\{ \begin{bmatrix} \mathbf{0}_d & \mathbf{A} \\ -\mathbf{A}^T & \mathbf{0}_{n-d} \end{bmatrix} \mid \mathbf{A} \in \mathbb{R}^{d\times(n-d)} \right\}$. $\mathbf{X}$ serves as the tangent vector to $\mathcal{G}_{n,d}$ at the identity and is completely determined by $\mathbf{A}$. The geodesic between two points $\mathbf{P}_1, \mathbf{P}_2 \in \mathcal{P}_n$, is computed by rotating $\mathbf{P}_1$ and $\mathbf{P}_2$ to $\mathbf{I}_{\mathcal{P}_n}$ and some $\mathbf{P} \in \mathcal{P}_n$ respectively. The exponential map, takes as inputs, the pole and the tangent vector and returns the subspace $span(\mathbf{U})$, represented by some point $\mathbf{UQ}$. Refer Srivastava and Klassen [39] and Taheri et al. [41] for algorithms to compute exponential and log maps for the Grassmannian.

Synthetic dataset for F2IS: We use the Basel Face Model dataset [35] in order to generate 250 random 3D models of human faces $\{S_i\}, i = 1 \dots 250$ (200 for training and 50 for testing chosen randomly).[2] We then generate a set of 64 faces for each subject where each face is obtained by varying the direction of the point source illumination for the frontal pose i.e., $S_i = \{\mathbf{F}_i^j\}, j = 1 \dots 64$. The directions of illumination are the same as the ones used in the Extended Yale Face Database B [16]. Each face image is converted to grayscale and resized to 28×28. Once we have the 250 sets of faces under the 64 illumination conditions, we calculate the illumination subspace for each subject as follows. For each subject, we first subtract the mean face image of that subject under all illumination conditions and then calculate the principal components (PCs). For a subject i and an illumination condition j, we will denote the input face image by $\mathbf{F}_i^j$ and the desired d top PCs by $\mathbf{E}_i^1, \mathbf{E}_i^2, \dots, \mathbf{E}_i^d$ (note that the PCs do not depend on the input illumination condition). It is clear that every $\mathbf{E}_i^k, k = 1, 2, \dots, d$ is of size 28×28 and $\langle \mathbf{E}_i^k, \mathbf{E}_i^l \rangle = 1$, if $k = l$ and 0 otherwise.

If we lexicographically order each $\mathbf{E}_i^k$ to form a vector $vec(\mathbf{E}_i^k)$ of size 784×1 and for each subject, arrange the $\mathbf{E}_i^k$'s to form a matrix $\mathbf{U}_i = [vec(\mathbf{E}_i^1)\, vec(\mathbf{E}_i^2) \dots vec(\mathbf{E}_i^d)]$, then the orthonormality constraint can be rewritten as $\mathbf{U}_i^T\mathbf{U}_i = \mathbf{I}_d$, where $\mathbf{I}_d$ is the identity matrix of size $d \times d$. As we argued earlier, due to the nature of the problem, $\mathbf{U}_i$ should be represented as a point on the Grassmann $\mathcal{G}_{784,d}$ using the projection matrix representation. With this notation, the desired mapping is $f : \mathbb{R}^{28\times 28} \rightarrow \mathcal{G}_{784,d}$ such that $f(\mathbf{F}_i^j) = \mathbf{U}_i\mathbf{Q} \in [\mathbf{U}_i]$, the required equivalence class or equivalently, $\mathbf{U}_i\mathbf{U}_i^T$.

For the inputs $\mathbf{F}_i^j$ during training and testing, we do not use all the illumination directions ($j's$). We only use illumination directions that light at least half of the face. This is because for extreme illumination directions, most of the image is unlit and does not contain information about the identity of the subject, which is an important factor for determining the output subspaces. We select the same 33 illumination directions for all subjects to form the inputs for the network. We randomly split the dataset into 200 subjects for training and 50 subjects for testing. Therefore there are $33 \times 200 = 6600$ and $33 \times 50 = 1650$ different input-output

[2]We use a synthetic dataset because we were unable to find any large publicly available real database that would enable training of neural networks without overfitting.

pairs for training and testing respectively. The 33 illumination directions used for creating inputs for both the training and test sets are given in the supplementary and are a subset of the illumination directions used in the Extended Yale Face Database B [16].

4.1. Proposed frameworks for solving F2IS

We propose two frameworks which employ networks with nearly the same architecture: The network consists of 3 `conv` layers and two `fc` layers. ReLU non-linearity is employed. Each `conv` layer produces 16 feature maps. All the filters in the `conv` layers are of size 11×11. The first `fc` layer produces a vector of size 512. Size of the second `fc` layer depends on the framework. Both networks are trained using the Adam optimizer [27] using a learning rate of 10^{-3} for 50000 iterations with a mini-batch size of 30. Euclidean loss between the desired output and ground truth is employed in both cases. We show that the choice of representation of the desired output is crucial in this application. We carry out three sets of experiments using subspace dimension $d = 3, 4$ and 5.

Baseline: The first framework is a baseline that attempts to directly map to the desired PCs represented as a matrix $\mathbf{U}_i$, given $\mathbf{F}_i^j$, i.e., $\mathtt{NN}(\mathbf{F}_i^j) = \mathbf{U}_i$. We use the Euclidean loss function between the ground-truth $\mathbf{U}_i$ and the network output $\hat{\mathbf{U}}_i$ for training: $L_b = ||\mathbf{U}_i - \hat{\mathbf{U}}_i||_F^2$ That is, instead of regressing to the desired subspace, the network attempts to map to its basis vectors (PCs). However, the mapping from $\mathbf{F}_i^j$ to $\mathbf{U}_i$ is consistent across subjects **only up to certain permutations and sign flips in the PCs**. Hence, without correcting these inconsistencies ad hoc, the problem is rendered too complicated, since during the training phase, the network receives conflicting ground-truth vectors depending on the subject. Thus, this framework performs poorly as expected. It is important to note that mapping to the correct representation $\mathbf{U}\mathbf{U}^T$ (which respects Grassmann geometry and is invariant to these inconsistencies) directly is not feasible because the size of $\mathbf{U}\mathbf{U}^T$ is too large ($\frac{784 \times 785}{2}$) and has rank constraints. This necessitates mapping via the tangent space, discussed next.

Mapping via the Grassmann tangent space – GrassmannNet-TS: The second framework represents the output subspaces as points on the Grassmann manifold and first maps to the Grassmann tangent space and then computes the required subspace using the Grassmann exponential map. This circumvents the problem of very large dimensionality encountered in the first approach since the tangent vector has a much smaller intrinsic dimensionality. This representation has a one-to-one mapping with the projection matrix representation and thus, is naturally invariant to the permutations of the PCs and all combinations of sign flips present in the data. And the mapping we intend to learn becomes feasible in a data-driven framework. Mathematically, $\mathtt{NN}$: $\mathbb{R}^{784 \times d} \to T_p\mathcal{G}_{784,d}$.

As shown in Section 4, a tangent at some pole p is given by the matrix $\mathbf{X}$, which in turn is completely specified by the matrix $\mathbf{A} \in \mathbb{R}^{(784-d)\times d}$, a much smaller matrix. Therefore, we design a network to map an input face $\mathbf{F}_i^j$ to the desired matrix $\mathbf{A_i}$. An training pair can be represented as $(\mathbf{F}_i^j, \mathbf{A}_i)$ and the let the output of the network be the vectorized version of a matrix $\hat{\mathbf{A}}_i \in \mathbb{R}^{(784-d)\times d}$. The $\mathbf{A}_i$'s are computed using the Grassmann logarithm map in [39]. We also note that $\mathbf{A}$ does not possess any additional structure to be enforced and thus a neural network can be easily trained to map to this space using just the Euclidean loss between $\mathbf{A}_i$ and the network output $\hat{\mathbf{A}}_i$: $L_G = ||\mathbf{A}_i - \hat{\mathbf{A}}_i||_F^2$.

Pole of tangent space: This is a design choice and we conduct experiments with two different poles: (1) We compute the the illumination subspace of the entire training set. We will denote these PCs by $\mathbf{E}_{Tr}^k, k = 1, 2, \ldots, d$ and the corresponding matrix representation by $\mathbf{U}_{Tr}^d$, which forms the pole of the Grassmann tangent space. (2) It is common practice to use the Fréchet (also known as geometric or Karcher) mean as the pole of the tangent space. We compute the Fréchet mean of the ground-truth subspaces of the training set using the iterative algorithm given by Turaga et al. [42]. We denote this pole as $\mathbf{U}_{Fr}^d$.

During the testing phase, using the output $\hat{\mathbf{A}}$ matrix, we employ the exponential map for a given pole to find the corresponding point on the Grassmann manifold. This framework of first mapping to the Grassmann tangent space using a network and then to the corresponding subspace using the Grassmann exponential map is referred to as **GrassmannNet-TS**. See supplementary material for details on visualizing the output Grassmann point.

4.2. Experimental results on F2IS

For the frameworks described in Sections 4.1, we describe the results on the test set of F2IS here. We compute the distance between predicted and ground-truth subspace as the measure to quantify the efficacy of the proposed frameworks. Various measures exist that quantify this notion based on principal angles between subspaces [18]. We use the Grassmann geodesic distance. For two subspaces represented by $\mathbf{U}_1$ and $\mathbf{U}_2 \in \mathcal{G}_{n,d}$, the geodesic distance is given by $D_G(\mathbf{U}_1, \mathbf{U}_2) = \left(\sum_{i=1}^d \theta_i^2\right)^{1/2}$, where θ_i's are the principal angles obtained by the SVD of $\mathbf{U}_1^T\mathbf{U}_2 = \mathbf{W}(\cos\mathbf{\Theta})\mathbf{V}^\mathbf{T}$, where $\cos\Theta = diag(\cos\theta_1, \ldots, \cos\theta_d)$. We use the implementation in [28] for computing the principal angles. We report the arithmetic mean of this distance measure for the entire test set. Note that the maximum value of $D_G(.)$ is $\frac{\pi\sqrt{d}}{2}$.

The results based on the mean subspace distance on the test set for the proposed frameworks for different values of

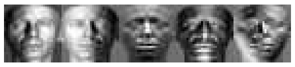
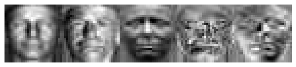
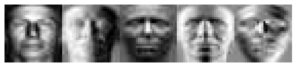
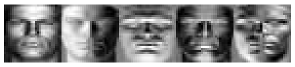
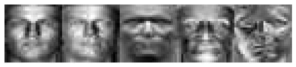
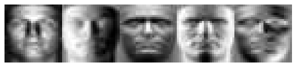
Input	Ground-truth PCs	Output of baseline n/w	Output of GrassmannNet-TS
			
		$D_G = 1.6694$	$D_G = 0.7006$
			
		$D_G = 1.2998$	$D_G = 0.7238$

Table 1. Test results for two input images using $d = 5$. We can clearly observe that the GrassmannNet-TS (with pole $\mathbf{U}^d_{Fr}$) framework performs much better than the baseline that attempts to regress directly to the PCs. The numbers below the output images indicate the subspace distance from the ground truth (lower the better). Note that the outputs need not be exactly the same as the groundtruth PCs since the quantity of interest is the subspace spanned by the groundtruth PCs. See Supplementary Material for more results.

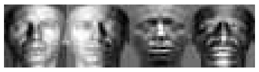
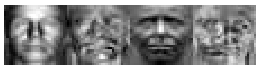
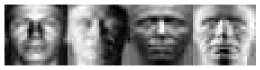
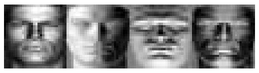
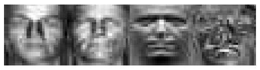
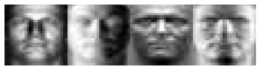
Input	Ground-truth PCs	Output of baseline n/w	Output of GrassmannNet-TS
			
		$D_G = 1.9400$	$D_G = 0.5489$
			
		$D_G = 1.2735$	$D_G = 0.6245$

Table 2. Test results for two input images using $d = 4$. As in the case of $d = 5$, GrassmannNet-TS (with pole $\mathbf{U}^d_{Fr}$) framework performs much better than the baseline that attempts to regress directly to the PCs. See Supplementary Material for more results.

Subspace Dim d	Baseline	GrassmannNet-TS	
		Pole = $\mathbf{U}^d_{Tr}$	Pole = $\mathbf{U}^d_{Fr}$
3	0.6613	0.3991	0.3953
4	1.0997	0.5489	0.5913
5	1.4558	0.8694	0.6174

Table 3. Mean geodesic distance between predictions and ground-truth on the test set using the proposed frameworks. GrassmannNet-TS expectedly provides excellent results compared to the baseline framework for all subspace dimensions. Note that the max $D_G(.)$ possible for $d = 3, 4$ and 5 are 2.72, 3.14 and 3.51 respectively

the subspace dimension d are presented in Table 3. The baseline, as expected, performs poorly. This is because during the training phase, the network received conflicting ground-truth information because of the permutation and sign flips inherently present in the data. On the other hand, GrassmannNet-TS yields excellent performance as it is invariant to these transformations by design. We reiterate that this is possible only in the case of regression directly on the Grassmann tangent space since mapping to the Grassmann manifold is infeasible because of the very large number of variables required to represent the projection matrix. The outputs of the baseline as well as GrassmannNet-TS using the Fréchet mean as the pole are also presented visually in Tables 1 and 2 for two test images for $d = 4, 5$ respectively, and show similar trends. The choice of the pole does not seem to affect the results significantly except in the case of $d = 5$ where the Fréchet mean performs better. Additional results are shown in the Supplementary Material.

5. Deep regression on the unit hypersphere for multi-class classification

Reformulating classification as mapping to the unit hypersphere: For multi-class classification problems, deep networks usually output a probability distribution, where one uses the mode of the distribution to predict the class label. What ensures that the output elements form a probability distribution is the ”softmax layer”. However, by using a square-root parametrization – replacing each element in the distribution by its square-root – we can map a probability distribution to a point on the non-negative orthant of a unit hypersphere $\mathcal{S}^C$, where C is now the number of classes. The square-root parameterization reduces the complicated Riemannian metric on the space of probability density functions, the Fisher-Rao metric, to the simpler Euclidean inner product on the tangent space of the unit hypersphere with closed form expressions for the geodesic distance, exponential and logarithm maps [38]. In this work, equipped with the knowledge of differential geometry of the sphere, we propose different loss functions for the tackling the classification problem. We consider two main variants – learning a network to map to the sphere directly or map to its tangent space. We note that the constraint for a point to be on a sphere or to be a probability distribution is simple and can be easily satisfied by using an appropriate normalization (dividing by its 2-norm or using softmax). However, mapping to the tangent space of the sphere provides a novel perspective to the same problem and is more general since as we showed earlier, it is necessary for the Grassmannian.

Consider a classification problem with C classes. For a given input vector $\mathbf{x}$, let the ground-truth probability distribution over the class labels be $\mathbf{c}_{pd}$. The corresponding point on $\mathcal{S}^C$ is given by $\mathbf{c}_S$, such that $\mathbf{c}_S(i) = \sqrt{\mathbf{c}_{pd}(i)}, i = 1 \ldots C$. The pole $\mathbf{u}_S$ for constructing the tangent space $T_{\mathbf{u}_S}\mathcal{S}^C$ is chosen to be the point on $\mathcal{S}^C$ corresponding to the uniform distribution $\mathbf{u}_{pd}, \mathbf{u}_{pd}(i) = \frac{1}{C}, i = 1 \ldots C$. Let ξ be the desired point on $T_{\mathbf{u}_S}\mathcal{S}^C$ for the input $\mathbf{x}$ and is given by output of the log map $\xi = \exp^{-1}_{\mathbf{u}_S}(\mathbf{c}_S)$. **Let the output of the last fully connected layer be denoted by $\hat{\mathbf{o}}$.**

Geometry of the unit hypersphere [2]: The the n-dimensional unit sphere denoted as $\mathcal{S}^n$ and is defined as $\mathcal{S}^n = \{(x_1, x_2, \ldots, x_{n+1}) \in \mathbb{R}^{n+1} | \sum_{i=1}^{n+1} x_i^2 = 1\}$. Given any two points $\mathbf{x}, \mathbf{y} \in \mathcal{S}^n$, the geodesic distance between $\mathbf{x}$ and $\mathbf{y}$ is calculated using $d(\mathbf{x}, \mathbf{y}) = \cos^{-1}\langle \mathbf{x}, \mathbf{y}\rangle$. For a given point $\mathbf{x} \in \mathcal{S}^n$, the tangent space of $\mathcal{S}^n$ at $\mathbf{x}$ is given by $T_{\mathbf{x}}\mathcal{S}^n = \{\xi \in \mathbb{R}^n | \mathbf{x}^T\xi = 0\}$. Since the Riemannian metric (the inner product on the tangent space) is the usual Euclidean inner product, the distance function on the tangent space induced by this inner product is the Euclidean distance. The exponential map $\exp : T_{\mathbf{x}}\mathcal{S}^n \to \mathcal{S}^n$ is computed using the following formula: $\exp_{\mathbf{x}} \xi = \cos(||\xi||)\mathbf{x} + \sin(||\xi||)\frac{\xi}{||\xi||}$, where $\xi \in T_{\mathbf{x}}\mathcal{S}^n$. For $\mathbf{x}, \mathbf{y} \in \mathcal{S}^n$, the inverse exponential map $\exp^{-1} : \mathcal{S}^n \to T_{\mathbf{x}}\mathcal{S}^n$ is given by $\exp^{-1}_{\mathbf{x}}(\mathbf{y}) = \frac{d(\mathbf{x},\mathbf{y})}{||P_{\mathbf{x}}(\mathbf{y}-\mathbf{x})||} P_{\mathbf{x}}(\mathbf{y} - \mathbf{x})$. $P_{\mathbf{x}}(\mathbf{v})$ is the projection of a vector $\mathbf{v} \in \mathbb{R}^n$ onto $T_{\mathbf{x}}\mathcal{S}^n$, given by $P_{\mathbf{x}}(\mathbf{v}) = (\mathbf{I}_n - \mathbf{x}\mathbf{x}^T)\mathbf{v}$, $\mathbf{I}_n$ is the $n \times n$ identity matrix.

5.1. Mapping to the hypersphere directly: SNet-M

In this case, the network directly outputs points on the sphere and a training pair is represented as $(\mathbf{x}, \mathbf{c}_S)$. We call this framework Snet-M. At test time, the network outputs a point on the sphere and the corresponding probability distribution is computed by squaring the elements of the output. We propose the following loss functions on the sphere. While training, the loss is averaged over the entire batch. In this case, we also employ a normalizing layer as the last layer of the network which guarantees that $\hat{\mathbf{o}}$ lies on $\mathcal{S}^C$.

(1) **Euclidean loss on $\mathcal{S}^C$** : This simply measures the Euclidean distance between two points on a sphere and does not take into account the non-linear nature of the manifold: $L_{S_{euc}} = ||\mathbf{c}_S - \frac{\hat{\mathbf{o}}}{||\hat{\mathbf{o}}||_2}||_2^2$.

(2) **Geodesic loss on $\mathcal{S}^C$** : The "true" distance between the two points on the sphere is given by $\theta = \cos^{-1}\langle \mathbf{c}_S, \frac{\hat{\mathbf{o}}}{||\hat{\mathbf{o}}||_2}\rangle$. Since minimizing this function directly leads to numerical difficulties, we instead minimize its surrogate, $L_{S_{geo}} = 1 - \cos\theta$.

5.2. Mapping to the hypersphere via its tangent space: SNet-TS

Here, given an input, the algorithm first produces an intermediate output on $T_{\mathbf{u_S}}\mathcal{S}^C$ and then the exponential map is used to compute the desired point on $\mathcal{S}^C$. The corresponding probability distribution is computed by simply squaring each element of the vector. We refer to this framework as SNet-TS. A training example, then, is of the form $(\mathbf{x}, \xi)$, where ξ is the desired tangent vector. We propose the following loss functions on $T_{\mathbf{u_S}}\mathcal{S}^C$.

(1) **Euclidean loss on** $T_{\mathbf{u_S}}\mathcal{S}^C$: Measures the Euclidean distance between two points on the tangent space of the sphere : $L_{T_{euc}} = ||\xi - \hat{\mathbf{o}}||_2^2$. The output, $\hat{\mathbf{o}}$, is however not guaranteed to lie on $T_{\mathbf{u_S}}\mathcal{S}^C$ since a point on $T_{\mathbf{u_S}}\mathcal{S}^C$ needs to satisfy the constraint $\mathbf{x}^T\xi = 0$. Therefore, we first project $\hat{\mathbf{o}}$ to the $T_{\mathbf{u_S}}S^C$ and then use the exponential map.

(2) **Euclidean + Orthogonal loss on** $T_{\mathbf{u}_S}\mathcal{S}^C$: In order to improve the "tangentness" of the output vector, we add the inner product loss that encourages the orthogonality of the output vector relative to the pole, which is the tangent space constraint: $L_{T_{orth}} = ||\xi - \hat{\mathbf{o}}||_2^2 + \lambda(\hat{\mathbf{o}}^T\mathbf{u}_S)^2$.

(3) **Projection loss on** $T_{\mathbf{u}_S}\mathcal{S}^C$: Since a closed form expression exists to project an arbitrary vector onto $T_{\mathbf{u}_S}\mathcal{S}^C$, we implement the projection layer as the last layer that guarantees that the output of the projection layer lies on $T_{\mathbf{u_S}}\mathcal{S}^C$. We compute the Euclidean loss between the projected vector and the desired tangent: $L_{T_{proj}} = ||\mathbf{c}_S - P_{\mathbf{u}_S}\hat{\mathbf{o}}||_2^2$.

5.3. Experiments with image classification

Image classification problem is a widely studied problem in computer vision and will serve as an example to demonstrate training a network to map to points on a unit hypersphere and its tangent space. We now describe the experiments conducted using MNIST and CIFAR-10 datasets. We train 6 networks with different loss functions. The first network is a baseline using the softmax layer to output a probability distribution directly and employs the well-known cross-entropy loss. The next two networks use the SNet-M framework and $L_{S_{euc}}$ and $L_{S_{geo}}$ as the loss functions. The desired outputs in this case lie on $\mathcal{S}^C$ and the network employs a normalizing layer at the end in order force the output vector to lie on the $\mathcal{S}^C$. The ground-truth output vectors are

Framework	Desired Output of Network Lies on	Loss Function	Test Accuracy on MNIST (%)	Test Accuracy on CIFAR-10 (%)
Baseline		Cross Entropy	99.224 (0.0306)	78.685 (0.3493)
SNet-M	$\mathcal{S}^C$	$L_{S_{euc}}$	99.263 (0.0479)	79.738 (0.4009)
		$L_{S_{geo}}$	99.293 (0.0343)	**80.024** (0.5131)
SNet-TS	$T_{\mathbf{u}_S}\mathcal{S}^C$	$L_{T_{euc}}$	99.293 (0.0691)	77.548 (0.5620)
		$L_{T_{orth}}$	99.279 (0.0448)	77.708 (0.3517)
		$L_{T_{proj}}$	**99.332** (0.0600)	76.047 (1.6225)

Table 4. Avg test accuracy (std. dev.) over 10 runs using different loss functions on S^C and $T_{\mathbf{u}_S}S^C$, compared to the cross entropy loss.

obtained by using the square-root parametrization. The final 3 networks employ the SNet-TS framework and $L_{T_{euc}}$, $L_{T_{orth}}$ and $L_{T_{proj}}$ as the loss functions. The desired output vector, in this case, should lie on $T_{\mathbf{u}_S}\mathcal{S}^C$. The required logarithm and exponential maps are computed using the Manifold Optimization toolbox [6]. We note that the purpose of the experiments is to show that for some chosen network architecture, the proposed loss functions that are inspired by the geometry of the hypersphere, perform comparably with the cross-entropy loss function.

MNIST: The MNIST dataset [31] consists a total of 60000 images of hand-written digits (0-9). Each image is of size 28×28 and is in grayscale. The task is to classify each image into one of the 10 classes (0-9). The dataset is split into training and testing sets with 50000 and 10000 images respectively. We use the LeNet-5 architecture as the neural network [31]. The network consists of 2 convolutional and max-pooling (`conv`) layers followed by 2 fully-connected (`fc`) layers. ReLU non-linearity is employed. The filters are of size 5×5. The first and second `conv` layers produce 32 and 64 feature maps respectively. The first and second fc layers output 1024 and 10 elements respectively. The networks are trained for 50000 iterations with a batch size of 100 using Adam optimizer [27] with learning rate of 10^{-3}.

CIFAR-10: The CIFAR-10 dataset [29] consists a total of 60000 RGB natural images. Each image is of size 32×32. The task is to classify each image into one of the 10 classes (Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship, Truck). The dataset is split into training and testing sets with 50000 and 10000 images respectively. The network consists of 2 `conv` layers with max-pooling and local response normalization followed by 2 `fc` layers. ReLU non-linearity is employed. Each input image is mean subtracted and divided by its standard deviation. Data augmentation using 10 24×24 random crops per input image is employed to reduce overfitting. At test time, the central 24×24 region is used as the input to the network, after performing the same normalization as the training inputs. The networks are trained for 500000 iterations with a batch size of 100 using Adam optimizer with learning rate of 10^{-3}.

For each dataset, we fix the network architecture and train the 6 versions of the network with different loss function as described above. We use $\lambda = 1$ for $L_{T_{orth}}$. The image recognition accuracies obtained on the test set (averaged over 10 runs) are shown in Table 4.

The results indeed show that some of the proposed loss functions tend to perform better than cross entropy. For both datasets and especially CIFAR-10, SNet-M yields better performance than cross entropy and within this framework, geodesic loss performs better compared to Euclidean loss. SNet-TS shows improvements in accuracy in the case of MNIST, albeit with higher variance in the accuracy.

6. Conclusion

In this paper, we have studied the problem of learning invariant representations, which are at the heart of many computer vision problems, where invariance to physical factors such as illumination, pose, etc often lead to representations with non-Euclidean geometric properties. We have shown how deep learning architectures can be effectively extended to such non-linear target domains, exploiting the knowledge of data geometry. Through two specific examples – predicting illumination invariant representations which lie on the Grassmannian, and multi-class classification by mapping to a scale-invariant unit hypersphere representation – we have demonstrated how the power of deep networks can be leveraged and enhanced by making informed choices about the loss function while also enforcing the required output geometric constraints exactly. Extensions to other geometrically constrained representations, such as symmetric positive-definite matrices are evident. On the theoretical side, extending the current framework to applications where data points may have wider spread from their centroid, and to non-differentiable manifolds which arise in vision remain interesting avenues for the future.

Acknowledgements

This work was supported in part by ARO grant number W911NF-17-1-0293 and NSF CAREER award 1451263. We thank Qiao Wang and Rushil Anirudh for helpful discussions.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

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

[3] M. Banerjee, R. Chakraborty, E. Ofori, M. S. Okun, D. E. Viallancourt, and B. C. Vemuri. A nonlinear regression technique for manifold valued data with applications to medical image analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2016.

[4] M. Banerjee, R. Chakraborty, E. Ofori, D. Vaillancourt, and B. C. Vemuri. Nonlinear regression on riemannian manifolds and its applications to neuro-image analysis. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 719–727. Springer, 2015.

[5] E. Begelfor and M. Werman. Affine invariance revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2087–2094. IEEE, 2006.

[6] N. Boumal, B. Mishra, P.-A. Absil, R. Sepulchre, et al. Manopt, a matlab toolbox for optimization on manifolds. *Journal of Machine Learning Research*, 15(1):1455–1459, 2014.

[7] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 2017.

[8] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. *International Conference on Learning Representations*, 2014.

[9] A. Byravan and D. Fox. Se3-nets: Learning rigid body motion using deep neural networks. *IEEE International Conference on Robotics and Automation*, 2016.

[10] R. Clark, S. Wang, H. Wen, A. Markham, and N. Trigoni. Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem. *AAAI Conference on Artificial Intelligence*, 2017.

[11] A. de Brébisson and P. Vincent. An exploration of softmax alternatives belonging to the spherical loss family. *International Conference on Learning Representations*, 2016.

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

[13] R. Epstein, P. W. Hallinan, and A. L. Yuille. 5$\pm$2 eigenimages suffice: an empirical investigation of low-dimensional lighting models. In *Proceedings of the Workshop on Physics-Based Modeling in Computer Vision*, page 108. IEEE, 1995.

[14] P. T. Fletcher. Geodesic regression and the theory of least squares on Riemannian manifolds. *International Journal of Computer Vision*, 105(2):171–185, 2013.

[15] T. Fletcher. Geodesic regression on Riemannian manifolds. In *Proceedings of the Third International Workshop on Mathematical Foundations of Computational Anatomy-Geometrical and Statistical Methods for Modelling Biological Shape Variability*, pages 75–86, 2011.

[16] A. S. Georghiades, P. N. Belhumeur, and D. J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):643–660, 2001.

[17] P. W. Hallinan. A low-dimensional representation of human faces for arbitrary lighting conditions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1994.

[18] J. Hamm and D. D. Lee. Grassmann discriminant analysis: a unifying view on subspace-based learning. In *Proceedings of the International Conference on Machine Learning*, pages 376–383. ACM, 2008.

[19] M. Harandi and B. Fernando. Generalized backpropagation,\'{E} tude de cas: Orthogonality. *arXiv preprint arXiv:1611.05927*, 2016.

[20] M. Henaff, J. Bruna, and Y. LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[21] Y. Hong, R. Kwitt, N. Singh, B. Davis, N. Vasconcelos, and M. Niethammer. Geodesic regression on the Grassmannian. In *European Conference on Computer Vision*, pages 632–646. Springer, 2014.

[22] Z. Huang and L. Van Gool. A Riemannian network for SPD matrix learning. *AAAI Conference on Artificial Intelligence*, 2017.

[23] Z. Huang, C. Wan, T. Probst, and L. Van Gool. Deep learning on Lie groups for skeleton-based action recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[24] Z. Huang, J. Wu, and L. Van Gool. Building deep networks on Grassmann manifolds. *arXiv preprint arXiv:1611.05742*, 2016.

[25] S. Jetley, N. Murray, and E. Vig. End-to-end saliency mapping via probability distribution prediction. In *2016 IEEE Conference on Computer Vision and Pattern Recognition*, pages 5753–5761, 2016.

[26] H. J. Kim, N. Adluru, M. D. Collins, M. K. Chung, B. B. Bendlin, S. C. Johnson, R. J. Davidson, and V. Singh. Multivariate general linear models (mglm) on Riemannian manifolds with applications to statistical analysis of diffusion weighted images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2705–2712, 2014.

[27] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

[28] A. V. Knyazev and M. E. Argentati. Principal angles between subspaces in an a-based scalar product: algorithms and per-

turbation estimates. *SIAM Journal on Scientific Computing*, 23(6):2008–2040, 2002.

[29] A. Krizhevsky. Learning multiple layers of features from tiny images. *Technical Report*, 2009.

[30] J. D. Lafferty and G. Lebanon. Diffusion kernels on statistical manifolds. *Journal of Machine Learning Research*, 6:129–163, 2005.

[31] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[32] Y. M. Lui. Advances in matrix manifolds for computer vision. *Image and Vision Computing*, 30(6):380–388, 2012.

[33] J. Masci, D. Boscaini, M. Bronstein, and P. Vandergheynst. Geodesic convolutional neural networks on Riemannian manifolds. In *Proceedings of the IEEE International Conference on Computer Vision workshops*, pages 37–45, 2015.

[34] M. Niepert, M. Ahmed, and K. Kutzkov. Learning convolutional neural networks for graphs. In *Proceedings of the International Conference on Machine Learning*, 2016.

[35] P. Paysan, R. Knothe, B. Amberg, S. Romdhani, and T. Vetter. A 3d face model for pose and illumination invariant face recognition. In *Sixth IEEE International Conference on Advanced video and signal based surveillance*, pages 296–301. IEEE, 2009.

[36] X. Pennec, P. Fillard, and N. Ayache. A Riemannian framework for tensor computing. *International Journal of Computer Vision*, 66(1):41–66, 2006.

[37] X. Shi, M. Styner, J. Lieberman, J. G. Ibrahim, W. Lin, and H. Zhu. Intrinsic regression models for manifold-valued data. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 192–199. Springer, 2009.

[38] A. Srivastava, I. Jermyn, and S. Joshi. Riemannian analysis of probability density functions with applications in vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[39] A. Srivastava and E. Klassen. Bayesian and geometric subspace tracking. *Advances in Applied Probability*, 36(01):43–56, 2004.

[40] A. Srivastava and P. Turaga. *Riemannian computing in computer vision*. Springer International Publishing, 1 2015.

[41] S. Taheri, P. Turaga, and R. Chellappa. Towards view-invariant expression analysis using analytic shape manifolds. In *IEEE International Conference on Automatic Face & Gesture Recognition and Workshops*, pages 306–313. IEEE, 2011.

[42] P. Turaga, A. Veeraraghavan, A. Srivastava, and R. Chellappa. Statistical computations on Grassmann and Stiefel manifolds for image and video-based recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(11):2273–2286, 2011.

[43] R. Vemulapalli, F. Arrate, and R. Chellappa. Human action recognition by representing 3D skeletons as points in a Lie group. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 588–595, 2014.

[44] R. Vemulapalli and R. Chellapa. Rolling rotations for recognizing human actions from 3D skeletal data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4471–4479, 2016.

[45] P. Vincent, A. de Brébisson, and X. Bouthillier. Efficient exact gradient update for training deep networks with very large sparse targets. In *Advances in Neural Information Processing Systems*, pages 1108–1116, 2015.