RIEMANNIAN RESIDUAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent methods in geometric deep learning have introduced various neural networks to operate on Riemannian manifolds. These methods are often inspired by and directly generalize standard Euclidean neural networks. In practice, extending these is difficult and has only been done for a select few manifolds. In this work, we examine the residual neural network (ResNet) and show how to extend this construction to general Riemannian manifolds. Originally introduced to help solve the vanishing gradient problem, ResNets have become ubiquitous in machine learning due to their beneficial learning properties, excellent empirical results, and easy-to-incorporate nature when building varied neural networks. We find that our Riemannian ResNets mirror these desirable properties and generalize well to non-Euclidean manifolds (regardless of topology).

1 INTRODUCTION

In machine learning, it is common to represent data as vectors in Euclidean space (i.e. $\mathbb{R}^n$). The primary reason for such a choice is convenience, as this space has a classical vectorial structure, a closed-form distance formula, and a simple inner-product computation. Moreover, the myriad Euclidean neural network constructions allow one to learn efficiently.

Despite the ubiquity and success of Euclidean embeddings, recent research (Nickel & Kiela, 2017) has brought attention to the fact that several kinds of complex data necessitate manifold considerations. Such data are various and range from gauge group samples found in lattice quantum field theory (Boyda et al., 2020) to motion samples on tori found in the context of robotics (Rezende et al., 2020). However, generalizing Euclidean neural network tools to complex manifold structures such as these can be quite difficult in practice. Most prior work design network architectures on a specific manifold (Ganea et al., 2018; Cohen et al., 2018).

In our paper, we address this issue by extending Residual Neural Networks (He et al., 2016) to Riemannian manifolds. We construct our network by parameterizing vector fields, a strategy that works generally for smooth manifolds. We compare our work to existing manifold-specific network designs on hyperbolic space and on the manifold of symmetric positive definite (SPD) matrices, showing that our network better adheres to the geometry of the underlying data.

2 RELATED WORK

Our work is inspired heavily by existing neural ordinary differential equation (ODE) (Chen et al., 2018) literature as well a series of papers that have attempted generalization of neural networks to specific manifolds such as hyperbolic space and the manifold of SPD matrices (Ganea et al., 2018; Huang & Gool, 2017).

2.1 RESIDUAL NETWORKS AND NEURAL ODES

Residual networks (ResNets) were originally developed to enable training of larger networks, previously prone to vanishing and exploding gradients (He et al., 2016). Later on, Chen et al. (2018) discovered that by adding a learned residual, ResNets are similar to Euler's method. More specifically, the ResNet represented by $\mathbf{h}_{t+1} = \mathbf{h}_t + f(\mathbf{h}, \theta_t)$ for $\mathbf{h}_t \in \mathbb{R}^D$ mimics the dynamics of the ODE defined by $\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t, \theta)$. Similar to our work, Lou et al. (2020); Katsman et al. (2021)

generalize neural ODEs to Riemannian manifolds. However, instead of using a manifold's vector fields to solve a neural ODE, we learn an objective by parameterizing the vector fields directly.

2.2 Riemannian Neural Networks

Neural networks have been extended to non-Euclidean spaces in many cases. For example, Ganea et al. (2018) extended basic neural network operations to conform with the geometry of hyperbolic space through gyrovector constructions (Ungar, 2009). Neural network constructs have been extended to the SPD manifold (Huang & Gool, 2017; Brooks et al., 2019; López et al., 2021) as well. Unlike any of the manifold-specific work described, our residual network construction can be applied generally to any smooth manifold.

3 Background

In this section, we cover the necessary background for our paper; in particular, we introduce the reader to the necessary constructs from Riemannian geometry. For a detailed introduction to Riemannian geometry, we refer the interested reader to textbooks such as Lee (2013); Kobyzev et al. (2020).

3.1 Riemannian Geometry

A topological manifold $(\mathcal{M}, g)$ of dimension n is a locally Euclidean space, meaning there exist homeomorphic[1] charts whose domains both cover the manifold and map from the manifold to $\mathbb{R}^n$ (i.e. the manifold "looks like" $\mathbb{R}^n$ locally). A smooth manifold is a topological manifold for which the charts are not simply homeomorphic, but diffeomorphic, meaning they map to $\mathbb{R}^n$ differentiably and have differentiable inverses. Further still, a Riemannian manifold[2] $(\mathcal{M}, g)$ is an n-dimensional smooth manifold with a smooth collection of inner products $(g_x)_{x\in\mathcal{M}}$ for every tangent space $T_x\mathcal{M}$. The Riemannian metric g induces a distance $d_g : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$ on the manifold.

Riemannian metrics also allow for a natural analogue of gradients on $\mathbb{R}^n$. For a function $f : \mathcal{M} \to \mathbb{R}$, we define the Riemannian gradient $\nabla_x f$ to be the vector on $T_x\mathcal{M}$ such that $g_x(\nabla_x f, v) = D_x f(v)$ for $v \in T_x\mathcal{M}$.

3.2 Geodesics and the Riemannian Exponential Map

Geodesics A geodesic is a smooth curve of minimal length between two points $p, q \in \mathcal{M}$, and can be seen as the generalization of a straight-line in Euclidean space. Although a choice of Riemannian metric g on $\mathcal{M}$ appears to only define geometry locally on $\mathcal{M}$, it induces global distances by integrating the length (of the "speed" vector in the tangent space) of a shortest path between two points: $d(p, q) = \inf_\gamma \int_0^1 \sqrt{g_{\gamma(t)}(\gamma'(t), \gamma'(t))}\, dt$ where $\gamma \in C^\infty([0, 1], \mathcal{M})$ is such that $\gamma(0) = p$ and $\gamma(1) = q$.

For $p \in \mathcal{M}$ and $v \in T_p\mathcal{M}$, there exists a unique geodesic γ_v where $\gamma(0) = p$, $\gamma'(0) = v$ and the domain of γ is as large as possible. We call γ_v the maximal geodesic (Lee, 1997).

Exponential Map The Riemannian exponential map is a way to map $T_p\mathcal{M}$ to a neighborhood around p using geodesics. This can be thought of as a local linearization, meaning that we can perform typical Euclidean operations in the tangent space before projecting to the manifold via the exponential map. For $p \in \mathcal{M}$ and $v \in T_p\mathcal{M}$, the exponential map at p is defined as $\exp_p(v) = \gamma_v(1)$.

3.3 Vector Fields

Let $T_p\mathcal{M}$ be the tangent space to a manifold $\mathcal{M}$ at a point p. Like in Euclidean space, a vector field assigns to each point $p \in \mathcal{M}$ a tangent vector $X_p \in T_p\mathcal{M}$.

Tangent Bundle Consider the disjoint union of the tangent spaces $T_p\mathcal{M}$, for all $p \in \mathcal{M}$:

$$T\mathcal{M} = \bigsqcup_{p\in\mathcal{M}} T_p\mathcal{M} = \bigsqcup_{p\in\mathcal{M}} \{(p, v) \mid v \in T_p\mathcal{M}\}.$$

[1] A homeomorphism is a continuous bijection with continuous inverse.

[2] Note that imposing Riemannian structure does not considerably limit the generality of our method, as any smooth manifold that is Hausdorff and second countable has a Riemannian metric (Lee, 1997).

For a smooth manifold $\mathcal{M}$ of dimension n, $T\mathcal{M}$ is a smooth manifold of dimension $2n$ and is called the tangent bundle of $\mathcal{M}$. Over an open subset U of $\mathcal{M}$, the tangent bundle is locally homeomorphic to the product manifold $U \times \mathbb{R}^n$ (Lee, 2013).

Smooth Vector Field A smooth vector field assigns a tangent vector $X_p \in T_p\mathcal{M}$ to each point $p \in \mathcal{M}$ such that X_p varies smoothly in p. Formally, let U be an open subset of $\mathcal{M}$. A section X of $T\mathcal{M}$ over U is a function $X : U \to T\mathcal{M}$ such that $X_p \in T_p\mathcal{M}$ for every $p \in U$. A vector field on U is any section X, and X is a smooth vector field provided it is also a smooth map.

3.4 Model Spaces in Riemannian Geometry

The three Riemannian model spaces are Euclidean space, $\mathbb{R}^n$, hyperbolic space $\mathbb{H}^n$, and spherical space $\mathbb{S}^n$, together with their canonical metrics. They are called as such because they encompass all of the spaces of constant sectional curvature, i.e. Euclidean space has curvature 0, hyperbolic space has constant negative curvature, and spherical space has constant positive curvature (Lee, 1997).

3.5 SPD Manifold

Let $SPD(n)$ be the manifold of $n \times n$ symmetric positive definite (SPD) matrices together with their canonical metric (Cruceru et al., 2021). We recall from Gallier & Quaintance (2020) that $SPD(n)$ has Riemannian exponential map equal to the matrix exponential. Note that $SPD(n)$ has non-constant negative sectional curvature, giving it a considerably less trivial geometry than that exhibited by the Riemannian model spaces (Bhatia, 2007).

4 Methodology

In this section, we provide the technical details behind our residual network construction on Riemannian manifolds. Our approach is inspired by the interplay between Neural ODEs and ResNets on Euclidean space (Chen et al., 2018; He et al., 2016).

4.1 Construction

We define a **Riemannian Residual Neural Network** (RResNet) on a manifold $\mathcal{M}$ to be a function $f_{nn} : \mathcal{M} \to \mathcal{M}$ defined by

$$f(x) := x^{(m)}, \quad x^{(0)} := x, \quad x^{(i)} := \exp_{x^{(i-1)}}(\ell_i(x^{(i-1)})) \; \forall i \in [m]$$

for $x \in \mathcal{M}$, where m is the number of layers and $\ell_i : \mathcal{M} \to T\mathcal{M}$ is a neural network-parameterized[3] vector field over $\mathcal{M}$. In practice, parameterizing a function from an abstract manifold $\mathcal{M}$ to its tangent bundle is very difficult. However, by the Whitney embedding theorem (Lee, 2013), we can embed $\mathcal{M} \hookrightarrow \mathbb{R}^D$ for some dimension $D \geq \dim \mathcal{M}$. As such, for a standard neural network $n_i : \mathbb{R}^D \to \mathbb{R}^D$ we can construct ℓ_i by

$$\ell_i(x) := \text{proj}_{T_x\mathcal{M}}(n_i(x))$$

where we note that $T_x\mathcal{M} \subset \mathbb{R}^D$ is a linear subspace (making the projection operator well defined). We note that this is the same construction used for defining the vector field flow in Lou et al. (2020).

We also extend our construction to work in settings where the underlying manifold changes from layer to layer. In particular, for a sequence of manifolds $\mathcal{M}^{(0)}, \mathcal{M}^{(1)}, \ldots, \mathcal{M}^{(m)}$ with (possibly learned) maps $h_i : \mathcal{M}^{(i-1)} \to \mathcal{M}^{(i)}$, our Riemannian ResNet $f_{nn} : \mathcal{M}^{(0)} \to \mathcal{M}^{(m)}$ is given by

$$f(x) := x^{(m)}, \quad x^{(0)} := x, \quad x^{(i)} := \exp_{h_i(x^{(i-1)})}(\ell_i(h_i(x^{(i-1)}))) \; \forall i \in [m]$$

with functions $\ell_i : \mathcal{M}^{(i)} \to T\mathcal{M}^{(i)}$ given as above. In practice, our $\mathcal{M}^{(i)}$ will be different dimensions of the same geometric space (e.g. $\mathbb{H}^n$ or $\mathbb{R}^n$ for varying n), and the maps h_i will either be the standard inclusions or standard neural networks.

Furthermore, as shown in Appendix C, our model is equivalent to the standard ResNet when the underlying manifold is Euclidean space.

[3]We abuse notation, and note we implicitly take $\ell_i \in \Gamma^\infty(\mathcal{M}, T\mathcal{M})$, meaning the ℓ_i maps from points to vectors in the associated tangent space.

4.2 Comparison with Other Constructions

When compared with the hyperbolic neural network (HNN) constructions in Ganea et al. (2018), our RResNets are more principled since they rely less on the geometry of Euclidean space. In particular, the main source of non-Euclidean structure in HNNs is the bias term, which is introduced by way of Möbius addition $\oplus$ (Ungar, 2009). By comparison, RResNets rely almost fully on the geometry of the underlying Riemannian manifold. In particular, our only usage of Euclidean space is to parameterize vector fields (which are inherently Euclidean), but their application to the manifold is through the non-Euclidean Riemannian exponential map. Furthermore, our method is far more general since it does not rely on the niceties of hyperbolic space (as does Ganea et al. (2018)).

5 Experiments

In this section, we perform a series of experiments to evaluate the effectiveness of RResNets on tasks arising on different manifolds. In particular, we explore hyperbolic space and the manifold of SPD matrices. Please see the appendix for full experimental details.

5.1 Hyperbolic Space

We first perform node classification and link prediction tasks on graph datasets with low Gromov δ-hyperbolicity (Chami et al., 2019), which means their underlying structure is highly hyperbolic. In particular, we test on the Airport (Chami et al., 2019) and PubMed (Sen et al., 2008) datasets. For the baseline, we use the results for Hyperbolic Neural Networks (HNN) (Ganea et al., 2018) reported by Chami et al. (2019). Table 1 summarizes the performance of RResNet relative to HNN.[4]

Dataset	Airport		PubMed	
Hyperbolicity	$\delta = 1$		$\delta = 3.5$	
Task	LP	NC	LP[5]	NC
HNN	93.0 ± 0.8	60.9 ± 0.4	–	69.6 ± 1.0
RResNet	$\mathbf{95.1 \pm 0.1}$	$\mathbf{72.4 \pm 2.4}$	–	$\mathbf{72.5 \pm 1.0}$

Table 1: The metrics reported are ROC AUC for link prediction (LP) and F1-score for node classification (NC). Ten trials were conducted; mean and standard deviation are reported. We compare the performance of HNN, as implemented by Chami et al. (2019), to RResNet. Note that we considerably outperform the baseline while using fewer parameters. The best result on each task is bolded.

Aside from Airport node classification for which we could not reproduce the baseline, RResNet consistently outperforms HNN. Moreover, we manage to do so with 20% – 50% fewer parameters. See the appendix for the architectural details. This improvement highlights how RResNet is more natural for learning in hyperbolic space; at the same time, it is a general Riemannian construction.

5.2 SPD Manifold

We compare our RResNet to existing SPD manifold-based models on the AFEW emotion classification dataset (Dhall et al., 2011). We experience faster convergence and improved training dynamics. Please see the appendix for details on these experiments.

6 Conclusion

We proposed a general construction of residual neural networks on Riemannian manifolds. Our introduced approach is a natural geodesically-oriented generalization that can be applied far more broadly than previous manifold-specific work. We also show that when RResNet is applied to hyperbolic space it outperforms baselines considerably while being more parameter efficient.

[4] We could not reproduce the baseline for NC on Airport; Chami et al. (2019) report this value as 80.5 ± 0.5.

[5] We had numerical issues with LP on PubMed that prevented us from obtaining stable results across trials.

REFERENCES

Cem Anil, James Lucas, and Roger B. Grosse. Sorting out lipschitz function approximation. In *ICML*, 2019.

Rajendra Bhatia. Positive definite matrices. 2007.

Denis Boyda, Gurtej Kanwar, Sébastien Racanière, Danilo Jimenez Rezende, Michael S Albergo, Kyle Cranmer, Daniel C Hackett, and Phiala E Shanahan. Sampling using $su(n)$ gauge equivariant flows. *arXiv preprint arXiv:2008.05456*, 2020.

Daniel A. Brooks, Olivier Schwander, Frédéric Barbaresco, Jean-Yves Schneider, and Matthieu Cord. Riemannian batch normalization for spd neural networks. In *NeurIPS*, 2019.

Mario Lezcano Casado and David Martínez-Rubio. Cheap orthogonal constraints in neural networks: A simple parametrization of the orthogonal and unitary group. *ArXiv*, abs/1901.08428, 2019.

Ines Chami, Zhitao Ying, Christopher Ré, and Jure Leskovec. Hyperbolic graph convolutional neural networks. In *Advances in neural information processing systems*, pp. 4868–4879, 2019.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31, pp. 6571–6583, 2018.

Taco S. Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical CNNs. In *International Conference on Learning Representations*, 2018.

Calin Cruceru, Gary B'ecigneul, and Octavian-Eugen Ganea. Computationally tractable riemannian manifolds for graph embeddings. In *AAAI*, 2021.

Abhinav Dhall, Roland Göcke, Simon Lucey, and Tom Gedeon. Static facial expression analysis in tough conditions: Data, evaluation protocol and benchmark. *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pp. 2106–2112, 2011.

Jean Gallier and Jocelyn Quaintance. *Differential Geometry and Lie Groups: A Computational Perspective*, volume 12. Springer, 2020.

Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. In *Advances in neural information processing systems*, pp. 5345–5355, 2018.

Ian J. Goodfellow, Yoshua Bengio, and Aaron C. Courville. Deep learning. *Nature*, 521:436–444, 2015.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Zhiwu Huang and Luc Van Gool. A riemannian network for spd matrix learning. In *AAAI*, 2017.

Isay Katsman, Aaron Lou, Derek Lim, Qingxuan Jiang, Ser-Nam Lim, and Christopher De Sa. Equivariant manifold flows. *ArXiv*, abs/2107.08596, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Ivan Kobyzev, Simon Prince, and Marcus Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

John M. Lee. Riemannian manifolds: An introduction to curvature. 1997.

John M Lee. *Introduction to Smooth Manifolds*. Graduate Texts in Mathematics. Springer New York, 2013.

F. Javier López, Béatrice Pozzetti, Steve J. Trettel, Michael Strube, and Anna Wienhard. Vector-valued distance and gyrocalculus on the space of symmetric positive definite matrices. *ArXiv*, abs/2110.13475, 2021.

Aaron Lou, Derek Lim, Isay Katsman, Leo Huang, Qingxuan Jiang, Ser-Nam Lim, and Christopher De Sa. Neural manifold ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 33, pp. 17548–17558, 2020.

Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. In *Advances in neural information processing systems*, pp. 6338–6347, 2017.

Danilo Jimenez Rezende, George Papamakarios, Sebastien Racaniere, Michael Albergo, Gurtej Kanwar, Phiala Shanahan, and Kyle Cranmer. Normalizing flows on tori and spheres. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 8083–8092, 2020.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. *AI Mag.*, 29:93–106, 2008.

Abraham Albert Ungar. A gyrovector space approach to hyperbolic geometry. In *A Gyrovector Space Approach to Hyperbolic Geometry*, 2009.

Appendix

A VECTOR FIELD DESIGN

Recall from the main paper that we can design a neural network-parameterized vector field $\ell_i : \mathcal{M} \to T\mathcal{M}$ for an embedded manifold $\mathcal{M}$ of dimension D, simply by defining a standard neural network $n_i : \mathbb{R}^D \to \mathbb{R}^D$ and then setting:

$$\ell_i(x) := \text{proj}_{T_x\mathcal{M}}(n_i(x)).$$

Though this vector field design is frequently trivial (assuming the manifold has a natural embedding in $\mathbb{R}^n$), it may be highly inefficient if an easy-to-implement but suboptimal embedding is used. This is especially the case if manifold structure is underexploited in the construction of such an embedding (see Section A.2). In this section, we give a natural vector field design for hyperbolic space, and explore a variety of possible vector field designs for the SPD manifold. In the general setting, note that obtaining a parsimonious (with respect to either representational dimension or parameter count) vector field design that is sufficiently expressive is nontrivial.

A.1 VECTOR FIELD DESIGN FOR HYPERBOLIC SPACE

For the hyperbolic vector field design, we apply the general design construction referenced in Section A above. Note that $\mathbb{H}^n$ is an n-dimensional manifold with a trivial $\mathbb{R}^{n+1}$ embedding given by any coordinate representation. Thus we need only parameterize a neural network $n_i : \mathbb{R}^{n+1} \to \mathbb{R}^{n+1}$ and set

$$\ell_i(x) = \text{proj}_{T_x\mathbb{H}^n}(n_i(x))$$

to obtain our neural network-parameterized vector fields. Observe that this vector field design is efficient and expressive, since $T_x\mathbb{H}^n \cong \mathbb{R}^n$.

A.2 VECTOR FIELD DESIGN FOR THE SPD MANIFOLD

Let $SPD(n)$ be the manifold of $n \times n$ SPD matrices with canonical metric, as in the main paper. We recall from Gallier & Quaintance (2020) that SPD has a Lie structure with algebra consisting of $n \times n$ symmetric matrices, denoted $S(n)$. The Riemannian exponential map (or equivalently, the matrix exponential map) is a bijection between $S(n)$ and $SPD(n)$. Recall by Lie symmetry (Gallier & Quaintance, 2020) that the tangent space at $X \in SPD(n)$ is given by:

$$T_X SPD(n) = XS(n) := \{Xy \mid y \in S(n)\}.$$

Observe that due to this tangent space structure, instead of utilizing the vector field construction given in Section A that requires an explicit projection operator, we may opt for more amenable designs oriented around the SPD manifold's Lie structure. We develop a variety of constructions below.

A.2.1 Design 1: Naïve

Most naïvely, we can observe that $SPD(n)$ is trivially embedded in $\mathbb{R}^{n^2}$, and so are its tangent vectors; we will use this observation to construct a simple vector field parameterization. Let vec : $\mathbb{R}^{n\times n} \to \mathbb{R}^{n^2}$ be row-major matrix vectorization and let $\text{vec}^{-1} : \mathbb{R}^{n^2} \to \mathbb{R}^{n\times n}$ be its inverse. Given a neural network $n_i : \mathbb{R}^{n^2} \to \mathbb{R}^{n^2}$ and an $X \in SPD(n)$, we may set:

$$\ell_i(X) = X\text{proj}_{S(n)}(\text{vec}^{-1}(n_i(\text{vec}(X))))$$

where $\text{proj}_{S(n)}$ is the typical matrix symmetrization operation given by:

$$\text{proj}_{S(n)}(X) = \frac{X + X^T}{2}.$$

Although this vector field representation is expressive, it also provides unneeded flexibility. For example, the intrinsic dimension of $T_X SPD(n) \cong S(n)$ is $\frac{n(n+1)}{2}$, but the n_i map to all of $\mathbb{R}^{n^2}$.

Based on this observation, we exploit tangent vector structure in the following vector field design to retain expressiveness while increasing efficiency.

A.2.2 Design 2: Structured

Observe that our tangent spaces satisfy $T_X SPD(n) \cong S(n)$, and moreover that $SPD(n) \subset S(n)$. We know that $S(n)$ has dimension $\frac{n(n+1)}{2}$ since each symmetric matrix is uniquely determined by its upper triangular part. Let $\iota : \mathbb{R}^{\frac{n(n+1)}{2}} \hookrightarrow S(n)$ be the row-major injection of the upper triangular part into a symmetric matrix and let $\iota^{-1} : S(n) \twoheadrightarrow \mathbb{R}^{\frac{n(n+1)}{2}}$ be its inverse. Given a neural network $n_i : \mathbb{R}^{\frac{n(n+1)}{2}} \to \mathbb{R}^{\frac{n(n+1)}{2}}$ and an $X \in SPD(n)$, we may set:

$$\ell_i(X) = X\iota(n_i(\iota^{-1}(X))).$$

Note that there is no longer any need for a projection to symmetric matrices, since we incorporate this structure directly into our vector field design. Moreover note that since $T_X SPD(n) \cong S(n) \cong \mathbb{R}^{\frac{n(n+1)}{2}}$, this vector field design is maximally expressive while being maximally efficient (representationally).

A.2.3 Design 3: Parsimonious

Although Design 2 is maximally expressive and efficient, in some cases where expressivity is less of a concern we may want a a reasonable parsimonious vector field design. Our answer to this is to directly parameterize a symmetric matrix via its upper triangular portion, and thereafter, generate a vector field via right multiplication. To be explicit, let our vector field be parameterized by euclidean parameters $v \in \mathbb{R}^{\frac{n(n+1)}{2}}$ and, for $X \in SPD(n)$, be given by:

$$\ell_i(X) = X\iota(v)$$

This is a learnable vector field induced by a single tangent vector at the identity. Although highly efficient, its location-agnosticism makes it highly inexpressive.

A.2.4 Design 4: Parsimonious Spectral

One may also consider exploiting manifold-specific structure in the context of Design 3 to produce a more expressive vector field that remains fairly efficient parametrically. A vector field design that accomplishes this is one that allows a map from the spectrum of the local SPD matrix to the spectrum of the symmetric matrix in the vector field construction. We let spec : $SPD(n) \to \mathbb{R}^n$ be the spectral map that takes SPD matrices to a vector of their eigenvalues, sorted in descending order. To be explicit, let our vector field be parameterized by $P \in O(n)$[6], a neural network $f_i : \mathbb{R}^n \to \mathbb{R}^n$, and, for $X \in SPD(n)$, be given by:

$$\ell_i(X) = XP\text{diag}(f_i(\text{spec}(X)))P^T$$

where diag : $\mathbb{R}^n \to \mathbb{R}^{n \times n}$ is the diagonal injection map. Observe that the spectrum of the symmetric matrix now depends locally on X, allowing for considerably more expressivity than in Design 3 at the cost of a low-dimensional neural network map $f_i : \mathbb{R}^n \to \mathbb{R}^n$. Moreover, the orthogonal constraint on P may be preserved throughout optimization via one of a variety of easy-to-implement methods (Casado & Martínez-Rubio, 2019; Anil et al., 2019).

Design 1 is naïve, but very inefficient. Design 2 exploits manifold structure to be maximally efficient while being maximally expressive. Design 3 showcases the other extreme (relative to Design 1) and gives a maximally parsimonious vector field construction. Design 4 showcases a more flexible version of Design 3 that allows for considerably greater learning capability[7] while still being representationally efficient. The purpose of describing these designs is to underscore the trade-off between expressivity and parameter-efficiency in designing parameterized vector fields (Designs 1 and 2 vs. Designs 3 and 4) as well as the need to utilize manifold-specific structure to obtain a maximally expressive and efficient vector field design (Design 1 vs. Design 2). Additionally, we highlight that expressivity for parameter-constrained vector field designs can be nontrivially increased with insignificant overhead via the introduction of manifold-specific dependencies (Design 3 vs. Design 4).

[6] $O(n)$ is the group of orthogonal matrices.

[7] Verified empirically.

B EXPERIMENTAL DETAILS

EXPERIMENTS ON HYPERBOLIC SPACE

B.1 ARCHITECTURAL DETAILS

We use the Poincaré disk model (Nickel & Kiela, 2017) to represent hyperbolic space. To test RResNet's performance on hyperbolic space we use a similar setup to Chami et al. (2019). First, in order to reduce the parameter count, we use a linear layer from the input dimension to a lower dimension before using RResNet as an encoder. For link-prediction tasks we use a Fermi-Dirac decoder and for node-classification tasks we use a linear decoder (Chami et al., 2019).

Architecture-related hyperparameters, including hidden dimension and number of layers, were optimized using a hyperparameter sweep.

B.2 RESULTS

Our models were trained using the Adam optimizer (Kingma & Ba, 2015) for 2000, 5000, and 10000 epochs on Airport link prediction, PubMed node classification, and Airport node classification, respectively. Training hyperparameters, including learning rate, weight decay, and dropout, were optimized using a hyperparameter sweep.

Dataset	Airport		PubMed	
Task	LP	NC	LP	NC
HNN	480	2628	–	8067
RResNet	**376**	**2343**	–	**3382**

Table 2: Parameter count (lower is better) for HNN baseline (Chami et al., 2019) versus RResNet.

All models have anywhere from $20\% - 50\%$ fewer parameters across tasks compared to the HNN baselines from Chami et al. (2019), as shown in Table 2. To obtain results, we collected metrics from 10 separate trials and reported mean and standard deviation.

EXPERIMENTS ON THE SPD MANIFOLD

B.3 ARCHITECTURAL DETAILS

We apply our model to the task of emotion recognition on the AFEW dataset (Dhall et al., 2011). Given videos encoded into 400×400 covariance matrices (which are trivially symmetric positive definite), our goal is to classify a matrix into one of seven classes. Because of how costly it would be to parameterize vector fields at this dimension, we use a BiMap layer (Huang & Gool, 2017), $\text{BiMap}_{d_{i+1}}^{d_i} : SPD(d_i) \to SPD(d_{i+1})$ as a base point remapping from 400×400 matrices to 50×50 matrices. We use vector field design 4 from Appendix A.1. In the context of this problem, we have:

$$\ell_1(X) = XP\text{diag}(f_1(\text{spec}(X)))P^T$$

where $f_1 : \mathbb{R}^{50} \to \mathbb{R}^{50}$, spec : $SPD(50) \to \mathbb{R}^{50}$, $P \in O(50)$. Note the vector field is a map $\ell_1 : SPD(50) \to T\,SPD(50)$. We express our forward pass as

$$g(x) = \exp_{\text{BiMap}_{50}^{400}(x)}(\ell_1(\text{BiMap}_{50}^{400}(x)))$$

which is a map $g : SPD(400) \to SPD(50)$. Thereafter we apply a logarithm to the eigenvalues of the 50×50 matrices (this helps linearize features Brooks et al. (2019)). Lastly we flatten the matrices and use a linear map from dimension 2500 to dimension 7 (representing the 7 different emotions). We use a simple cross entropy loss (Goodfellow et al., 2015) to train the model.

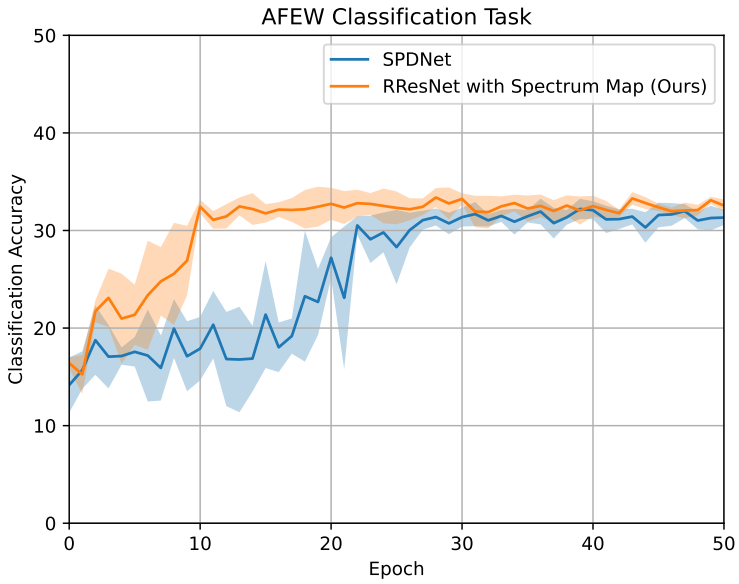

Figure 1: AFEW classification accuracies for RResNet (ours) compared to the SPDNet (Brooks et al., 2019) baseline. Results are averaged over five trials for each model. Error bars denote one standard deviation away from the mean accuracy. We observe that our model converges much faster than SPDNet.

B.4 Results

We compare our RResNet design above (Section B.3) to SPDNet (Brooks et al., 2019; Huang & Gool, 2017), a network architecture for SPD matrix learning. All models have a comparable number of parameters. To replicate the results of Brooks et al. (2019), we use a learning rate of $5 \cdot 10^{-2}$ for the baseline. We use a learning rate of $1 \cdot 10^{-1}$ for our model. We observe that our model has faster convergence than SPDNet (see Figure 1). Empirically, we observed that the beneficial effects of our RResNet construction match those of the SPD batch norm introduced in Brooks et al. (2019).

C Theoretical Results

We show that our construction agrees with the standard ResNet when the underlying manifold is Euclidean space. This aligns with our intuition and shows that our construction is a natural generalization of previous work.

Proposition 1. *When $\mathcal{M}^{(i)} \cong \mathbb{R}^{n_i}$, our RResNet is a standard residual network.*

Proof. Note that the vector fields take the form:

$$\ell_i(x) = \text{proj}_{T_x\mathbb{R}^n}(n_i(x)) = n_i(x)$$

meaning that our ℓ_i are standard neural networks. The $h_i : \mathbb{R}^{n_{i-1}} \to \mathbb{R}^{n_i}$ can be replaced by Euclidean linear layers that go from dimension n_{i-1} to dimension n_i. Also observe since $\exp_x(v) = x + v$, our neural network construction becomes:

$$\begin{aligned} f(x) &= x^{(m)} \\ x^{(0)} &= x \\ x^{(i)} &= \exp_{h_i(x^{(i-1)})}(\ell_i(h_i(x^{(i-1)}))) \\ &= h_i(x^{(i-1)}) + \ell_i(h_i(x^{(i-1)})) \\ &= h_i(x^{(i-1)}) + n_i(h_i(x^{(i-1)})) \end{aligned}$$

where the last equality holds $\forall i \in [m]$. Moreover, if all n_i are the same, we can use the identity map for our h_i, and we have:

$$x^{(i)} = x^{(i-1)} + n_i(x^{(i-1)}) \; \forall i \in [m]$$

Hence our neural network architecture reduces precisely to that of Euclidean residual neural networks. □