Siamese Networks: The Tale of Two Manifolds

Soumava Kumar Roy[1,4], Mehrtash Harandi[2,4], Richard Nock[1,3,4], Richard Hartley[1]
[1]The Australian National University; [2]Monash University;
[3]The University of Sydney; [4]DATA61-CSIRO, Australia
Soumava.KumarRoy@anu.edu.au, Mehrtash.Harandi@monash.edu,
Richard.Nock@data61.csiro.au, Richard.Hartley@anu.edu.au

Abstract

Siamese networks are non-linear deep models that have found their ways into a broad set of problems in learning theory, thanks to their embedding capabilities. In this paper, we study Siamese networks from a new perspective and question the validity of their training procedure. We show that in the majority of cases, the objective of a Siamese network is endowed with an invariance property. Neglecting the invariance property leads to a hindrance in training the Siamese networks. To alleviate this issue, we propose two Riemannian structures and generalize a well-established accelerated stochastic gradient descent method to take into account the proposed Riemannian structures. Our empirical evaluations suggest that by making use of the Riemannian geometry, we achieve state-of-the-art results against several algorithms for the challenging problem of fine-grained image classification.

1. Introduction

Siamese networks, introduced in 90s by Bromley [4], are ubiquitous in machine learning and one can find their trace in similarity/metric learning [15, 40, 43, 45], hashing [37], clustering [28], and zero/one/few shot learning [21, 52, 65]. Perhaps a glimpse at computer vision literature can provide a better picture as to why Siamese networks are essential these days. Siamese networks have been successfully employed to address face recognition/verification [5, 17, 49, 47, 41], person re-identification [10, 54], image/patch/point descriptors [64, 25, 42, 63], localization [51, 32], image retrieval [12], stereo matching [35], Deep Metric Learning (DML) [8, 33, 53, 61, 22] and even object tracking [50, 2, 44].

In majority of cases, a Siamese Neural Network (SiNN) realizes a non-linear embedding of data with the objective being to *attain a semantically meaningful space where related patterns (*e.g.*, faces of the same person) are close to each other while proximity of semantically-unrelated patterns (*e.g.*, faces of different people) are avoided*. The non-linear embedding is accomplished in two stages; **1.** Feature extraction performed by two sister networks (usually with shared weights), **2.** followed by an embedding parameterized by a positive semi-definite matrix.

Previous studies on Siamese networks mainly focus on the following challenges;

1. **Design Challenges.** For a given problem, what is the right design of the network?
2. **Training Challenges.** It may sound surprising but as evidenced by a large body of works (*e.g.*, [41, 40, 43, 45, 61, 26]), proper sampling/mining of data plays a crucial role in successful training of Siamese networks. We will discuss this point in more details later in the paper.

In this work, we will address an important and to a great extent a neglected issue in training Siamese networks. We will show that the objective function of a Siamese network comes with a form of invariance. As a result, the geometry of the search space is no longer Euclidean. Studies in the field of Riemannian geometry suggest that better outcomes can be yield if the true geometry of the search space is taken into account during the optimization [1, 19]. As such, in this work we develop tools and algorithms to take into account the invariance in Siamese networks. Our contributions in this work are

- We propose and develop a novel matrix manifold and its associated manifold operations that offer invariance to the structure of siamese networks.
- As an alternative solution, we show that the required invariance can also be attained by using a Stiefel manifold to model the true geometry of the search space.
- We also make another contribution by incorporating a Riemannian form of the stochastic gradient descent with momentum into the backprop algorithm. This is required to preserve the geometry during training the network.

- Lastly, we show that by exploiting the true geometry in training Siamese networks, significant improvements can be achieved for the task of categorizing unseen fine-grained classes (*e.g.*, birds or cars). We achieve this by employing the simplest structures, demonstrating the importance of proper training in comparison to sophisticated models that make use of spectral clustering [28] for example.

Notations: Throughout this paper, we use bold lower-case letters (*e.g.*, $\boldsymbol{x}$) to show column vectors and bold upper-case letters (*e.g.*, $\boldsymbol{X}$) to show matrices.$[\cdot]_i$ is used to denote the i-th element of a vector and $\mathbf{I}_n$ shows the $n \times n$ identity matrix. The Frobenius norm of a matrix is shown by $\|\boldsymbol{X}\|_F = \sqrt{\text{Tr}(\boldsymbol{X}^\top \boldsymbol{X})}$, with $\text{Tr}(\cdot)$ indicating the matrix trace. The set of full rank tall matrices of size $n \times p$ and the orthogonal group are shown by $\mathbb{R}_*^{n\times p}$ and $\mathcal{O}(p) := \{\boldsymbol{R} \in \mathbb{R}^{p\times p} | \ \boldsymbol{R}^\top \boldsymbol{R} = \boldsymbol{R}\boldsymbol{R}^\top = \mathbf{I}_p\}$, respectively. A matrix $\mathbb{R}^{p\times p} \ni \boldsymbol{M} \succeq \mathbf{0}$ is Positive Semi-Definite (PSD) if $\boldsymbol{M} = \boldsymbol{M}^\top$ and for any non-zero vector $\boldsymbol{x} \in \mathbb{R}^p$ we have $\boldsymbol{x}^\top \boldsymbol{M}\boldsymbol{x} \geq 0$. A matrix $\boldsymbol{M} \succ \mathbf{0}$ is Positive Definite (PD) if it is PSD and for any non-zero vector $\boldsymbol{x} \in \mathbb{R}^p$ we have $\boldsymbol{x}^\top \boldsymbol{M}\boldsymbol{x} > 0$.

2. Siamese Networks

As mentioned before, an SiNN makes use of a twin network to realize a non-linear embedding ϕ (usually a Deep Neural Network) from its input domain $\mathcal{X}$ (*e.g.*, images) to some Euclidean spaces $\mathbb{R}^n$, *i.e.*, $\phi : \mathcal{X} \to \mathbb{R}^n$. The non-linear embedding is accomplished in two stages; **1.** Feature extraction performed by two sister networks; followed by, **2.** learning a discriminative embedding space parameterized by $\mathbb{R}^{n\times n} \ni \boldsymbol{M} \succeq \mathbf{0}$. $\boldsymbol{M}$ is constrained to be a PSD matrix in order to realize a valid Mahalanobis distance $d^2_{\boldsymbol{M}}(\boldsymbol{x}_i,\boldsymbol{x}_j)=(\boldsymbol{x}_i\text{-}\boldsymbol{x}_j)^\top \boldsymbol{M}(\boldsymbol{x}_i\text{-}\boldsymbol{x}_j)$, with $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{X}$. Thus one can factorize $\boldsymbol{M}\text{=}\boldsymbol{L}\boldsymbol{L}^\top$ and learn $\boldsymbol{L}$ using an SiNN to accomplish the aforementioned factorization.

In its original form [4], the sister networks are identical (parameterized by Θ). The embedding part identifies a metric in the resulting space to increase the discriminatory power of the whole system. Such a construction enjoys two key properties:

- It guarantees the consistency of its predictions. Weight sharing ensures that two similar samples will not map to different parts of the embedded space as each leg uses the same functionality.
- The network is symmetric, meaning that it does not matter how an input pair is fed to the network.

Our focus here is on the embedding part but stress that the developments done in this work can be applied verbatim to the cases where the weight-sharing between the sister networks are removed intentionally.

To learn the parameters $(\Theta, \boldsymbol{M})$ of a SiNN, one makes use of the PSD property of $\boldsymbol{M}$ and factorize it into $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{L}^\top$ with $\boldsymbol{L} \in \mathbb{R}_*^{n\times p}$, $p \leq n$. Such factorization is essential in SiNN as the PSD constraint can be enforced implicitly. This is very important as the Gradient Descent (GD) updating scheme used in the BackPropagation (BP) [30] algorithm cannot preserve any form of constraints.

Furthermore, the factorization $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{L}^\top$ can be thought of performing joint dimensionality reduction and metric learning over the outputs of the sister networks. To see this, consider the Singular Value Decomposition (SVD) of $\boldsymbol{L} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top = \boldsymbol{U}\boldsymbol{B}$ with $\boldsymbol{U}^\top\boldsymbol{U} = \mathbf{I}_p$ and $\boldsymbol{B} = \boldsymbol{D}\boldsymbol{V}^\top$. It is now clear that the embedding part of a SiNN first performs a dimensionality reduction using $\boldsymbol{U}$ followed by scaling/shrinking and rotating the resulting low-dimensional space to increase its discriminatory power.

Remark 1. *We note that to have a valid metric, one should enforce $\boldsymbol{M}$ to be positive definite,* i.e., *$\boldsymbol{M} \succ \mathbf{0}$. However, learning a metric in high-dimensional spaces is not without difficulties. For one, the number of training samples grows exponentially with the dimensionality. For example, the high-level features extracted from deep models in computer vision (*e.g., *AlexNet [24], Inception [48] or ResNet [13]) are usually very high-dimensional. As such, SiNNs that use such models as backbone opt for semi-definite solutions (*e.g., *[40, 43, 45, 28]).*

3. Problem Statement

Despite its wide adoption, factorization $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{L}^\top$ comes with an undesirable property as a result of a symmetry. To be specific, changing $\boldsymbol{L}$ to $\boldsymbol{L}\boldsymbol{R}$ for $\boldsymbol{R} \in \mathcal{O}(p)$ does not change $\boldsymbol{M}$. In other words, there exists an equivalence class of solutions for every $\boldsymbol{M}$, which in return makes the search space non-Euclidean. This is where we question the common practice in training SiNNs and make our contributions.

To give the reader an intuition as to why one needs to consider the invariance during the optimization, consider a simple example. Assume you want to minimize a function $f : \mathbb{R}^2 \to \mathbb{R}$ but your function comes with an invariance property in the form $f(\boldsymbol{x}) = f(\boldsymbol{R}\boldsymbol{x})$ for $\boldsymbol{R} \in \mathcal{O}(2)$. Such an invariance drastically changes the search space from $\mathbb{R}^2$ to $\mathbb{R}^+$ as circles centered at origin are equivalent.

Furthermore, recent studies in the context of low-rank matrix approximation suggest that explicitly considering the geometry resulted by such a symmetry is the key to faster optimizations and better outcomes [20, 38, 39]. In what follows, we first discuss how the updating scheme in the BP algorithm should be altered to preserve certain constraints. This is required as we later develop two geometries to explicitly take on board the invariance to the action of the orthogonal group during training a deep network.

4. Related Work

As mentioned in § 1, Siamese networks are used in a large body of works to address various problems in machine learning, computer vision, speech processing and related fields. Since we are mainly interested in theoretical aspects of Siamese nets, we briefly review recent advances in Siamese nets and in particular focus on the literature of deep metric/similarity learning [41, 40, 43, 18, 53, 45, 28, 61].

Traditionally, Siamese networks were trained with the so called *"contrastive loss"* [4, 5]. The main idea here is to construct a Euclidean space where positive pairs (*i.e.*, samples sharing the same class label) are close to each other while simultaneously negative pairs (*i.e.*, samples coming from different classes) are pushed away. Models trained by the contrastive loss require a weaker form of supervision. This is because only the knowledge about pairs (positive/negative) is needed. On the downside, models trained by the contrastive loss can be considered as holistic solutions, meaning that the focus is on absolute distances, where relative distances matter more in many cases.

Exploiting relative distances and local information in metric learning can be attributed to the seminal work of Weinberger and Saul [59] where the concept of triplets is introduced. In a nutshell, instead of creating positive and negative pairs blindly, for any training example $\boldsymbol{x}_a$ (called the anchor point), some neighbor samples are chosen for training. From this, one creates triplets in the form $\big(\boldsymbol{x}_a, \boldsymbol{x}_+, \boldsymbol{x}_-\big)$ where $\boldsymbol{x}_+$ is a neighbor point sharing the same label as $\boldsymbol{x}_a$ with $\boldsymbol{x}_-$ being a neighbor from a different class. The goal of training is to learn a metric $\boldsymbol{M}$ such that for all triplets

$$d^2_{\boldsymbol{M}}\big(\boldsymbol{x}_a, \boldsymbol{x}_+\big) \leq d^2_{\boldsymbol{M}}\big(\boldsymbol{x}_a, \boldsymbol{x}_-\big) + \tau \ ,$$

with τ being a predefined margin, and $d^2_{\boldsymbol{M}}\big(\boldsymbol{x}, \boldsymbol{y}\big) = \big(\boldsymbol{x} - \boldsymbol{y}\big)^\top \boldsymbol{M}\big(\boldsymbol{x} - \boldsymbol{y}\big)$. Clearly the model does not consider well separated negative pairs during training. Moreover, only neighbor positive pairs are used for learning, meaning positive pairs that are very far from each other will never affect the learning. Nowadays, the concept of triplets along its variants (*e.g.*, quadruplets [27]) is the method of choice when for training Siamese networks as it is believed to create more discriminative and robust models. As of lately, *mining* forms a central theme in deep metric learning [41, 40, 43, 61, 26]. Some lessons learned from successful models are

- **Uniform Class Distribution.** Each mini-batch should contain the same number of samples per class [41, 40, 43, 18, 53, 45, 28].

- **Normalization.** It is helpful to normalize the output of the sister networks [41, 40, 45, 28]. We note that in [43], instead of normalization, authors use an ℓ_2 regularization penalty.

- **Semi-Hard Negative Mining.** The idea introduced in [41] constructs triplets by finding a "semi-hard" negative example. Given the anchor $\boldsymbol{x}_a$ and its associated positive sample $\boldsymbol{x}_+$, a semi-hard negative sample is one with the property that $d^2_M\big(\boldsymbol{x}_a, \boldsymbol{x}_+\big) \leq d^2_M\big(\boldsymbol{x}_a, \boldsymbol{x}_-\big)$ but is still the hardest negative, meaning among all negative samples in the mini-batch satisfying the aforementioned distance criterion, it has the smallest to the anchor.

- **Classification Loss.** To achieve state-of-the-art, several studies benefit from the classification loss. Examples include **1.** pretraining for the classification task, followed by fine-tuning only the top layers with the metric loss [49] and **2.** combining the metric loss with the classification loss as in [46].

Showing success in face recognition/verification, a number of algorithms [34, 7, 57, 56] make use of softmax based loss functions within DNNs.We note that the softmax based algorithms are designed for closed-set problems, while DML and SiNNs are efficient in learning from unseen classes or limited data [62, 21, 52]. Moreover, as Horiguchi *et al.* [16] demonstrate that softmax-based and DML solutions cannot replace/eradicate one another.

5. Geometry-Aware Layers[1]

Inspired by studies on the geometry of low-rank matrix decomposition [20, 39], in this section we propose two novel layers, namely the quotient Convolutional layer (qConv for short) and the Stiefel layer to address the invariance property of the Siamese networks. On the other hand, we make use of a form of Riemannian stochastic gradient descent algorithm to preserve the structures envisaged on the qConv and Stiefel layers during the training phase. This, ultimately, boils down to equipping the BP algorithm with two extra operators, namely **projection onto the tangent space** $\Pi_{\boldsymbol{X}} : \mathbb{R}^{n\times p} \to T_{\boldsymbol{X}}\mathcal{M}$ and **retraction** $\Upsilon_{\boldsymbol{X}} : T_{\boldsymbol{X}}\mathcal{M} \to \mathcal{M}$ with $T_{\boldsymbol{X}}\mathcal{M}$ denoting the tangent space of the Riemannian manifold $\mathcal{M}$ at $\boldsymbol{X}$.

5.1. qConv Layer

Consider a (right) action of a Lie-group G on a a manifold $\overline{\mathcal{M}}$. We denote the action of group element g on point $\boldsymbol{x} \in \overline{\mathcal{M}}$ by $\boldsymbol{x}\, g$. (For instance as discussed, the Lie-group $\mathcal{O}(p)$ acts on the manifold $\mathbb{R}^{n\times p}_*$ by ordinary matrix multiplication on the right). The orbit of a point $\boldsymbol{x}$ is the set $\mathrm{Orb}(\boldsymbol{x}) = \{\boldsymbol{x}\, g \mid g \in G\}$, sometimes written as $[\boldsymbol{x}]$.

Under the conditions of Theorem 1 below, the set of orbits, denoted by $\overline{\mathcal{M}}/G$, forms a smooth manifold.

[1]The code is available at https://github.com/sumo8291/Siamese-Networks.git.

Theorem 1 (Quotient Manifold Theorem (Theorem 21.10 in [31])**.** *Suppose that a Lie group G acts* **smoothly**, **freely** *and* **properly** *on a smooth manifold $\overline{\mathcal{M}}$. Then $\overline{\mathcal{M}}/G$ is a topological manifold of dimension $\dim(\overline{\mathcal{M}}) - \dim(G)$, and has a unique smooth structure, with the property that the quotient map is a smooth submersion.*

We denote the quotient mapping by $\pi : \overline{\mathcal{M}} \to \mathcal{M} = \overline{\mathcal{M}}/G$.The inverse-image $\pi^{-1}([\boldsymbol{x}])$ for some point $[\boldsymbol{x}] \in \mathcal{M}$, is called the *fibre* of $[\boldsymbol{x}]$. The differential of this mapping at a point $p \in \overline{\mathcal{M}}$ is denoted by $d\pi_p : T_p(\overline{\mathcal{M}}) \to T_{\pi(p)}(\mathcal{M})$. The statement that π is a smooth submersion means that the mapping $d\pi_p$ is a surjection at any point p.

Recall the factorization $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{L}^\top$, widely used in training SiNN, is invariant to the action of orthogonal group, *i.e.*, by changing $\boldsymbol{L}$ to $\boldsymbol{L}\boldsymbol{R}$, $\boldsymbol{R} \in \mathcal{O}(p)$ the metric $\boldsymbol{M}$ is not going to change. Based on the previous discussion, the true geometry of the search space can be identified as the quotient of $\mathbb{R}_*^{n\times p}$. To develop the Riemannian geometry of $\mathcal{M} = \mathbb{R}_*^{n\times p}/\mathcal{O}(p)$ and eventually obtain $\Pi_{\cdot}(\cdot)$ and $\Upsilon_{\cdot}(\cdot)$, we start by defining a Riemannian metric on $\mathbb{R}_*^{n\times p}$:

$$\bar{g}_{\boldsymbol{L}}(\bar{\xi}, \bar{\vartheta}) := \mathrm{Tr}\left(\bar{\xi}\left(\boldsymbol{L}^\top\boldsymbol{L}\right)^{-1}\bar{\vartheta}^\top\right) = \left\langle \bar{\xi}, \bar{\vartheta}\left(\boldsymbol{L}^\top\boldsymbol{L}\right)^{-1}\right\rangle_F . \tag{1}$$

where $\langle\cdot,\cdot\rangle_F$ denotes the Frobenius inner product. Here $\bar{\xi}, \bar{\vartheta} \in T_{\boldsymbol{L}}\mathcal{M} = \mathbb{R}^{n\times p}$. This metric has the important property that it is right-invariant under the action of $\mathcal{O}(p)$, meaning $\bar{g}_{\boldsymbol{L}}(\bar{\xi}, \bar{\vartheta}) = \bar{g}_{\boldsymbol{L}\boldsymbol{R}}(\bar{\xi}\boldsymbol{R}, \bar{\vartheta}\boldsymbol{R})$.It should be noted that there are other **choices** of right-invariant metric. For instance, one can define $\bar{g}_{\boldsymbol{L}}(\bar{\xi}, \bar{\vartheta}) = \langle \bar{\vartheta}, \bar{\xi}\rangle_F$, which is also right-invariant. However, the chosen metric has the attractive feature that it is scale invariant, namely $\bar{g}_{s\boldsymbol{L}}(s\bar{\xi}, s\bar{\vartheta}) = \bar{g}_{\boldsymbol{L}}(\bar{\xi}, \bar{\vartheta})$.With this inner product, the tangent space $T_{\boldsymbol{L}}\overline{\mathcal{M}}$ at $\boldsymbol{L}$ can be split into two complementary parts, namely the horizontal space $\mathcal{H}_{\boldsymbol{L}}\overline{\mathcal{M}}$ and the vertical space $\mathcal{V}_{\boldsymbol{L}}\overline{\mathcal{M}}$. That is, $T_{\boldsymbol{L}}\mathbb{R}_*^{n\times p} = \mathcal{H}_{\boldsymbol{L}}\mathbb{R}_*^{n\times p} \oplus \mathcal{V}_{\boldsymbol{L}}\mathbb{R}_*^{n\times p}$. The vertical space is defined as the kernel of the differential map: $\mathcal{V}_{\boldsymbol{L}}\overline{\mathcal{M}} = \ker(d\pi_{\boldsymbol{L}})$, and the horizontal space $\mathcal{H}_{\boldsymbol{L}}\overline{\mathcal{M}}$ is its orthogonal complement with respect to the metric $\bar{g}(\cdot,\cdot)$.

Lemma 1. *The vertical space $\mathcal{V}_{\boldsymbol{L}}\overline{\mathcal{M}}$ at $\boldsymbol{L} \in \mathbb{R}_*^{n\times p}$ is equal to the set $\{\boldsymbol{L}\Lambda \mid \Lambda^\top = -\Lambda\}$. This is a vector space of dimension $p(p-1)/2$.*

Proof. We envisage the tangent space $T_{\boldsymbol{L}}\overline{\mathcal{M}}$ as the vector space of all matrices $\{\gamma'(0)\}$ where $\gamma(t) : [-1, 1] \to \mathbb{R}^{n\times p}$ is a smoothcurve with $\gamma(0) = \boldsymbol{L}$. In other words, $T_{\boldsymbol{L}}\overline{\mathcal{M}}$ is the set of derivatives, at $\boldsymbol{L}$, of smooth curves.Consider the curve $\gamma(t) = \boldsymbol{L}\boldsymbol{R}(t)$ where $\boldsymbol{R}(t) \in \mathcal{O}(p)$ and $\boldsymbol{R}(0)$ is the identity. Note that $\gamma'(0) = \boldsymbol{L}\Lambda$ for some skew-symmetric matrix Λ. Under the projection $\pi : \mathbb{R}^{n\times p} \to \mathcal{M}$, we see that $\pi(\boldsymbol{L}\boldsymbol{R}(t)) = \pi(\boldsymbol{L})$, a constant curve. Consequently, $\gamma'(0)$ is in the kernel of $d\pi_{\boldsymbol{L}} : T_{\boldsymbol{L}}\overline{\mathcal{M}} \to T_{\pi(\boldsymbol{L})}\mathcal{M}$. Note that the set of matrices $\{\boldsymbol{L}\Lambda \mid \Lambda^\top = -\Lambda\}$ forms a vector space of dimension $p(p-1)/2 = \dim(\mathcal{O}(p))$. Furthermore,

$$\begin{aligned}\dim(\ker(d\pi_{\boldsymbol{L}})) &= \dim(T_{\boldsymbol{L}}\overline{\mathcal{M}}) - \dim(T_{\pi(\boldsymbol{L})}\mathcal{M}) \\ &= \dim(\overline{\mathcal{M}}) - \dim(\mathcal{M}) = \dim(\mathcal{O}(p)) .\end{aligned}$$

Here, the first equality is true, because π is a submersion, and the last equality also follows from Theorem 1. It follows that $\{\boldsymbol{L}\Lambda\}$ forms the whole of $\mathcal{V}_{\boldsymbol{L}}\overline{\mathcal{M}} = \ker(\pi^*)$, as required. □

Lemma 2. *The horizontal space $\mathcal{H}_{\boldsymbol{L}}\overline{\mathcal{M}}$ with respect to the metric $\bar{g}$ is equal to*

$$\mathcal{H}_{\boldsymbol{L}}\overline{\mathcal{M}} = \{\bar{\zeta} \in T_{\boldsymbol{L}}\overline{\mathcal{M}} \mid \bar{\zeta}^\top\boldsymbol{L}\left(\boldsymbol{L}^\top\boldsymbol{L}\right) = \left(\boldsymbol{L}^\top\boldsymbol{L}\right)\boldsymbol{L}^\top\bar{\zeta}\} .$$

Proof. Assume $\bar{\zeta} \in \mathcal{H}_L$. Then, $\bar{\zeta}$ is perpendicular to any $\bar{\xi} = \boldsymbol{L}\Lambda \in \mathcal{V}_{\boldsymbol{L}}$, so

$$0 = \bar{g}_{\boldsymbol{L}}(\bar{\zeta}, \bar{\xi}) = \mathrm{Tr}\left(\left(\boldsymbol{L}^\top\boldsymbol{L}\right)^{-1}\bar{\zeta}^\top\ \boldsymbol{L}\Lambda\right) .$$

Since this must hold for all skew-symmetric Λ, it follows that $\left(\boldsymbol{L}^\top\boldsymbol{L}\right)^{-1}\bar{\zeta}^\top\boldsymbol{L}$ is symmetric, so $\left(\boldsymbol{L}^\top\boldsymbol{L}\right)^{-1}\bar{\zeta}^\top\boldsymbol{L} = \boldsymbol{L}^\top\bar{\zeta}\left(\boldsymbol{L}^\top\boldsymbol{L}\right)^{-1}$. This shows that $\bar{\zeta}^\top\boldsymbol{L}\left(\boldsymbol{L}^\top\boldsymbol{L}\right) = \left(\boldsymbol{L}^\top\boldsymbol{L}\right)\boldsymbol{L}^\top\bar{\zeta}$. The converse follows similarly. □

The argument of Lemma 1 shows that the differential map $d\pi_{\boldsymbol{L}}$ maps the horizontal space $\mathcal{H}_{\boldsymbol{L}}\overline{\mathcal{M}}$ isomorphically onto the tangent space $T_{\pi(\boldsymbol{L})}\mathcal{M}$, which allows us to identify the tangent space at a point in $\mathcal{M}$ with the horizontal space at any point in its fibre.

Theorem 2. *Let $\bar{\xi} \in T_{\boldsymbol{L}}\overline{\mathcal{M}} = \mathbb{R}^{n\times p}$. The horizontal part of $\bar{\xi}$ at $\boldsymbol{L}$ is given by $\bar{\xi}^\uparrow = \bar{\xi} + \boldsymbol{L}\Lambda$ with $\Lambda \in \mathbb{R}^{p\times p}$ being the solution of the following Sylvester equation,*

$$\Lambda\left(\boldsymbol{L}^\top\boldsymbol{L}\right)^2 + \left(\boldsymbol{L}^\top\boldsymbol{L}\right)^2\Lambda = \bar{\xi}^\top\boldsymbol{L}\boldsymbol{L}^\top\boldsymbol{L} - \boldsymbol{L}^\top\boldsymbol{L}\boldsymbol{L}^\top\bar{\xi} . \tag{2}$$

Proof. Let the projection of $\bar{\xi}$ onto the horizontal subspace be $\bar{\zeta}$. Then, $\bar{\zeta} - \bar{\xi}$ is a vertical vector, so $\bar{\zeta} = \bar{\xi} + \boldsymbol{L}\Lambda$, by Lemma 1. Since $\bar{\zeta}$ is in the horizontal subspace, the condition in Lemma 2 holds. Substituting $\bar{\zeta} = \bar{\xi} + \boldsymbol{L}\Lambda$ into the equation $\bar{\zeta}^\top\boldsymbol{L}\left(\boldsymbol{L}^\top\boldsymbol{L}\right) = \left(\boldsymbol{L}^\top\boldsymbol{L}\right)\boldsymbol{L}^\top\bar{\zeta}$ gives the required result. □

The above development provides us with the key to our optimization update step for $\boldsymbol{L}$. At a current point $\boldsymbol{L}$, an update direction $\bar{\xi} \in \mathbb{R}^{n\times p} = T_{\boldsymbol{L}}\mathbb{R}_*^{n\times p}$ is computed. Instead of making an update in this direction, it is first projected to a vector $\bar{\xi}^\uparrow$ in the horizontal subspace $\mathcal{H}_{\boldsymbol{L}}\overline{\mathcal{M}}$ according to Theorem 2, and an update is made in this horizontal direction. Thus, our retraction operator is defined by

$$\Upsilon_{\boldsymbol{L}}(\bar{\xi}) = \boldsymbol{L} + \bar{\xi}^\uparrow . \tag{3}$$

Note that the projection operator $\Pi : \mathbb{R}^{n\times p} \to T_{\boldsymbol{L}}\overline{\mathcal{M}}$ is trivial, since $T_{\boldsymbol{L}}\overline{\mathcal{M}}$ is identified with $\mathbb{R}^{n\times p}$.

Computational Complexity.

The complexity of an update for the qConv depends on the computational cost of the projection on the horizontal space.

- **Forming the Sylvester equation.** To form the Sylvester equation, we need to compute $\boldsymbol{L}^\top \boldsymbol{L}$, $\left(\boldsymbol{L}^\top \boldsymbol{L}\right)^2$, $\Lambda\left(\boldsymbol{L}^\top \boldsymbol{L}\right)^2$, $\bar{\xi}^\top \boldsymbol{L}$ and $\bar{\xi}^\top \boldsymbol{L}(\boldsymbol{L}^\top \boldsymbol{L})$. This adds up to $2np^2 + 3p^3$ flops.
- **Solving the Sylvester equation.** Solving the Sylvester equation has a complexity of p^3.
- **Obtaining the horizontal vector.** This step needs a matrix multiplication and has a complexity of np^2.

All in all, an update of qConv demands $3np^2 + 4p^3$ extra flops. We note that this complexity is linear in n and all the steps can be done in a GPU.

5.2. Stiefel Layer

Instead of the factorization considered in § 5.1, we can make use of the Singular Value Decomposition to yield $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{U}^\top = \boldsymbol{U}\boldsymbol{D}^{1/2}\boldsymbol{D}^{\top/2}\boldsymbol{U}^\top = \boldsymbol{U}\boldsymbol{S}\boldsymbol{S}^\top\boldsymbol{U}^\top$. Here, $\boldsymbol{U} \in \mathbb{R}^{n\times p}$ is a matrix with the property that $\boldsymbol{U}^\top\boldsymbol{U} = \mathbf{I}_p$ and $\boldsymbol{S} = \boldsymbol{D}^{1/2} \in \mathbb{R}^{p\times p}$ is a diagonal matrix. The benefit of working with $\boldsymbol{S}$ is that unlike $\boldsymbol{D}^{1/2}$, diagonal elements of $\boldsymbol{S}$ can become negative during optimization. This however does not make the resulting metric $\boldsymbol{M} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{S}^\top\boldsymbol{U}^\top$ indefinite.

The advantage of the SVD factorization is in its uniqueness. To be precise, SVD is invariant to the permutation of the columns of its factors. This however is a very mild condition and as evidenced by our experiments can be neglected. As such, we can replace the embedding layer in a SiNN with two layers, one being a layer with orthogonal weights, followed by a layer that only scales its inputs according to $\boldsymbol{S}$. While no especial care is required for the layer that encodes the matrix $\boldsymbol{S}$, we need to enforce orthogonality on the $\boldsymbol{U}$ layer. This can be achieved by making use of the geometry of the Stiefel manifold. To be more specific, let us formally define the Stiefel manifold.

Definition 1 (The Stiefel Manifold)**.** *The set of $(n \times p)$-dimensional matrices, $p \leq n$, with orthonormal columns endowed with the Frobenius inner product forms a compact Riemannian manifold called the Stiefel manifold* $\mathrm{St}(p, n)$ *[9].*

$$\mathrm{St}(p, n) \triangleq \{\boldsymbol{U} \in \mathbb{R}^{n\times p} : \boldsymbol{U}^\top\boldsymbol{U} = \mathbf{I}_p\} \ . \tag{4}$$

Similar to the quotient geometry developed above, we need the knowledge of the orthogonal projection and the retraction on $\mathrm{St}(p, n)$. The form of orthogonal projection reads as [1]

$$\Pi_{\boldsymbol{U}}\left(\boldsymbol{v}\right) = \boldsymbol{v} - \boldsymbol{U}\mathrm{sym}\Big(\boldsymbol{U}^T\boldsymbol{v}\Big) \ . \tag{5}$$

In Eq. (5), $\mathrm{sym}(\boldsymbol{A}) = \frac{1}{2}(\boldsymbol{A} + \boldsymbol{A}^T)$. Various forms of retraction are defined on $\mathrm{St}(p, n)$ [1]. Among them, we recommend the following retraction

$$\Upsilon_{\boldsymbol{U}}\left(\xi\right) = \mathrm{qf}(\boldsymbol{U} + \xi) \ . \tag{6}$$

Here, $\mathrm{qf}(\boldsymbol{A})$ is the adjusted $\boldsymbol{Q}$ factor of the QR decomposition [11]. In practice, to obtain $\mathrm{qf}(\cdot)$, one performs QR decomposition followed by swapping the sign of elements of all columns whose corresponding diagonal elements in $\boldsymbol{R}$ are negative.

Remark 2 (Cayley Transform)**.** *Preserving orthogonality can also be attained using the Cayley transform [60]. The Cayley transform is indeed a valid form of retraction on the Stiefel manifold. The retraction provided in Eq.* (6) *is however computationally cheaper and hence preferable.*

Computational Complexity.

The complexity of an update for the Stiefel structure depends on the computational cost of the following major steps:

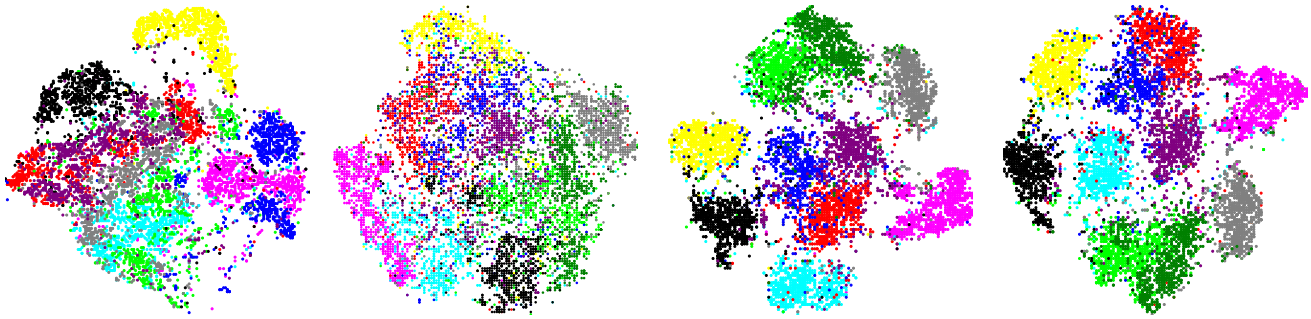
- **Orthogonal projection.** Projecting $\boldsymbol{v}$ to the tangent space of $\mathrm{St}(p, n)$ as in Eq. (5) involves multiplications between matrices of size **1-** $n\times p$ and $p\times p$ and **2-** $p\times n$ and $n \times p$. This adds up to $2np^2$ flops.
- **Retraction.** The retraction involves computing and adjusting the QR decomposition of an $n \times p$ matrix. The complexity of the QR decomposition using the Householder algorithm is $2p^2(n - p/3)$. Adjustments change the sign of elements of a column if the corresponding diagonal element of $\boldsymbol{R}$ is negative which does not incur much. Hence, the total complexity of the retraction is $O\big(2p^2(n - p/3)\big)$.

All the above steps are linear in n, again making the extra flops compared to the convolutional layers affordable. We also note that all the above operations can be done in a GPU. Table 1 provides a summary of the qConv and Stiefel geometry.

6. Stochastic Optimization

To train a SiNN, a set of triplets in the form of $\big\{\big(\boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i, y_i\big) : \boldsymbol{x}_i, \tilde{\boldsymbol{x}}_i \in \mathcal{X}, y_i = \{-1, 1\}\big\}$ are required. Here, $y_i = 1$ if $\boldsymbol{x}_i$ and $\tilde{\boldsymbol{x}}_i$ are semantically similar (*e.g.*, belonging to the same class) and $y_i = -1$ otherwise. We note that SiNNs require a weaker form of supervision in comparison to standard classification problems and a high level semantic supervision is sufficient for training. With the factorization $\boldsymbol{M} = \boldsymbol{L}\boldsymbol{L}^\top, \boldsymbol{L} \in \mathbb{R}^{n\times p}_*$, training an SiNN reads as

$$\min_{\Theta, \boldsymbol{L}} \sum_i \ell\Big(y_i, \boldsymbol{L}^\top f_\theta(\boldsymbol{x}_i), \boldsymbol{L}^\top f_\theta(\tilde{\boldsymbol{x}}_i)\Big) \ . \tag{7}$$



(a) Auto-Encoder (b) Conv (c) qConv (d) Stiefel

Figure 1. t-SNE visualization on the MNIST dataset for various configurations; after the models had converged w.r.t their respective training loss. Please see text for more details.

Table 1. Matrix representation, form of Riemannian gradient and retraction for the qConv and Stiefel layers.

	$\boldsymbol{M}=\boldsymbol{L}\boldsymbol{L}^\top$	$\boldsymbol{M}=\boldsymbol{U}\boldsymbol{S}^2\boldsymbol{U}^T$
Matrix representation	$\boldsymbol{L}$	$(\boldsymbol{U},\boldsymbol{S})$
Projecting onto the tangent space	$\boldsymbol{\xi}_{\boldsymbol{L}}+\boldsymbol{L}\Lambda$ with Λ obtained from Eqn. (2).	$\Big(\boldsymbol{\xi}_{\boldsymbol{U}}-\boldsymbol{U}\mathrm{sym}\Big(\boldsymbol{U}^T\boldsymbol{\xi}_{\boldsymbol{U}}\Big),\boldsymbol{\xi}_{\boldsymbol{S}}\Big)$
Retraction	$\Upsilon_{\boldsymbol{L}}(\bar{\xi})=\boldsymbol{L}+\bar{\xi}^{\uparrow}$	$\Upsilon_{\boldsymbol{U}}(\bar{\xi})=\mathrm{qf}(\boldsymbol{U}+\bar{\xi})$

Here, $\ell:\mathbb{R}\times\mathbb{R}^p\times\mathbb{R}^p\to\mathbb{R}$ is a loss function which incurs a penalty if the embedded points $\boldsymbol{L}^\top f_\theta(\boldsymbol{x}_i)$ and $\boldsymbol{L}^\top f_\theta(\tilde{\boldsymbol{x}}_i)$ are dissimilar for $y_i=1$ and vice-versa.

With this modeling, one can minimize (7) using the BP algorithm equipped with any gradient descent-based optimizers. As we see in the previous section, accounting for the invariance to the orthogonal group entails minimizing a constrained form of (7). In particular, we need to minimize either

$$\min_{\Theta,\boldsymbol{L}\in\mathcal{M}}\sum_i \ell\Big(y_i,\boldsymbol{L}^\top f_\theta(\boldsymbol{x}_i),\boldsymbol{L}^\top f_\theta(\tilde{\boldsymbol{x}}_i)\Big)\,, \tag{8}$$

or

$$\min_{\Theta,\boldsymbol{S},\boldsymbol{U}\in\mathrm{St}(p,n)}\sum_i \ell\Big(y_i,\boldsymbol{S}\boldsymbol{U}^\top f_\theta(\boldsymbol{x}_i),\boldsymbol{S}\boldsymbol{U}^\top f_\theta(\tilde{\boldsymbol{x}}_i)\Big)\,. \tag{9}$$

Minimizing (8) or (9) with respect to Θ (and $\boldsymbol{S}$ for the latter) is straightforward and the conventional BP algorithm can be applied directly. However, the same cannot be said about $\boldsymbol{L}$ and $\boldsymbol{U}$ as we need to preserve the foreseen structures during the updates of the BP algorithm. To achieve our goal, we make use of the Riemannian Stochastic Gradient Descent (rSGD) [3] and equip it with a momentum term. This innocent looking modification in practice seems to be very beneficial. To preserve the Riemannian structure, we propose the following updating scheme which we call rSGD with momentum or rSGD-M for short;

$$\boldsymbol{m}^{(t+1)}=\nu\,\Pi_{\boldsymbol{X}^{(t)}}\Big(\boldsymbol{m}^{(t)}\Big)+\eta\,\Pi_{\boldsymbol{X}^{(t)}}\Big(\nabla_{\boldsymbol{X}^{(t)}}J\Big), \tag{10}$$

$$\boldsymbol{X}^{(t+1)}=\Upsilon_{\boldsymbol{X}^{(t)}}\Big(-\boldsymbol{m}^{(t+1)}\Big). \tag{11}$$

Here, $\nu,\eta\in(0,1]$ are the momentum coefficient and the learning rate, $\nabla_{\boldsymbol{x}}$ is the gradient operator evaluated at $\boldsymbol{x}$, J is the function to be minimized and $\Pi_{\cdot}(\cdot)$ and $\Upsilon_{\cdot}(\cdot)$ are the projection onto the tangent space and retraction operators, respectively.

Before concluding this section, we use the MNIST [29] dataset to evaluate the benefits of the introduced geometrical constraints. In doing so, we first vectorize each image in the MNIST dataset into a 784 dimensional vector. For each configuration we train a single layer network that projects 784 dimensional input vector onto a 20 dimensional embedding space. Our goal here is to study and contrast the normal practice in training SiNN against what can be attained by the developments done in this work. Thus, we train the networks using the triplet embedding loss defined below:

$$\mathcal{L}_{tri}=\frac{1}{|P|}\sum_{i=1}^{|P|}\Big[\big\|\boldsymbol{x}_i^a-\boldsymbol{x}_i^p\big\|^2-\big\|\boldsymbol{x}_i^a-\boldsymbol{x}_i^n\big\|^2+\tau\Big]_+\,. \tag{12}$$

Here $[y]_+=\max(0,y)$ is the hinge loss and $\tau>0$ is a user-specified margin; and $(\boldsymbol{x}_i^a,\boldsymbol{x}_i^p,\boldsymbol{x}_i^n)$ represent a triplet. Semi-hard triplet mining strategy [41] was used to *mine* $v=5$ triplets for every $\boldsymbol{x}_i^a$ to generate $|P|$ triplets within a mini-batch of size N. N and τ were set to 100 and 1.0 for the three geometrical configurations *i.e.* Conv, qConv and Stiefel. Moreover as a baseline algorithm, we have trained a single 20 dimensional Auto-Encoder (AE) using the standard mean-squared loss. We report the final classification test error and average training time per single epoch after training each of the model for 100 epochs.

Table 2. Classification error using 1-Nearest Neighbour and average single-epoch training time on the MNIST dataset for AE, Conv, qConv and Stiefel.

Configuration	AE	Conv	qConv	Stiefel
error (%)	11.13	14.84	9.93	9.75
time (sec)	10.5	12.1	12.3	12.4

From Table 2, it is observed that both the qConv and the Stiefel layers outperform the traditional Conv and AE in terms of classification accuracy, with the Stiefel layer being slightly better than the qConv layer. This clearly shows the importance of enforcing such geometrical configurations in learning an embedding metric. Fig. 1 shows the t-SNE [36] plots of the various configurations after their respective models had converged during training. This indeed exhibits that by incorporating either of the proposed geometry, one can obtain compact embedding clusters over Conv and AE. This indeed demonstrates the enhanced discriminative ability of the qConv and the Stiefel layers over that of Conv and AE. Interestingly, the incurred complexity is also marginal as shown by the time required to perform one epoch of training in Table 2.

7. Empirical Evaluations

As the first experiment, we contrast the proposed qConv and Stiefel layers against the normal practice (*i.e.*, convolutional layers) in training Siamese networks using a somehow shallow network. In particular, we trained a shallow 5-layer on the *STL10* [6] dataset and studied the accuracy and running times of the vanilla convolutional layer against the qConv and Stiefel layers. The structure of the network reads as conv1(5×5) $\rightarrow$ max-pool $\rightarrow$ relu $\rightarrow$ conv2(5×5) $\rightarrow$ relu $\rightarrow$ avg-pool $\rightarrow$ conv3(5×5) $\rightarrow$ relu $\rightarrow$ conv4(4×4) $\rightarrow$ relu $\rightarrow$ fc5 $\rightarrow$ softmax-loss. We removed the softmax layer and replaced the *fc5* layer with qConv or the Stiefel layer in our experiments. We trained the entire network from the very same initialization point for all the studied geometries using Eqn. (12). For this experiment, we report the accuracy of a nearest neighborhood classifier after the embedding layer. The classification accuracies of the vanilla Siamese network, Siamese network equipped with qConv and Siamese network with the Stiefel layer are 42.7%, 48.5% and 48.6%, respectively, clearly demonstrating a huge improvements when the proper geometry is used for training. Focusing on running times, one epoch of training for convolutional, qConv and Stiefel layers takes 143, 150 and 152 seconds, respectively (The run times for all layers were measured using a GeForce GTX TITAN-X GPU.)

Fine-Grained Image Classification

In this part, we assess and contrast our proposed methods against several state-of-the-art methods on two fine-grained image datasets namely *Caltech-UCSD Birds* (CUB-200-2011) [55], *Stanford Cars* dataset (CARS196) [23] and *Stanford Online Products* (SOP) [40]. A brief overview of the dataset is provided in the supplementary material. In our evaluations, we have used the *Normalized Mutual Information* (**NMI**) and *Recall@K* (**R@K**) metrics. The former is an information theoretic measure, widely used to evaluate the performance of a clustering techniques while the latter indicates the fraction of queries for which there exists an image of the same class with the first K positions of the retrieved list. We compare our developments against the following baseline and state-of-the-arts deep learning methods; which can be broadly divided into four different categories: **(1) Structure** based methods; which constraints the manifold of the learnt embedding space. (a) **Trip-SH** [41] encourages the closest (semi-hard) *negative* $(\boldsymbol{x}_-)$ to be further away from the *positive* $(\boldsymbol{x}_+)$ for a given anchor $(\boldsymbol{x}_a)$; (b) **NMI-based** [45] that consists of a structured prediction loss to constrain the global structure of the embedding space in order to alleviate isolated clusters; (c) **DSC** [28] which uses spectral clustering concepts to construct the embedding space which encourages separate but compact clusters and (d) **Angular** [58] loss which constrains the angular relationship at the negative point of a triplet triangle. **(2) Sampling** based approaches; which aim to propose new and efficient sampling based strategies for better mining of informative samples. (a) **Lifted-Struct** [40] constrains a batch of triplets by subsequently adding importance-sampled difficult negatives to a training mini-batch; (b) **Npairs** [43] uses a novel loss that needs twice the available $2B$ negative pairs with B denoting the number of negative examples to form the triplets; and (c) **DWS** [61] makes use of an efficient Distance Weighted Sampling strategy to steadily mine informative examples, thereby reducing noisy gradients during back-propagation. **(3) Statistical** based approaches such as **Histogram** [53] and **Distribution** [25] loss that aim to reduce the overlap between the distributions of similarities (dissimilarities) between positive and negative pairs. **(4) Generative** modeling based approaches such as **DAML** [8] and **DVML** [33]to explicitly model the intra-class variance and disentangle the intra-class invariance.

Implementation Details

We follow the protocol considered in [28] and assess the proposed geometries using the Inception-V1 model [48]. All the images have been resized to 256×256 and are cropped at 224×224. The training images are augmented with random cropping and are randomly flipped horizontally, where as the test images are cropped from the center. Moreover, we have used only a single crop per image for both training and testing similar to [40]. To fine-tune the Inception model, we first normalized its outputs to have unit norm followed by adding an embedding layer per suggestions in [28] (the

Table 3. NMI and Recall@K evaluation on CUB-200-2011 [55].

Method	NMI	R@1	R@2	R@4	R@8
Trip-SH [41]	55.4	42.6	54.9	66.4	77.2
Lifted-Struct [40]	56.5	43.6	56.6	68.6	79.6
Npairs [43]	57.2	45.4	58.4	69.5	79.5
NMI-based [45]	59.2	48.2	61.4	71.8	81.9
DSC (end-to-end/SC) [28]	58.1	49.8	62.6	73.6	82.8
Angular [58]	61.1	53.6	65.0	75.3	83.7
Distribution * [25]	59.4	47.9	60.6	71.9	81.6
Histogram [53]	-	52.8	64.4	74.7	83.9
DWS* [61]	61.3	52.4	**65.1**	75.1	83.6
DVML* [33]	61.4	**52.7**	64.9	75.5	**84.3**
DAML* [8]	60.8	51.5	64.7	75.2	83.5
Ours (Quotient)	**62.5**	52.2	64.0	**75.7**	83.3
Ours (Stiefel)	62.3	52.3	64.5	75.3	84.0

Table 4. NMI and Recall@K evaluation on CARS196 [23].

Method	NMI	R@1	R@2	R@4	R@8
Trip-SH [41]	53.4	51.5	63.8	73.5	82.4
Lifted-Struct [40]	56.9	53.0	65.7	76.0	84.3
Npairs [43]	57.8	53.9	66.8	77.8	86.4
NMI-based [45]	59.0	58.1	70.6	80.3	87.8
DSC (end-to-end/SC) [28]	58.0	59.4	71.3	80.6	88.3
Angular [58]	62.4	71.3	80.7	87.0	91.8
Distribution* [25]	61.6	64.4	75.4	83.6	89.5
Histogram* [53]	-	66.2	77.2	85.0	90.8
DWS* [61]	62.1	70.3	78.2	86.9	90.7
DVML* [33]	62.7	71.6	79.7	87.8	91.1
DAML* [8]	63.1	72.5	82.1	88.5	**92.9**
Ours (Quotient)	63.5	71.9	81.5	87.9	92.5
Ours (Stiefel)	**64.2**	**73.2**	**82.2**	**88.6**	92.2

Table 5. NMI and Recall@K evaluation on SOP [40].

Method	NMI	R@1	R@10	R@100
Trip-SH [41]	89.5	66.7	82.4	73.5
Lifted-Struct [40]	88.7	62.5	80.8	91.9
Npairs [43]	89.4	66.4	83.2	93.0
NMI-based [45]	89.5	67.0	**83.7**	93.2
DSC (end-to-end/SC) [28]	89.4	67.6	**83.7**	**93.3**
Angular [58]	87.8	67.9	83.2	92.2
Distribution* [25]	88.9	63.5	81.3	91.3
Histogram [53]	-	63.9	81.7	92.2
DWS* [61]	89.4	67.1	82.7	92.3
DVML* [33]	89.2	67.2	82.3	92.1
DAML* [8]	89.2	67.1	82.8	92.5
Ours (Quotient)	89.3	68.5	82.9	92.6
Ours (Stiefel)	**89.9**	**69.2**	83.1	92.7

embedding layer is initialized using a classification loss).We used a batch size of 64 samples and ensured that there were at least 4 samples from each class in a batch. We used RMSProp [14]optimizer to update the parameters of all the layers using backpropagation. The learning rate for all the datasets was fixed at 10^{-4} and decreased by a factor of 0.1 after every 25 epochs. The entire network was trained using Eqn. (12); where τ is set to 0.5 for all the datasets. Several recent studies including [40, 28] suggest that the Inception network equipped with a metric loss is more or less robust to the embedding dimensionality. Therefore we have followed the experimental protocol of [28] and have fixed the dimensionality of embedding layer to the number of training categories k for both the CUB-200-2011 and CARS916 datasets; and 512 for SOP dataset. In the evaluation phase, we apply the standard *KMeans* algorithm with squared Euclidean distance on the output representations and calculate the **NMI** and **R@K** metrics.

Note: For a fair comparison, we have reported the results of the aforementioned algorithms with triplet embedding loss on the Inception network; even though [8, 61, 33] use a combination of different loss functions to obtain improved embedding space. Moreover, the algorithms marked as "$*$" were self-implemented as either the algorithms were not evaluated on those datasets [25], or a different backbone network other than Inception was used [61], or the embedding dimensionality was not same [8, 33].

Tables 3, 4 and 5 show the results of the quantitative comparisons between our methods and other deep metric learning techniques for the CUB-200-2011, CARS196 and SOP datasets, respectively. Studying this table reveals that the proposed geometries not only outperform the vanilla SiNN [41] significantly but also comfortably surpass the performance of the various state-of-the-art methods in all the three datasets. In particular, recent methods such as DWS [61], DAML [8] and DVML [33] make use of additional complex networks, sampling strategies or intricate loss functions to obtain a discriminative embedding space for fine-grained classification. However, we successfully demonstrate that a deep network equipped with the proposed geometrical configurations and trained using the standard semi-hard triplet embedding outperforms the aforementioned algorithms. This in-turn validates the importance of the proposed configurations in training SiNN.

8. Conclusions and Future Work

In this paper, we focus on Siamese networks that realize a non-linear embedding characterized by a positive semi-definite matrix and showed how Riemannian geometry can be used to take into account the somehow hidden invariance property of the network. Specifically we have developed a novel geometrical manifold, namely qConv; and used it along with the standard Stiefel manifold to exploit the invariances in the siamese networks. Stiefel is slightly more computationally expensive than qConv, although their empirical results are quite similar. A future work can be to further develop the associated geometry to handle non-symmetric and indefinite cases. We note that this can be achieved for example by removing the constraint that both legs of the Siamese network should go through the same convolutional layer. Another venue that goes beyond the current work is to study the convergence of the proposed Riemannian SGD algorithm with momentum.

References

[1] P-A Absil, Robert Mahony, and Rodolphe Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2009. 1, 5

[2] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-Convolutional Siamese Networks for Object Tracking. In *ECCV*, pages 850–865. Springer, 2016. 1

[3] Silvere Bonnabel. Stochastic Gradient Descent on Riemannian Manifolds. *IEEE Transactions on Automatic Control*, 58(9):2217–2229, 2013. 6

[4] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature Verification using a "Siamese" Time Delay Neural Network. In *NIPS*, pages 737–744, 1994. 1, 2, 3

[5] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a Similarity Metric Discriminatively, with Application to Face Verification. In *CVPR*, volume 1, pages 539–546, 2005. 1, 3

[6] Adam Coates, Andrew Y Ng, and Honglak Lee. An Analysis of Single-Layer Networks in Unsupervised Feature Learning. In *Proc. Int. Conf. Artificial Intelligence & Statistics*, 2011. 7

[7] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive Angular Margin Loss for Deep Face Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2019. 3

[8] Yueqi Duan, Wenzhao Zheng, Xudong Lin, Jiwen Lu, and Jie Zhou. Deep Adversarial Metric Learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2780–2789, 2018. 1, 7, 8

[9] Alan Edelman, Tomás A Arias, and Steven T Smith. The Geometry of Algorithms with Orthogonality Constraints. *SIAM journal on Matrix Analysis and Applications*, 20(2):303–353, 1998. 5

[10] Pengfei Fang, Jieming Zhou, Soumava Kumar Roy, Lars Petersson, and Mehrtash Harandi. Bilinear Attention Networks for Person Retrieval. In *Proceedings of the IEEE International Conference on Computer Vision*, 2019. 1

[11] Gene H Golub and Charles F Van Loan. *Matrix Computations*. The Johns Hopkins University Press, 4 edition, 2013. 5

[12] Albert Gordo, Jon Almazan, Jerome Revaud, and Diane Larlus. End-to-End Learning of Deep Visual Representations for Image Retrieval. *Int. Journal of Computer Vision*, 124(2):237–254, 2017. 1

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, pages 770–778, 2016. 2

[14] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural Networks for Machine Learning Lecture 6a Overview of Mini-Batch Gradient Descent. 8

[15] Elad Hoffer and Nir Ailon. Deep Metric Learning using Triplet Network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92. Springer, 2015. 1

[16] Shota Horiguchi, Daiki Ikami, and Kiyoharu Aizawa. Significance of Softmax-based Features in Comparison to Distance Metric Learning-based Features. *IEEE transactions on pattern analysis and machine intelligence*, 2019. 3

[17] Junlin Hu, Jiwen Lu, and Yap-Peng Tan. Discriminative Deep Metric Learning for Face Verification in the Wild. In *CVPR*, pages 1875–1882, 2014. 1

[18] Chen Huang, Chen Change Loy, and Xiaoou Tang. Local Similarity-Aware Deep Feature Embedding. In *NIPS*, pages 1262–1270, 2016. 3

[19] Zhiwu Huang, Jiqing Wu, and Luc Van Gool. Building Deep Networks on Grassmann Manifolds. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 1

[20] M. Journée, F. Bach, P.-A. Absil, and R. Sepulchre. Low-Rank Optimization on the Cone of Positive Semidefinite Matrices. *SIAM Journal on Optimization*, 20(5):2327–2351, 2010. 2, 3

[21] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese Neural Networks for One-Shot Image Recognition. In *ICML Deep Learning Workshop*, 2015. 1, 3

[22] Piotr Koniusz, Yusuf Tas, Hongguang Zhang, Mehrtash Harandi, Fatih Porikli, and Rui Zhang. Museum Exhibit Identification Challenge for the Supervised Domain Adaptation and Beyond. In *The European Conference on Computer Vision (ECCV)*, 2018. 1

[23] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d Object Representations for Fine-Grained Categorization. In *Proc. of the IEEE Int. Conf. on Computer Vision Workshops*, 2013. 7, 8

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet Classification with Deep Convolutional Neural Networks. In *NIPS*, 2012. 2

[25] BG Kumar, Gustavo Carneiro, Ian Reid, et al. Learning Local Image Descriptors with Deep Siamese and Triplet Convolutional Networks by Minimising Global Loss Functions. In *CVPR*, pages 5385–5394, 2016. 1, 7, 8

[26] Vijay BG Kumar, Ben Harwood, Gustavo Carneiro, Ian Reid, and Tom Drummond. Smart Mining for Deep Metric Learning. *arXiv preprint arXiv:1704.01285*, 2017. 1, 3

[27] Marc T Law, Nicolas Thome, and Matthieu Cord. Learning a Distance Metric from Relative Comparisons Between Quadruplets of Images. *Int. Journal of Computer Vision*, 121(1):65–94, 2017. 3

[28] Marc T Law, Raquel Urtasun, and Richard S Zemel. Deep Spectral Clustering Learning. In *ICML*, pages 1985–1994, 2017. 1, 2, 3, 7, 8

[29] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 6

[30] Yann A LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural Networks: Tricks of the Trade*, pages 9–48. Springer, 2012. 2

[31] John M Lee. *Introduction to Smooth Manifolds – Second Edition*. Springer, 2003. 4

[32] Tsung-Yi Lin, Yin Cui, Serge Belongie, and James Hays. Learning Deep Representations for Ground-to-Aerial Geolocalization. In *CVPR*, pages 5007–5015, 2015. 1

[33] Xudong Lin, Yueqi Duan, Qiyuan Dong, Jiwen Lu, and Jie Zhou. Deep Variational Metric Learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 689–704, 2018. 1, 7, 8

[34] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. Sphereface: Deep Hypersphere Embedding for Face Recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 212–220, 2017. 3

[35] Wenjie Luo, Alexander G Schwing, and Raquel Urtasun. Efficient Deep Learning for Stereo Matching. In *CVPR*, pages 5695–5703, 2016. 1

[36] Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008. 7

[37] Jonathan Masci, Michael M Bronstein, Alexander M Bronstein, and Jürgen Schmidhuber. Multimodal Similarity-Preserving Hashing. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 36(4):824–830, 2014. 1

[38] Gilles Meyer, Silvère Bonnabel, and Rodolphe Sepulchre. Regression on Fixed-Rank Positive Semidefinite Matrices: a Riemannian Approach. *Journal of Machine Learning Research*, 12(Feb):593–625, 2011. 2

[39] B Mishra, G Meyer, S Bonnabel, and R Sepulchre. Fixed-Rank Matrix Factorizations and Riemannian Low-Rank Optimization. *Computational Statistics*, 29(3-4):591–621, 2014. 2, 3

[40] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep Metric Learning via Lifted Structured Feature Embedding. In *CVPR*, pages 4004–4012, 2016. 1, 2, 3, 7, 8

[41] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A Unified Embedding for Face Recognition and Clustering. In *CVPR*, pages 815–823, 2015. 1, 3, 6, 7, 8

[42] Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, Pascal Fua, and Francesc Moreno-Noguer. Discriminative Learning of Deep Convolutional Feature Point Descriptors. In *ICCV*, pages 118–126, 2015. 1

[43] Kihyuk Sohn. Improved Deep Metric Learning with Multi-class N-pair Loss Objective. In *NIPS*, pages 1857–1865, 2016. 1, 2, 3, 7, 8

[44] Jeany Son, Mooyeol Baek, Minsu Cho, and Bohyung Han. Multi-Object Tracking with Quadruplet Convolutional Neural Networks. In *CVPR*, pages 5620–5629, 2017. 1

[45] Hyun Oh Song, Stefanie Jegelka, Vivek Rathod, and Kevin Murphy. Deep Metric Learning via Facility Location. In *CVPR*, 2017. 1, 2, 3, 7, 8

[46] Yi Sun, Yuheng Chen, Xiaogang Wang, and Xiaoou Tang. Deep Learning Face Representation by Joint Identification-Verification. In *NIPS*, pages 1988–1996, 2014. 3

[47] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Deep Learning Face Representation from Predicting 10,000 Classes. In *CVPR*, pages 1891–1898, 2014. 1

[48] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. In *CVPR*, pages 1–9, 2015. 2, 7

[49] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the Gap to Human-Level Performance in Face Verification. In *CVPR*, pages 1701–1708, 2014. 1, 3

[50] Ran Tao, Efstratios Gavves, and Arnold WM Smeulders. Siamese Instance Search for Tracking. In *CVPR*, pages 1420–1429, 2016. 1

[51] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient Object Localization using Convolutional Networks. In *CVPR*, pages 648–656, 2015. 1

[52] Eleni Triantafillou, Richard Zemel, and Raquel Urtasun. Few-Shot Learning Through an Information Retrieval Lens. In *NIPS*, 2017. 1, 3

[53] Evgeniya Ustinova and Victor Lempitsky. Learning Deep Embeddings with Histogram Loss. In *NIPS*, pages 4170–4178, 2016. 1, 3, 7, 8

[54] Rahul Rama Varior, Mrinal Haloi, and Gang Wang. Gated Siamese Convolutional Neural Network Architecture for Human Re-Identification. In *ECCV*, pages 791–808, 2016. 1

[55] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical report, California Institute of Technology, 2011. 7, 8

[56] Feng Wang, Xiang Xiang, Jian Cheng, and Alan Loddon Yuille. Normface: L_2 Hypersphere Embedding for Face Verification. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 1041–1049. ACM, 2017. 3

[57] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large Margin Cosine Loss for Deep Face Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5265–5274, 2018. 3

[58] Jian Wang, Feng Zhou, Shilei Wen, Xiao Liu, and Yuanqing Lin. Deep Metric Learning with Angular Loss. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2593–2601, 2017. 7, 8

[59] Kilian Q Weinberger and Lawrence K Saul. Distance Metric Learning for Large Margin Nearest Neighbor Classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009. 3

[60] Zaiwen Wen and Wotao Yin. A Feasible Method for Optimization with Orthogonality Constraints. *Mathematical Programming*, 142(1):397–434, 2013. 5

[61] Chao-Yuan Wu, R Manmatha, Alexander J Smola, and Philipp Krähenbühl. Sampling Matters in Deep Embedding Learning. In *ICCV*, 2017. 1, 3, 7, 8

[62] Yongqin Xian, Christoph H Lampert, Bernt Schiele, and Zeynep Akata. Zero-Shot Learning-a Comprehensive Evaluation of the Good, the Bad and the Ugly. *IEEE transactions on pattern analysis and machine intelligence*, 2018. 3

[63] Tsun-Yi Yang, Jo-Han Hsu, Yen-Yu Lin, and Yung-Yu Chuang. DeepCD: Learning Deep Complementary Descriptors for Patch Representations. In *CVPR*, pages 3314–3322, 2017. 1

[64] Sergey Zagoruyko and Nikos Komodakis. Learning to Compare Image Patches via Convolutional Neural Networks. In *CVPR*, pages 4353–4361, 2015. 1

[65] Hongguang Zhang, Jing Zhang, and Piotr Koniusz. Few-Shot Learning via Saliency-Guided Hallucination of Samples. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1