Riemannian Residual Neural Networks

Isay Katsman[*]
Yale University
isay.katsman@yale.edu

Eric M. Chen[*], **Sidhanth Holalkere**[*]
Cornell University
{emc348, sh844}@cornell.edu

Anna Asch
Cornell University
aca89@cornell.edu

Aaron Lou
Stanford University
aaronlou@stanford.edu

Ser-Nam Lim[†]
University of Central Florida
sernam@ucf.edu

Christopher De Sa
Cornell University
cdesa@cs.cornell.edu

Abstract

Recent methods in geometric deep learning have introduced various neural networks to operate over data that lie on Riemannian manifolds. Such networks are often necessary to learn well over graphs with a hierarchical structure or to learn over manifold-valued data encountered in the natural sciences. These networks are often inspired by and directly generalize standard Euclidean neural networks. However, extending Euclidean networks 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 in a geometrically principled manner. 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: when compared to existing manifold neural networks designed to learn over hyperbolic space and the manifold of symmetric positive definite matrices, we outperform both kinds of networks in terms of relevant testing metrics and training dynamics.

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 existing Euclidean neural network constructions enable performant learning.

Despite the ubiquity and success of Euclidean embeddings, recent research [41] has brought attention to the fact that several kinds of complex data require manifold considerations. Such data are various and range from covariance matrices, represented as points on the manifold of symmetric positive definite (SPD) matrices [26], to angular orientations, represented as points on tori, found in the context of robotics [43]. However, generalizing Euclidean neural network tools to manifold structures such as these can be quite difficult in practice. Most prior works design network architectures for a specific manifold [11, 17], thereby inefficiently necessitating a specific design for each new manifold.

We address this issue by extending Residual Neural Networks [23] to Riemannian manifolds in a way that naturally captures the underlying geometry. We construct our network by parameterizing vector fields and leveraging geodesic structure (provided by the Riemannian $\exp$ map) to "add" the learned

[*] indicates equal contribution.
[†] Work done while at Meta AI.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

vectors to the input points, thereby naturally generalizing a typical Euclidean residual addition. This process is illustrated in Figure 1. Note that this strategy is exceptionally natural, only making use of inherent geodesic geometry, and works generally for all smooth manifolds. We refer to such networks as Riemannian residual neural networks.

Though the above approach is principled, it is underspecified, as constructing an efficient learnable vector field for a given manifold is often nontrivial. To resolve this issue, we present a general way to induce a learnable vector field for a manifold $\mathcal{M}$ given only a map $f : \mathcal{M} \to \mathbb{R}^k$. Ideally, this map should capture intrinsic manifold geometry. For example, in the context of Euclidean space, this map could consist of a series of k projections onto hyperplanes. There is a natural equivalent of this in hyperbolic space that instead projects to horospheres (horospheres correspond to hyperplanes in Euclidean space). More generally, we propose a feature map that once more relies only on geodesic information, consisting of projection to random (or learned) geodesic balls. This final approach provides a fully geometric way to construct vector fields, and therefore natural residual networks, for any Riemannian manifold.

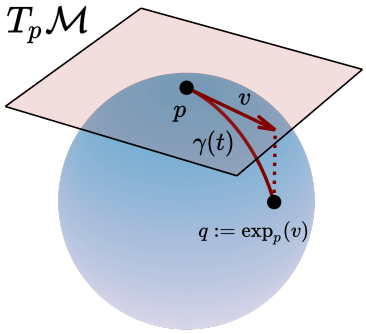

Figure 1: An illustration of a manifold-generalized residual addition. The traditional Euclidean formula $p \leftarrow p + v$ is generalized to $p \leftarrow \exp_p(v)$, where exp is the Riemannian exponential map. $\mathcal{M}$ is the manifold and $T_p\mathcal{M}$ is the tangent space at p.

After introducing our general theory, we give concrete manifestations of vector fields, and therefore residual neural networks, for hyperbolic space and the manifold of SPD matrices. We compare the performance of our Riemannian residual neural networks to that of existing manifold-specific networks on hyperbolic space and on the manifold of SPD matrices, showing that our networks perform much better in terms of relevant metrics due to their improved adherence to manifold geometry.

Our contributions are as follows:

1. We introduce a novel and principled generalization of residual neural networks to general Riemannian manifolds. Our construction relies only on knowledge of geodesics, which capture manifold geometry.
2. Theoretically, we show that our methodology better captures manifold geometry than pre-existing manifold-specific neural network constructions. Empirically, we apply our general construction to hyperbolic space and to the manifold of SPD matrices. On various hyperbolic graph datasets (where hyperbolicity is measured by Gromov δ-hyperbolicity) our method considerably outperforms existing work on both link prediction and node classification tasks. On various SPD covariance matrix classification datasets, a similar conclusion holds.
3. Our method provides a way to directly vary the geometry of a given neural network without having to construct particular operations on a per-manifold basis. This provides the novel capability to directly compare the effect of geometric representation (in particular, evaluating the difference between a given Riemannian manifold $(\mathcal{M}, g)$ and Euclidean space $(\mathbb{R}^n, ||\cdot||_2)$) while fixing the network architecture.

2 Related Work

Our work is related to but distinctly different from existing neural ordinary differential equation (ODE) [9] literature as well a series of papers that have attempted generalizations of neural networks to specific manifolds such as hyperbolic space [17] and the manifold of SPD matrices [26].

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 [23]. Later on, many discovered that by adding a learned residual, ResNets are similar to Euler's method [9, 21, 37, 45, 53]. 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 $\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t, \theta)$. Neural ODEs are defined precisely as ODEs of this form, where

the local dynamics are given by a parameterized neural network. Similar to our work, Falorsi and Forré [15], Katsman et al. [29], Lou et al. [36], Mathieu and Nickel [38] generalize neural ODEs to Riemannian manifolds (further generalizing manifold-specific work such as Bose et al. [3], that does this for hyperbolic space). However, instead of using a manifold's vector fields to solve a neural ODE, we learn an objective by parameterizing the vector fields directly (Figure 2). Neural ODEs and their generalizations to manifolds parameterize a continuous collection of vector fields over time for a single manifold in a dynamic flow-like construction. Our method instead parameterizes a discrete collection of vector fields, entirely untethered from any notion of solving an ODE. This makes our construction a strict generalization of both neural ODEs and their manifold equivalents [15, 29, 36, 38].

2.2 Riemannian Neural Networks

Past literature has attempted generalizations of Euclidean neural networks to a number of manifolds.

Hyperbolic Space Ganea et al. [17] extended basic neural network operations (e.g. activation function, linear layer, recurrent architectures) to conform with the geometry of hyperbolic space through gyrovector constructions [51]. In particular, they use gyrovector constructions [51] to build analogues of activation functions, linear layers, and recurrent architectures. Building on this approach, Chami et al. [8] adapt these constructions to hyperbolic versions of the feature transformation and neighborhood aggregation steps found in message passing neural networks. Additionally, batch normalization for hyperbolic space was introduced in Lou et al. [35]; hyperbolic attention network equivalents were introduced in Gülçehre et al. [20]. Although gyrovector constructions are algebraic and allow for generalization of neural network operations to hyperbolic space and beyond, we note that they do not capture intrinsic geodesic geometry. In particular, we note that the gyrovector-based hyperbolic linear layer introduced in Ganea et al. [17] reduces to a Euclidean matrix multiplication followed by a learned hyperbolic bias addition (see Appendix D.2). Hence all non-Euclidean learning for this case happens through the bias term. In an attempt to resolve this, further work has focused on imbuing these neural networks with more hyperbolic functions [10, 49]. Chen et al. [10] notably constructs a hyperbolic residual layer by projecting an output onto the Lorentzian manifold. However, we emphasize that our construction is more general while being more geometrically principled as we work with fundamental manifold operations like the exponential map rather than relying on the niceties of Lorentz space.

Yu and De Sa [55] make use of randomized hyperbolic Laplacian features to learn in hyperbolic space. We note that the features learned are shallow and are constructed from a specific manifestation of the Laplace-Beltrami operator for hyperbolic space. In contrast, our method is general and enables non-shallow (i.e., multi-layer) feature learning.

SPD Manifold Neural network constructs have been extended to the manifold of symmetric positive definite (SPD) matrices as well. In particular, SPDNet [26] is an example of a widely adopted SPD manifold neural network which introduced SPD-specific layers analogous to Euclidean linear and ReLU layers. Building upon SPDNet, Brooks et al. [5] developed a batch normalization method to be used with SPD data. Additionally, López et al. [34] adapted gyrocalculus constructions used in hyperbolic space to the SPD manifold.

Symmetric Spaces Further work attempts generalization to symmetric spaces. Sonoda et al. [50] design fully-connected networks over noncompact symmetric spaces using particular theory from Helgason-Fourier analysis [25], and Chakraborty et al. [7] attempt to generalize several operations such as convolution to such spaces by adapting and developing a weighted Fréchet mean construction. We note that the Helgason-Fourier construction in Sonoda et al. [50] exploits a fairly particular structure, while the weighted Fréchet mean construction in Chakraborty et al. [7] is specifically introduced for convolution, which is not the focus of our work (we focus on residual connections).

Unlike any of the manifold-specific work described above, our residual network construction can be applied generally to any smooth manifold and is constructed solely from geodesic information.

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 [32].

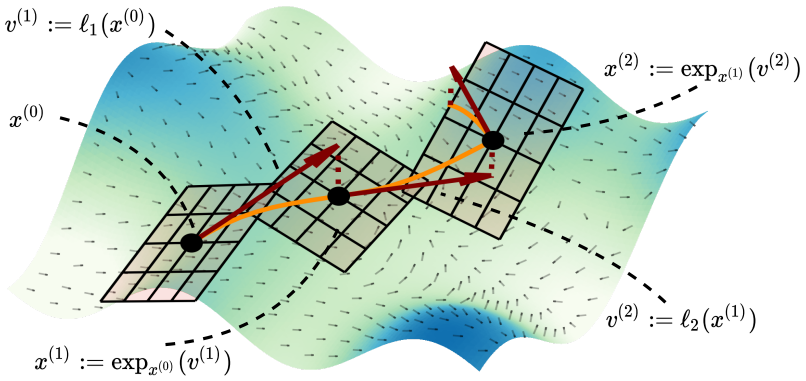

Figure 2: A visualization of a Riemannian residual neural network on a manifold $\mathcal{M}$. Our model parameterizes vector fields on a manifold. At each layer in our network, we take a step from a point in the direction of that vector field (brown), which is analogous to the residual step in a ResNet.

3.1 Riemannian Geometry

A topological manifold $(\mathcal{M}, g)$ of dimension n is a locally Euclidean space, meaning there exist homeomorphic[1] functions (called "charts") whose domains both cover the manifold and map from the manifold into $\mathbb{R}^n$ (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 are smooth bijections mapping into $\mathbb{R}^n$ and have smooth inverses. We denote $T_p\mathcal{M}$ as the tangent space at a point p of the manifold $\mathcal{M}$. Further still, a Riemannian manifold[2] $(\mathcal{M}, g)$ is an n-dimensional smooth manifold with a smooth collection of inner products $(g_p)_{p\in\mathcal{M}}$ for every tangent space $T_p\mathcal{M}$. The Riemannian metric g induces a distance $d_g : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$ on the manifold.

3.2 Geodesics and the Riemannian Exponential Map

Geodesics A geodesic is a 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:

$$d(p, q) = \inf_{\gamma} \int_0^1 \sqrt{g_{\gamma(t)}(\gamma'(t), \gamma'(t))}\, dt \tag{1}$$

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 [32].

Exponential Map The Riemannian exponential map is a way to map $T_p\mathcal{M}$ to a neighborhood around p using geodesics. The relationship between the tangent space and the exponential map output 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 to capture the local on-manifold behavior corresponding to the tangent space operations. For $p \in \mathcal{M}$ and $v \in T_p\mathcal{M}$, the exponential map at p is defined as $\exp_p(v) = \gamma_v(1)$.

One can think of $\exp$ as a manifold generalization of Euclidean addition, since in the Euclidean case we have $\exp_p(v) = p + v$.

[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 [32].

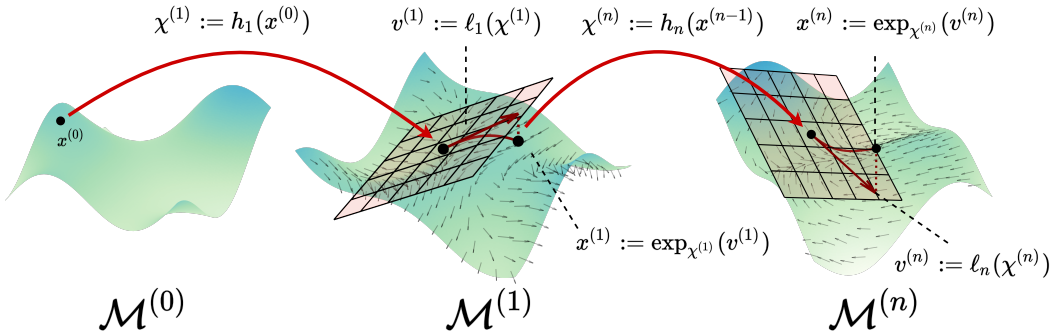

Figure 3: An overview of our generalized Riemannian Residual Neural Network (RResNet) methodology. We start by mapping $x^{(0)} \in \mathcal{M}^{(0)}$ to $\chi^{(1)} \in \mathcal{M}^{(1)}$ using a base point mapping h_1. Then, using our paramterized vector field ℓ_i, we compute a residual $v^{(1)} := \ell_1(\chi^{(1)})$. Finally, we project $v^{(1)}$ back onto the manifold using the Riemannian exp map, leaving us with $x^{(1)}$. This procedure can be iterated to produce a multi-layer Riemannian residual neural network that is capable of changing manifold representation on a per layer basis.

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 $X_p \in T_p\mathcal{M}$. 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.

Tangent Bundle The tangent bundle of a smooth manifold $\mathcal{M}$ is the disjoint union of the tangent spaces $T_p\mathcal{M}$, for all $p \in \mathcal{M}$, denoted by $T\mathcal{M} := \bigsqcup_{p\in\mathcal{M}} T_p\mathcal{M} = \bigsqcup_{p\in\mathcal{M}} \{(p, v) \mid v \in T_p\mathcal{M}\}$.

Pushforward A derivative (also called a *pushforward*) of a map $f : \mathcal{M} \to \mathcal{N}$ between two manifolds is denoted by $D_pf : T_p\mathcal{M} \to T_{f(p)}\mathcal{N}$. This is a generalization of the classical Euclidean Jacobian (since $\mathbb{R}^n$ is a manifold), and provides a way to relate tangent spaces at different points on different manifolds.

Pullback Given $\phi : \mathcal{M} \to \mathcal{N}$ a smooth map between manifolds and $f : \mathcal{N} \to \mathbb{R}$ a smooth function, the pullback of f by ϕ is the smooth function $\phi^* f$ on $\mathcal{M}$ defined by $(\phi^* f)(x) = f(\phi(x))$. When the map ϕ is implicit, we simply write f^* to mean the pullback of f by ϕ.

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 $\mathbb{S}^n$, that encompass all manifolds with constant sectional curvature. Hyperbolic space manifests in several representations like the Poincaré ball, Lorentz space, and the Klein model. We use the Poincaré ball model for our Riemannian ResNet design (see Appendix A for more details on the Poincaré ball model).

3.5 SPD Manifold

Let $SPD(n)$ be the manifold of $n \times n$ symmetric positive definite (SPD) matrices. We recall from Gallier and Quaintance [16] that $SPD(n)$ has a Riemannian exponential map (at the identity) equivalent to the matrix exponential. Two common metrics used for $SPD(n)$ are the log-Euclidean metric [16], which induces a flat structure on the matrices, and the canonical affine-invariant metric [12, 42], which induces non-constant negative sectional curvature. The latter gives $SPD(n)$ a considerably less trivial geometry than that exhibited by the Riemannian model spaces [2] (see Appendix A for more details on $SPD(n)$).

4 Methodology

In this section, we provide the technical details behind Riemannian residual neural networks.

4.1 General Construction

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

$$f(x) := x^{(m)} \tag{2}$$

$$x^{(0)} := x \tag{3}$$

$$x^{(i)} := \exp_{x^{(i-1)}}(\ell_i(x^{(i-1)})) \tag{4}$$

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 vector field over $\mathcal{M}$. This residual network construction is visualized for the purpose of intuition in Figure 2. In practice, parameterizing a function from an abstract manifold $\mathcal{M}$ to its tangent bundle is difficult. However, by the Whitney embedding theorem [33], we can embed $\mathcal{M} \hookrightarrow \mathbb{R}^D$ smoothly 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

$$\ell_i(x) := \text{proj}_{T_x\mathcal{M}}(n_i(x)) \tag{5}$$

where we note that $T_x\mathcal{M} \subset \mathbb{R}^D$ is a linear subspace (making the projection operator well defined). Throughout the paper we call this the embedded vector field design[3]. We note that this is the same construction used for defining the vector field flow in Lou et al. [36], Mathieu and Nickel [38], Rozen et al. [44].

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 $f : \mathcal{M}^{(0)} \to \mathcal{M}^{(m)}$ is given by

$$f(x) := x^{(m)} \tag{6}$$

$$x^{(0)} := x \tag{7}$$

$$x^{(i)} := \exp_{h_i(x^{(i-1)})}(\ell_i(h_i(x^{(i-1)}))) \forall i \in [m] \tag{8}$$

with functions $\ell_i : \mathcal{M}^{(i)} \to T\mathcal{M}^{(i)}$ given as above. This generalization is visualized in Figure 3. In practice, our $\mathcal{M}^{(i)}$ will be different dimensional versions of the same geometric space (e.g. $\mathbb{H}^n$ or $\mathbb{R}^n$ for varying n). If the starting and ending manifolds are the same, the maps h_i will simply be standard inclusions. When the starting and ending manifolds are different, the h_i may be standard neural networks for which we project the output, or the h_i may be specially design learnable maps that respect manifold geometry. As a concrete example, our h_i for the SPD case map from an SPD matrix of one dimension to another by conjugating with a Stiefel matrix [26]. Furthermore, as shown in Appendix D, our model is equivalent to the standard ResNet when the underlying manifold is $\mathbb{R}^n$.

Comparison with Other Constructions We discuss how our construction compares with other methods in Appendix E, but here we briefly note that unlike other methods, our presented approach is fully general and better conforms with manifold geometry.

4.2 Feature Map-Induced Vector Field Design

Most of the difficulty in application of our general vector field construction comes from the design of the learnable vector fields $\ell_i : \mathcal{M}^{(i)} \to T\mathcal{M}^{(i)}$. Although we give an embedded vector field design above, it is not very principled geometrically. We would like to considerably restrict these vector fields so that their range is informed by the underlying geometry of $\mathcal{M}$. For this, we note that it is possible to induce a vector field $\xi : \mathcal{M} \to T\mathcal{M}$ for a manifold $\mathcal{M}$ with any smooth map $f : \mathcal{M} \to \mathbb{R}^k$. In practice, this map should capture intrinsic geometric properties of $\mathcal{M}$ and can be viewed as a feature map, or de facto linearization of $\mathcal{M}$. Given an $x \in \mathcal{M}$, we need only pass x through f to get its feature representation in $\mathbb{R}^k$, then note that since:

$$D_p f : T_p\mathcal{M} \to T_{f(p)}\mathbb{R}^k,$$

we have an induced map:

$$(D_p f)^* : (T_{f(p)}\mathbb{R}^k)^* \to (T_p\mathcal{M})^*,$$

where $(D_p f)^*$ is the pullback of $D_p f$. Note that $T_p\mathbb{R}^k \cong \mathbb{R}^k$ and $(\mathbb{R}^k)^* \cong \mathbb{R}^k$ by the dual space isomorphism. Moreover $(T_p\mathcal{M})^* \cong T_p\mathcal{M}$ by the tangent-cotangent space isomorphism [33]. Hence, we have the induced map:

$$(D_p f)^*_r : \mathbb{R}^k \to T_p\mathcal{M},$$

[3]Ideal vector field design is in general nontrivial and the embedded vector field is not a good choice for all manifolds (see Appendix B).

obtained from $(D_p f)^*$, simply by both precomposing and postcomposing the aforementioned isomorphisms, where relevant. $(D_p f)^*_r$ provides a natural way to map from the feature representation to the tangent bundle. Thus, we may view the map $\ell_f : \mathcal{M} \to T\mathcal{M}$ given by:

$$\ell_f(x) = (D_x f)^*_r(f(x))$$

as a deterministic vector field induced entirely by f.

Learnable Feature Map-Induced Vector Fields We can easily make the above vector field construction learnable by introducing a Euclidean neural network $n_\theta : \mathbb{R}^k \to \mathbb{R}^k$ after f to obtain $\ell_{f,\theta}(x) = (D_x f)^*(n_\theta(f(x)))$.

Feature Map Design One possible way to simplify the design of the above vector field is to further break down the map $f : \mathcal{M} \to \mathbb{R}^k$ into k maps $f_1, \ldots, f_k : \mathcal{M} \to \mathbb{R}$, where ideally, each map f_i is constructed in a similar way (e.g. performing some kind of geometric projection, where the f_i vary only in terms of the specifying parameters). As we shall see in the following subsection, this ends up being a very natural design decision.

In what follows, we shall consider only smooth feature maps $f : \mathcal{M} \to \mathbb{R}^k$ induced by a single parametric construction $g_\theta : \mathcal{M} \to \mathbb{R}$, i.e. the k dimensions of the output of f are given by different choices of θ for the same underlying feature map[4]. This approach also has the benefit of a very simple interpretation of the induced vector field. Given feature maps $g_{\theta_1}, \ldots, g_{\theta_k} : \mathcal{M} \to \mathbb{R}$ that comprise our overall feature map $f : \mathcal{M} \to \mathbb{R}^k$, our vector field is simply a linear combination of the maps $\nabla g_{\theta_i} : \mathcal{M} \to T\mathcal{M}$. If the g_{θ_i} are differentiable with respect to θ_i, we can even learn the θ_i themselves.

4.2.1 Manifold Manifestations

In this section, in an effort to showcase how simple it is to apply our above theory to come up with natural vector field designs, we present several constructions of manifold feature maps $g_\theta : \mathcal{M} \to \mathbb{R}$ that capture the underlying geometry of $\mathcal{M}$ for various choices of $\mathcal{M}$. Namely, in this section we provide several examples of $f : \mathcal{M} \to \mathbb{R}$ that induce $\ell_f : \mathcal{M} \to T\mathcal{M}$, thereby giving rise to a Riemannian neural network by Section 4.1.

Euclidean Space To build intuition, we begin with an instructive case. We consider designing a feature map for the Euclidean space $\mathbb{R}^n$. A natural design would follow simply by considering hyperplane projection. Let a hyperplane $w^T x + b = 0$ be specified by $w \in \mathbb{R}^n, b \in \mathbb{R}$. Then a natural feature map $g_{w,b} : \mathbb{R}^n \to \mathbb{R}$ parameterized by the hyperplane parameters is given by hyperplane projection [14]: $g_{w,b}(x) = \frac{|w^T x + b|}{||w||_2}$.

Hyperbolic Space We wish to construct a natural feature map for hyperbolic space. Seeking to follow the construction given in the Euclidean context, we wish to find a hyperbolic analog of hyperplanes. This is provided to us via the notion of horospheres [24]. Illustrated in Figure 4, horospheres naturally generalize hyperplanes to hyperbolic space. We specify a horosphere in the Poincaré ball model of hyperbolic space $\mathbb{H}^n$ by a point of tangency $\omega \in \mathbb{S}^{n-1}$ and a real value $b \in \mathbb{R}$. Then a natural feature map $g_{\omega,b} : \mathbb{H}^n \to \mathbb{R}$ parameterized by the horosphere parameters would be given by horosphere projection [4]: $g_{\omega,b}(x) = -\log\left(\frac{1-||x||_2^2}{||x-\omega||_2^2}\right) + b$.

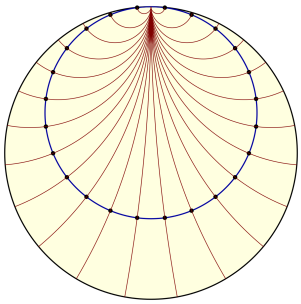

Figure 4: Example of a horosphere in the Poincaré ball representation of hyperbolic space. In this particular two-dimensional case, the hyperbolic space $\mathbb{H}_2$ is visualized via the Poincaré disk model, and the horosphere, shown in blue, is called a horocycle.

Symmetric Positive Definite Matrices The manifold of SPD matrices is an example of a manifold where there is no innate representation of a hyperplane. Instead, given $X \in SPD(n)$, a reasonable feature map $g_k : SPD(n) \to \mathbb{R}$, parameterized by k, is to map X to its kth largest eigenvalue: $g_k(X) = \lambda_k$.

[4]We use the term "feature map" for both the overall feature map $f : \mathcal{M} \to \mathbb{R}^k$ and for the inducing construction $g_\theta : \mathcal{M} \to \mathbb{R}$. This is well-defined since in our work we consider only feature maps $f : \mathcal{M} \to \mathbb{R}^k$ that are induced by some $g_\theta : \mathcal{M} \to \mathbb{R}$.

General Manifolds For general manifolds there is no perfect analog of a hyperplane, and hence there is no immediately natural feature map. Although this is the case, it is possible to come up with a reasonable alternative. We present such an alternative in Appendix B.4 together with pertinent experiments.

Example: Euclidean Space One motivation for the vector field construction $\ell_f(x) = (D_x f)^*_r(f(x))$ is that in the Euclidean case, ℓ_f will reduce to a standard linear layer (because the maps f and $(D_x f)^*$ are linear), which, in combination with the Euclidean $\exp$ map, will produce a standard Euclidean residual neural network.

Explicitly, for the Euclidean case, note that our feature map $f : \mathbb{R}^n \to \mathbb{R}^k$ will, for example, take the form $f(x) = Wx, W \in \mathbb{R}^{k\times n}$ (here we have $b = 0$ and W has normalized row vectors). Then note that we have $Df = W$ and $(Df)^* = W^T$. We see for the standard feature map-based construction, our vector field $\ell_f(x) = (D_x f)^*(f(x))$ takes the form $\ell_f(x) = W^T Wx$.

For the learnable case (which is standard for us, given that we learn Riemannian residual neural networks), when the manifold is Euclidean space, the general expression $\ell_{f,\theta}(x) = (D_x f)^*(n_\theta(f(x)))$ becomes $\ell_{f,\theta}(x) = W^T n_\theta(Wx)$. When the feature maps are trivial projections (onto axis-aligned hyperplanes), we have $W = I$ and $\ell_{f,\theta}(x) = n_\theta(x)$. Thus our construction can be viewed as a generalization of a standard neural network.

	Dataset	Disease		Airport		PubMed		CoRA	
	Hyperbolicity	$\delta = 0$		$\delta = 1$		$\delta = 3.5$		$\delta = 11$	
	Task	LP	NC	LP	NC	LP	NC	LP	NC
Shallow	Euc	$59.8_{\pm 2.0}$	$32.5_{\pm 1.1}$	$92.0_{\pm 0.0}$	$60.9_{\pm 3.4}$	$83.3_{\pm 0.1}$	$48.2_{\pm 0.7}$	$82.5_{\pm 0.3}$	$23.8_{\pm 0.7}$
	Hyp [41]	$63.5_{\pm 0.6}$	$45.5_{\pm 3.3}$	$94.5_{\pm 0.0}$	$70.2_{\pm 0.1}$	$87.5_{\pm 0.1}$	$68.5_{\pm 0.3}$	$87.6_{\pm 0.2}$	$22.0_{\pm 1.5}$
	Euc-Mixed	$49.6_{\pm 1.1}$	$35.2_{\pm 3.4}$	$91.5_{\pm 0.1}$	$68.3_{\pm 2.3}$	$86.0_{\pm 1.3}$	$63.0_{\pm 0.3}$	$84.4_{\pm 0.2}$	$46.1_{\pm 0.4}$
	Hyp-Mixed	$55.1_{\pm 1.3}$	$56.9_{\pm 1.5}$	$93.3_{\pm 0.0}$	$69.6_{\pm 0.1}$	$83.8_{\pm 0.3}$	$\mathbf{73.9}_{\pm 0.2}$	$85.6_{\pm 0.5}$	$45.9_{\pm 0.3}$
NN	MLP	$72.6_{\pm 0.6}$	$28.8_{\pm 2.5}$	$89.8_{\pm 0.5}$	$68.6_{\pm 0.6}$	$84.1_{\pm 0.9}$	$72.4_{\pm 0.2}$	$83.1_{\pm 0.5}$	$51.5_{\pm 1.0}$
	HNN [17]	$75.1_{\pm 0.3}$	$41.0_{\pm 1.8}$	$90.8_{\pm 0.2}$	$80.5_{\pm 0.5}$	$\mathbf{94.9}_{\pm 0.1}$	$69.8_{\pm 0.4}$	$\mathbf{89.0}_{\pm 0.1}$	$\mathbf{54.6}_{\pm 0.4}$
	RResNet Horo	$\mathbf{98.4}_{\pm 0.3}$	$\mathbf{76.8}_{\pm 2.0}$	$\mathbf{95.2}_{\pm 0.1}$	$\mathbf{96.9}_{\pm 0.3}$	$\mathbf{95.0}_{\pm 0.3}$	$72.3_{\pm 1.7}$	$86.7_{\pm 6.3}$	$52.4_{\pm 5.5}$

Table 1: Above we give graph task results for RResNet Horo compared with several non-graph-based neural network baselines (baseline methods and metrics are from Chami et al. [8]). Test ROC AUC is the metric reported for link prediction (LP) and test F1 score is the metric reported for node classification (NC). Mean and standard deviation are given over five trials. Note that RResNet Horo considerably outperforms HNN on the most hyperbolic datasets, performing worse and worse as hyperbolicity increases, to a more extreme extent than previous methods that do not adhere to geometry as closely (this is expected).

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 SPD manifold.

5.1 Hyperbolic Space

We perform numerous experiments in the hyperbolic setting. The purpose is twofold:

1. We wish to illustrate that our construction in Section 4 is not only more general, but also intrinsically more geometrically natural than pre-existing hyperbolic constructions such as HNN [17], and is thus able to learn better over hyperbolic data.
2. We would like to highlight that non-Euclidean learning benefits the most hyperbolic datasets. We can do this directly since our method provides a way to vary the geometry of a fixed neural network architecture, thereby allowing us to directly investigate the effect of changing geometry from Euclidean to hyperbolic.

5.1.1 Direct Comparison Against Hyperbolic Neural Networks [17]

To demonstrate the improvement of RResNet over HNN [17], we first perform node classification (NC) and link prediction (LP) tasks on graph datasets with low Gromov δ-hyperbolicity [8], which means the underlying structure of the data is highly hyperbolic. The RResNet model is given the

	AFEW[13]	FPHA[18]	NTU RGB+D[48]	HDM05[39]
SPDNet	$33.24_{\pm 0.56}$	$65.39_{\pm 1.48}$	$41.47_{\pm 0.34}$	$66.77_{\pm 0.92}$
SPDNetBN	$35.39_{\pm 0.93}$	$65.03_{\pm 1.35}$	$41.92_{\pm 0.37}$	$67.25_{\pm 0.44}$
RResNet Affine-Invariant	$35.17_{\pm 1.78}$	$\mathbf{66.53_{\pm 01.64}}$	$41.00_{\pm 0.50}$	$67.91_{\pm 1.27}$
RResNet Log-Euclidean	$\mathbf{36.38_{\pm 1.29}}$	$64.58_{\pm 0.98}$	$\mathbf{42.99_{\pm 0.23}}$	$\mathbf{69.80_{\pm 1.51}}$

Table 2: We run our SPD manifold RResNet on four SPD matrix datasets and compare against SPDNet [26] and SPDNet with batch norm [5]. We report the mean and standard deviation of validation accuracies over five trials and bold which method performs the best.

name "RResNet Horo." It utilizes a horosphere projection feature map-induced vector field described in Section 4. All model details are given in Appendix C.2. We find that because we adhere well to the geometry, we attain good performance on datasets with low Gromov δ-hyperbolicities (e.g. $\delta = 0, \delta = 1$). As soon as the Gromov hyperbolicity increases considerably beyond that (e.g. $\delta = 3.5, \delta = 11$), performance begins to degrade since we are embedding non-hyperbolic data in an unnatural manifold geometry. Since we adhere to the manifold geometry more strongly than prior hyperbolic work, we see performance decay faster as Gromov hyperbolicity increases, as expected. In particular, we test on the very hyperbolic Disease ($\delta = 0$) [8] and Airport ($\delta = 1$) [8] datasets. We also test on the considerably less hyperbolic PubMed ($\delta = 3.5$) [47] and CoRA ($\delta = 11$) [46] datasets. We use all of the non-graph-based baselines from Chami et al. [8], since we wish to see how much we can learn strictly from a proper treatment of the embeddings (and no graph information). Table 1 summarizes the performance of "RResNet Horo" relative to these baselines.

Moreover, we find considerable benefit from the feature map-induced vector field over an embedded vector field that simply uses a Euclidean network to map from a manifold point embedded in $\mathbb{R}^n$. The horosphere projection captures geometry more accurately, and if we swap to an embedded vector field we see considerable accuracy drops on the two hardest hyperbolic tasks: Disease NC and Airport NC. In particular, for Disease NC the mean drops from 76.8 to 75.0, and for Airport NC we see a very large decrease from 96.9 to 83.0, indicating that geometry captured with a well-designed feature map is especially important. We conduct a more thorough vector field ablation study in Appendix C.5.

5.1.2 Impact of Geometry

A major strength of our method is that it allows one to investigate the direct effect of geometry in obtaining results, since the architecture can remain the same for various manifolds and geometries (as specified by the metric of a given Riemannian manifold). This is well-illustrated in the most hyperbolic Disease NC setting, where swapping out hyperbolic for Euclidean geometry in an RResNet induced by an embedded vector field decreases the F1 score from a 75.0 mean to a 67.3 mean and induces a large amount of numerical stability, since standard deviation increases from 5.0 to 21.0. We conduct a more thorough geometry ablation study in Appendix C.5.

5.2 SPD Manifold

A common application of SPD manifold-based models is learning over full-rank covariance matrices, which lie on the manifold of SPD matrices. We compare our RResNet to SPDNet [26] and SPDNet with batch norm [5] on four video classification datasets: AFEW [13], FPHA [18], NTU RGB+D [48], and HDM05 [39]. Results are given in Table 2. Please see Appendix C.6 for details on the experimental setup. For our RResNet design, we try two different metrics: the log-Euclidean metric [16] and the affine-invariant metric [12, 42], each of which captures the curvature of the SPD manifold differently. We find that adding a learned residual improves performance and training dynamics over existing neural networks on SPD manifolds with little effect on runtime. We experiment with several vector field designs, which we outline in Appendix B. The best vector field design (given in Section 4.2), also the one we use for all SPD experiments, necessitates eigenvalue computation. We note the cost of computing eigenvalues is not a detrimental feature of our approach since previous works (SPDNet [26], SPDNet with batchnorm [5]) already make use of eigenvalue computation[5]. Empirically, we observe that the beneficial effects of our RResNet construction are similar to those of the SPD batch norm introduced in Brooks et al. [5] (Table 2, Figure 5 in Appendix C.6). In addition, we find that our operations are stable with ill-conditioned input matrices, which commonly occur in the wild. To contrast, the batch norm computation in SPDNetBN, which relies on Karcher flow

[5]One needs this computation for operations such as the Riemannian $\exp$ and $\log$ over the SPD manifold.

	Dataset	Disease	Airport	PubMed	CoRA
	Hyperbolicity	$\delta = 0$	$\delta = 1$	$\delta = 3.5$	$\delta = 11$
GNN	GCN [31]	$69.7_{\pm 0.4}$	$81.4_{\pm 0.6}$	$78.1_{\pm 0.2}$	$81.3_{\pm 0.3}$
GNN	GAT [52]	$70.4_{\pm 0.4}$	$81.5_{\pm 0.3}$	$79.0_{\pm 0.3}$	$\mathbf{83.0}_{\pm 0.7}$
GNN	SAGE [22]	$69.1_{\pm 0.6}$	$82.1_{\pm 0.5}$	$77.4_{\pm 2.2}$	$77.9_{\pm 2.4}$
GNN	SGC [54]	$69.5_{\pm 0.2}$	$80.6_{\pm 0.1}$	$78.9_{\pm 0.0}$	$81.0_{\pm 0.1}$
GGNN	HGCN [8]	$74.5_{\pm 0.9}$	$90.6_{\pm 0.2}$	$\mathbf{80.3}_{\pm 0.3}$	$79.9_{\pm 0.2}$
GGNN	Fully HNN [10]	$\mathbf{96.0}_{\pm 1.0}$	$90.9_{\pm 1.4}$	$78.0_{\pm 1.0}$	$80.2_{\pm 1.3}$
GGNN	**G-RResNet Horo**	$\mathbf{95.4}_{\pm 1.0}$	$\mathbf{97.4}_{\pm 0.1}$	$75.5_{\pm 0.8}$	$64.4_{\pm 7.6}$

Table 3: Above we give node classification results for G-RResNet Horo compared with several graph-based neural network baselines (baseline methods and metrics are from Chami et al. [8]). Test F1 score is the metric reported. Mean and standard deviation are given over five trials. Note that G-RResNet Horo obtains a state-of-the-art result on Airport. As for the less hyperbolic datasets, G-RResNet Horo does worse on PubMed and does very poorly on CoRA, once more, as expected due to unsuitability of geometry. The GNN label stands for "Graph Neural Networks" and the GGNN label stands for "Geometric Graph Neural Networks."

[28, 35], suffers from numerical instability when the input matrices are nearly singular. Overall, we observe our RResNet with the affine-invariant metric outperforms existing work on FPHA, and our RResNet using the log-Euclidean metric outperforms existing work on AFEW, NTU RGB+D, and HDM05. Being able to directly interchange between two metrics while maintaining the same neural network design is an unique strength of our model.

6 Riemannian Residual Graph Neural Networks

Following the initial comparison to non-graph-based methods in Table 1, we introduce a simple graph-based method by modifying RResNet Horo above. We take the previous model and pre-multiply the feature map output by the underlying graph adjacency matrix A in a manner akin to what happens with graph neural networks [54]. This is the simple modification that we introduce to the Riemannian ResNet to incorporate graph information; we call this method G-RResNet Horo. We compare directly against the graph-based methods in Chami et al. [8] as well as against Fully Hyperbolic Neural Networks [10] and give results in Table 3. We test primarily on node classification since we found that almost all LP tasks are too simple and solved by methods in Chami et al. [8] (i.e., test ROC is greater than 95%). We also tune the matrix power of A for a given dataset; full architectural details are given in Appendix C.2. Although this method is simple, we see further improvement and in fact attain a state-of-the-art result for the Airport [8] dataset. Once more, as expected, we see a considerable performance drop for the much less hyperbolic datasets, PubMed and CoRA.

7 Conclusion

We propose a general construction of residual neural networks on Riemannian manifolds. Our approach is a natural geodesically-oriented generalization that can be applied more broadly than previous manifold-specific work. Our introduced neural network construction is the first that decouples geometry (i.e. the representation space expected for input to layers) from the architecture design (i.e. actual "wiring" of the layers). Moreover, we introduce a geometrically principled feature map-induced vector field design for the RResNet. We demonstrate that our methodology better captures underlying geometry than existing manifold-specific neural network constructions. On a variety of tasks such as node classification, link prediction, and covariance matrix classification, our method outperforms previous work. Finally, our RResNet's principled construction allows us to directly assess the effect of geometry on a task, with neural network architecture held constant. We illustrate this by directly comparing the performance of two Riemannian metrics on the manifold of SPD matrices. We hope others will use our work to better learn over data with nontrivial geometries in relevant fields, such as lattice quantum field theory, robotics, and computational chemistry.

Limitations We rely fundamentally on knowledge of geodesics of the underlying manifold. As such, we assume that a closed form (or more generally, easily computable, differentiable form) is given for the Riemannian exponential map as well as for the tangent spaces.

Acknowledgements

We would like to thank Facebook AI for funding equipment that made this work possible. In addition, we thank the National Science Foundation for awarding Prof. Christopher De Sa a grant that helps fund this research effort (NSF IIS-2008102) and for supporting both Isay Katsman and Aaron Lou with graduate research fellowships. We would also like to acknowledge Prof. David Bindel for his useful insights on the numerics of SPD matrices.

References

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

[2] Rajendra Bhatia. Geometry of positive matrices. In *Positive Definite Matrices*, pages 201–236. Princeton University Press, 2007. ISBN 9780691129181. URL `http://www.jstor.org/stable/j.ctt7rxv2.9`.

[3] Joey Bose, Ariella Smofsky, Renjie Liao, Prakash Panangaden, and Will Hamilton. Latent variable modelling with hyperbolic normalizing flows. In *Proceedings of the 37th International Conference on Machine Learning*, pages 1045–1055, 2020.

[4] Martin R. Bridson and Andr'e Haefliger. Metric spaces of non-positive curvature. 1999.

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

[6] 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.

[7] Rudrasis Chakraborty, Jose J. Bouza, Jonathan H. Manton, and Baba C. Vemuri. Manifoldnet: A deep neural network for manifold-valued data with applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44:799–810, 2018.

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

[9] 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, pages 6571–6583, 2018.

[10] Weize Chen, Xu Han, Yankai Lin, Hexu Zhao, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. Fully hyperbolic neural networks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2021.

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

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

[13] 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)*, pages 2106–2112, 2011.

[14] Laurent El Ghaoui. Hyper-textbook: Optimization models and applications, 2021. URL `https://inst.eecs.berkeley.edu/~ee127/sp21/livebook/l_vecs_hyp.html`.

[15] Luca Falorsi and Patrick Forré. Neural ordinary differential equations on manifolds. *arXiv preprint arXiv:2006.06663*, 2020.

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

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

[18] Guillermo Garcia-Hernando, Shanxin Yuan, Seungryul Baek, and Tae-Kyun Kim. First-person hand action benchmark with rgb-d videos and 3d hand pose annotations. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 409–419, 2018.

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

[20] Caglar Gülçehre, Misha Denil, Mateusz Malinowski, Ali Razavi, Razvan Pascanu, Karl Moritz Hermann, Peter W. Battaglia, Victor Bapst, David Raposo, Adam Santoro, and Nando de Freitas. Hyperbolic attention networks. *ArXiv*, abs/1805.09786, 2019.

[21] Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. *Inverse Problems*, 34, 2017.

[22] William L. Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NIPS*, 2017.

[23] 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)*, pages 770–778, 2016.

[24] Ernst Heintze and H I Hof. Geometry of horospheres. *Journal of Differential Geometry*, 12: 481–491, 1977.

[25] Sigurdur Helgason. Radon-fourier transforms on symmetric spaces and related group representations. *Bulletin of the American Mathematical Society*, 71:757–763, 1965.

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

[27] Mohamed E. Hussein, Marwan Torki, Mohammad Abdelaziz Gowayyed, and Motaz Ahmad El-Saban. Human action recognition using a temporal hierarchy of covariance descriptors on 3d joint locations. In *IJCAI*, 2013.

[28] H. Karcher. Riemannian center of mass and mollifier smoothing. *Communications on Pure and Applied Mathematics*, 30(5):509–541, 1977. doi: https://doi.org/10.1002/cpa.3160300502. URL `https://onlinelibrary.wiley.com/doi/abs/10.1002/cpa.3160300502`.

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

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

[31] Thomas Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ArXiv*, abs/1609.02907, 2017.

[32] John M. Lee. *Riemannian Manifolds: An Introduction to Curvature*. 1997.

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

[34] 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.

[35] Aaron Lou, Isay Katsman, Qingxuan Jiang, Serge Belongie, Ser-Nam Lim, and Christopher De Sa. Differentiating through the fréchet mean. In *International Conference on Machine Learning*, 2020.

[36] 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, pages 17548–17558, 2020.

[37] Yiping Lu, Aoxiao Zhong, Quanzheng Li, and Bin Dong. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. *ArXiv*, abs/1710.10121, 2017.

[38] Emile Mathieu and Maximilian Nickel. Riemannian continuous normalizing flows. In *Advances in Neural Information Processing Systems*, volume 33, pages 2503–2515, 2020.

[39] M. Müller, T. Röder, M. Clausen, B. Eberhardt, B. Krüger, and A. Weber. Documentation mocap database hdm05. Technical Report CG-2007-2, Universität Bonn, June 2007.

[40] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

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

[42] Xavier Pennec, Pierre Fillard, and Nicholas Ayache. A riemannian framework for tensor computing. *International Journal of Computer Vision*, 66:41–66, 2005.

[43] 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*, pages 8083–8092, 2020.

[44] Noam Rozen, Aditya Grover, Maximilian Nickel, and Yaron Lipman. Moser flow: Divergence-based generative modeling on manifolds. In *Neural Information Processing Systems*, 2021.

[45] Lars Ruthotto and Eldad Haber. Deep neural networks motivated by partial differential equations. *Journal of Mathematical Imaging and Vision*, 62:352–364, 2018.

[46] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data articles. *AI Magazine*, 29:93–106, 09 2008. doi: 10.1609/aimag.v29i3.2157.

[47] 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.

[48] Amir Shahroudy, Jun Liu, Tian-Tsong Ng, and Gang Wang. Ntu rgb+d: A large scale dataset for 3d human activity analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1010–1019, 2016.

[49] Ryohei Shimizu, Yusuke Mukuta, and Tatsuya Harada. Hyperbolic neural networks++, 2020.

[50] Sho Sonoda, Isao Ishikawa, and Masahiro Ikeda. Fully-connected network on noncompact symmetric space and ridgelet transform based on helgason-fourier analysis. *ArXiv*, abs/2203.01631, 2022.

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

[52] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio', and Yoshua Bengio. Graph attention networks. *ArXiv*, abs/1710.10903, 2018.

[53] W. Weinan. A proposal on machine learning via dynamical systems. *Communications in Mathematics and Statistics*, 5(1):1–11, March 2017. ISSN 2194-6701. doi: 10.1007/s40304-017-0103-z.

[54] Felix Wu, Tianyi Zhang, Amauri H. de Souza, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. Simplifying graph convolutional networks. *ArXiv*, abs/1902.07153, 2019.

[55] Tao Yu and Christopher M De Sa. Hyla: Hyperbolic laplacian features for graph learning. *ArXiv*, abs/2202.06854, 2022.

Appendix

A Riemannian Geometry: Relevant Reference Material

Here we give some relevant reference material that provides the reader with the fundamental operations used for the Poincaré ball model of hyperbolic space, as well as the two Riemannian SPD manifold structures employed.

A.1 The Poincaré Ball Model

Hyperbolic space can be represented via several isometric models. We use the Poincaré ball model, which is defined by the set

$$\left\{x \in \mathbb{R}^n \mid \|x\|_2^2 < -\frac{1}{K}\right\}, \tag{9}$$

where $K < 0$ is the space's constant negative curvature together with the metric given in the table below. We give a summary of hyperbolic operations in Table 4.

Manifold	**Euclidean** $\mathbb{R}^n$	**Poincaré Ball** $\mathbb{H}^n$
Dimension, $\dim(\mathcal{M})$	n	n
Metric g_x,	$g^{\mathbb{E}}$	$(\lambda_x^K) g^{\mathbb{E}}$, where $g^{\mathbb{E}} = I$
Tangent Space, $T_x\mathcal{M}$	$\mathbb{R}^n$	$\mathbb{R}^n$
Projection, $\text{proj}_{T_x\mathcal{M}}(v)$	v	v
Exp Map, $\exp_x(v)$	$x + v$	$x \oplus_K \left(\tanh\left(\sqrt{\lvert K\rvert}\frac{\lambda_x^K \|v\|_2}{2}\right)\frac{v}{\sqrt{\lvert K\rvert}\|v\|_2}\right)$
Geodesic Distance, $d(x, y)$	$\|y - x\|_2$	$\frac{1}{\sqrt{\lvert K\rvert}}\cosh^{-1}\left(1 - \frac{2K\|x-y\|_2^2}{(1+K\|x\|_2^2)(1+K\|y\|_2^2)}\right)$

Table 4: Summary of Poincaré ball operations. We provide equivalent operations on Euclidean space for reference. $\oplus_K$ denotes Möbius addition [51], and $\lambda_x^K = \frac{2}{1+K\|x\|_2^2}$, a conformal factor.

A.2 The SPD Manifold

We provide a summary of operations on the manifold of SPD matrices, in Table 5. For the SPD manifold, we illustrate the differences between the affine-invariant and log-Euclidean metrics. $\exp$ and $\log$ denote the matrix exponential and logarithm, respectively.

Manifold	**Euclidean** $\mathbb{R}^n$	**SPD**(n) Affine-Invariant	**SPD**(n) Log-Euclidean
Dimension, $\dim(\mathcal{M})$	n	$\frac{n(n+1)}{2}$	$\frac{n(n+1)}{2}$
Metric g_x,	$g^{\mathbb{E}}$	$\text{tr}(X^{-1}UX^{-1}V)$	$\text{tr}((D\log_X(U))^T D\log_X(V))$
Tangent Space, $T_x\mathcal{M}$	$\mathbb{R}^n$	$\{V \mid V = V^T\}$	$\{V \mid V = V^T\}$
Projection, $\text{proj}_{T_x\mathcal{M}}(v)$	v	$\frac{V+V^T}{2}$	$\frac{V+V^T}{2}$
Exp Map, $\exp_x(v)$	$x + v$	$X\exp(X^{-1}V)$	$\exp(\log(X) + V)$
Geodesic Distance, $d(x, y)$	$\|y - x\|_2$	$\|\log(X^{-1}Y)\|_F$	$\|\log(Y) - \log(X)\|_F$

Table 5: Summary of SPD operations. We provide equivalent operations on Euclidean space for reference. We use both the affine-invariant and log-Euclidean metrics.

B 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:

$$\ell_i(x) := \text{proj}_{T_x\mathcal{M}}(n_i(x)). \tag{10}$$

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 4.1). In this section, we give a natural embedded vector field design for hyperbolic space, a more geometric feature map-induced 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.

B.1 Vector Field Design for Hyperbolic Space

For the embedded hyperbolic vector field design, we apply the general design construction referenced 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

$$\ell_i(x) = \text{proj}_{T_x\mathbb{H}^n}(n_i(x)) \tag{11}$$

to obtain our neural network-parameterized vector fields. Observe that this vector field design is efficient and expressive, since $T_x\mathbb{H}^n \cong \mathbb{R}^n$, but is perhaps too expressive in that the vector field is not constructed around the geodesic geometry of hyperbolic space. For this, we employ the horosphere projection-induced vector field design introduced in Section 4.2 of the main paper. We simply fix a number of horospheres, randomly initialize them, and then further learn hyperparameters specifying a given horosphere.

B.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 and Quaintance [16] 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 [16] that the tangent space at $X \in SPD(n)$ is given by:

$$T_X SPD(n) = S(n) := \{P \mid P = P^T\}. \tag{12}$$

Observe that due to this tangent space structure, instead of utilizing the vector field construction given in Section 4.1 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.

B.2.1 Design 1: Embedded

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 $\text{vec} : \mathbb{R}^{n\times n} \to \mathbb{R}^{n^2}$ 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:

$$\ell_i(X) = \text{proj}_{T_X SPD(n)}(\text{vec}^{-1}(n_i(\text{vec}(X)))) \tag{13}$$

where $\text{proj}_{T_X SPD(n)}$ symmetrizes a matrix in the tangent space of the identity matrix, before transforming it back to the tangent space of X. It is given by:

$$\text{proj}_{T_X SPD(n)}(V) = \frac{V + V^T}{2}. \tag{14}$$

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.

B.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:

$$\ell_i(X) = \iota(n_i(\iota^{-1}(X))). \tag{15}$$

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).

B.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. 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:

$$\ell_i(X) = \iota(v) \tag{16}$$

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

B.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 $Q \in O(n)$[6], a neural network $f_i : \mathbb{R}^n \to \mathbb{R}^n$, and, for $X \in SPD(n)$, be given by:

$$\ell_i(X) = Q\text{diag}(f_i(\text{spec}(X)))Q^T \tag{17}$$

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 [1, 6].

Design 1 is naive, 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.3 Vector Field Design for Spherical Space

For the spherical vector field design, we again apply the general design construction referenced at the start of Appendix B. Similar to $\mathbb{H}^n$, $\mathbb{S}^n$ is an n-dimensional manifold which we treat as embedded in $\mathbb{R}^{n+1}$. Hence we parameterize a neural network $n_i : \mathbb{R}^{n+1} \to \mathbb{R}^{n+1}$ and set

$$\ell_i(x) = \text{proj}_{T_x\mathbb{S}^n}(n_i(x)) \tag{18}$$

to obtain our neural network-parameterized vector fields. As in the hyperbolic case, this vector field design is efficient and expressive, since $T_x\mathbb{S}^n \cong \mathbb{R}^n$.

B.4 Feature Map-induced Vector Fields for General Manifolds

There is no perfect analog of a hyperplane for general manifolds. Hence, there is no immediately natural feature map in the general case. Despite this, we attempt to present a reasonable analog to hyperplane projection that extends to general manifolds. In particular, for a geodesically complete[8] manifold $\mathcal{M}$, consider specifying a pseudo-hyperplane by a point $p \in \mathcal{M}$ and a non-zero vector $v \in T_p\mathcal{M} \setminus \{\mathbf{0}\}$ whose orthogonal complement we exponentiate at the base point p to give the following definition:

$$h_{p,v} = \exp_p(\{w \in T_p\mathcal{M} | w^T v = 0\}) \tag{19}$$

This definition[9] has the benefit of reducing to the usual Euclidean hyperplane definition when the manifold under consideration is $\mathbb{R}^n$. However, this hyperplane definition is not particularly suitable for general manifolds since it assumes geodesic completeness, which may not hold. Here we propose an alternative general definition of a hyperplane that exponentiates the intersection of a local orthogonal complement with a closed ball of radius r, $\bar{B}_r(0) \subset T_p\mathcal{M}$, given below:

$$h_{p,v,r} = \exp_p(\bar{B}_r(\mathbf{0}) \cap \{w \in T_p\mathcal{M} | w^T v = 0\}) \tag{20}$$

Notice that this $h_{p,v,r}$ pseudo-hyperplane is a strict generalization of $h_{p,v}$ that does not require geodesic completeness (since r is finite), and that in the limit as $r \to \infty$ we recover $h_{p,v}$.

A general feature map can then be defined by projecting to such a pseudo-hyperplane:

$$g_{p,v,r}(x) = \min_{y \in h_{p,v,r}} d_{\mathcal{M}}(x, y) \tag{21}$$

where $d_{\mathcal{M}}$ is the induced geodesic distance on $\mathcal{M}$.

We test this general construction for hyperbolic space and compare it with the horosphere projection construction in Appendix C.4. The general construction performs reasonably well, but does not perform as well as the horosphere projection we give in this section. A more natural and performant manifold-dependent map can frequently be obtained by carefully considering the particular structure of the manifold (e.g. the spectral projection we give for $SPD(n)$).

C Experimental Details

Experiments on Hyperbolic Space

C.1 Datasets

We apply our hyperbolic RResNet to node classification and link prediction on four graph datasets with varying δ-hyperbolicity.

Airport ($\delta = 1$). Airport is a dataset consisting of 2236 nodes where nodes represent airports and edges represent airline routes [8]. For node classification, each airport is given a label corresponding

[8] A manifold $\mathcal{M}$ is said to be geodesically complete if any geodesic can be followed indefinitely [32].

[9] This notion was originally introduced in the context of hyperbolic space in Ungar [51].

to the population of the country it is in. Each airport has a 4-dimensional feature vector consisting of geographic information.

Pubmed ($\delta = 3.5$) and CoRA ($\delta = 11$). Pubmed and CoRA are both citation networks consisting of 2708 and 19717 nodes each [46, 47]. In citation networks, each node represents a paper and edges indicate a shared author between papers. Each node has a label consisting of what academic subareas the paper belongs to.

Disease ($\delta = 0$). Disease is a synthetic dataset generated by simulating the SIR disease spreading model [8]. Node labels for classification indicate whether a node was infected or not and node features indicate a particular node's susceptibility to the disease.

C.2 Architectural and Training Details

All of our testing uses the Poincaré ball model [41] to represent hyperbolic space. We use a similar setup to Chami et al. [8] to test RResNet's performance on hyperbolic space. 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 [8].

For our results using a feature map induced vector field, we take the projection onto a fixed number of horospheres. Each horosphere is randomly initialized with ω drawn uniformly from $\mathbb{S}^{n-1}$ and $b \sim \mathcal{N}(0, 1)$. We pass the horocycle projections to a linear layer followed by a Euclidean nonlinearity (typically ReLU [40]). During the training of each network, ω and b are optimized using the same optimizer as the rest of the network. For further details regarding implementation, please see the accompanying Github code.

Horosphere projections are not the only natural feature map one can use, one alternative we experimented with was using parametetrized real eigenfunctions of the hyperbolic Laplacian as feature maps but we were unable to achieve similar performance to horosphere projections (results were significantly worse).

We use 250 horospheres for Disease, Airport, and CoRA and 50 horospheres for Pubmed. Models were trained for 500, 10000, 5000, and 5000 epochs for Disease, Airport, Pubmed, and CoRA, respectively, with the Adam optimizer [30]. All other hyperparameters, such as learning rate and weight decay, were determined using random search.

All experiments were run on a single NVIDIA Quadro RTX A6000 48GB GPU.

C.3 Comparison Between Embedded and Horocycle-induced Vector Field Designs

Dataset	Disease ($\delta = 0$)	Airport ($\delta = 1$)	Pubmed ($\delta = 3.5$)	CoRA ($\delta = 11$)
RResNet Embedded	$75.0_{\pm 5}$	$83.0_{\pm 2.0}$	$\mathbf{73.2}_{\pm 1.0}$	$\mathbf{59.6}_{\pm 1.0}$
RResNet Horo	$\mathbf{76.8}_{\pm 2.0}$	$\mathbf{96.9}_{\pm 0.3}$	$71.4_{\pm 2.2}$	$52.4_{\pm 5.5}$

Table 6: Node classification results for RResNet with two different vector field designs (test F1 score is the metric given).

In order to investigate the effect vector field design has, we look at the performance of RResNet when using the embedded or horosphere projection-induced vector field in Table 8. On more hyperbolic datasets (Disease and Airport), the more geometrically principled design attains higher F1 scores. This effect is reversed on the less hyperbolic datasets (Pubmed and CoRA), indicating that a more geometrically principled vector field only helps when the data geometry is similar to the model geometry, as expected.

C.4 Comparison Between Horocycle-induced and Pseudo-Hyperplane-induced Vector Field Designs

In Table 7 we compare the RResNet construction with vector fields induced by projection to pseudo-hyperplanes (as defined in the main paper in Section 4.2) for hyperbolic space (RResNet Pseudo-Hyperplane) to the horocycle projection-induced vector field RResNet construction (RResNet Horocy-

Dataset	Disease ($\delta = 0$)	Airport ($\delta = 1$)	Pubmed ($\delta = 3.5$)	CoRA ($\delta = 11$)
RResNet Horocycle	$\mathbf{76.8}_{\pm 2.0}$	$\mathbf{96.9}_{\pm 0.3}$	$\mathbf{71.4}_{\pm 2.2}$	$\mathbf{52.4}_{\pm 5.5}$
RResNet Pseudo-Hyperplane	$\mathbf{77.2}_{\pm 0.4}$	$90.3_{\pm 4.5}$	$66.7_{\pm 5.0}$	$41.4_{\pm 5.7}$

Table 7: Node classification results for RResNet with two different vector field designs (test F1 score is the metric given).

cle). Note that RResNet Pseudo-Hyperplane performs worse for most tasks, although the construction is more general (as mentioned in the main paper).

C.5 Ablation Study

Nonlinearity Ablation

Dataset	Disease ($\delta = 0$)	Airport ($\delta = 1$)	Pubmed ($\delta = 3.5$)	CoRA ($\delta = 11$)
RResNet Horo w/o Nonlinearity	$71.9_{\pm 9.2}$	$\mathbf{96.9}_{\pm 3.0}$	$\mathbf{71.2}_{\pm 1.1}$	$49.6_{\pm 2.0}$
RResNet Horo	$\mathbf{76.8}_{\pm 2.0}$	$\mathbf{96.9}_{\pm 0.3}$	$\mathbf{71.4}_{\pm 2.2}$	$\mathbf{52.4}_{\pm 5.5}$

Table 8: Node classification results for RResNet with and without a nonlinearity between layers (test F1 score is the metric given).

To study the expressiveness of the horocycle induced vector field design, we ablate the nonlinearity in the vector field. With the nonlinearity, the F1 score either increases or remains the same across all datasets, which the advantage being most pronounced for Disease.

Geometry Ablation

Dataset	Disease ($\delta = 0$)
RResNet Embedded (Euclidean)	$67.3_{\pm 21.0}$
RResNet Embedded (Hyperbolic)	$75.0_{\pm 5.0}$
RResNet Feature Map (Euclidean)	$73.1_{\pm 3.4}$
RResNet Feature Map (Hyperbolic)	$\mathbf{76.8}_{\pm 2.0}$

Table 9: Node classification results of RResNet with different vector field designs and model geometry (test F1 score is the metric given). When swapping geometry for a specific model, all hyperparameters are kept the same, which we are able to do easily with our architecture.

We look at the performance of varying RResNets on the most hyperbolic dataset to identify the effect model geometry has in Table 9. As expected, using hyperbolic space yields higher F1 scores with lower standard deviations. In particular, the high standard deviation of 21.0 for "RResNet Embedded (Euclidean)" indicates that it fails to properly learn in a number of trials.

Residual Connection Ablation

It is reasonable to try other residual connection implementations outside of our natural geometric Riemannian exp-map based implementation. In particular, one may try to implement a Riemannian residual neural networks directly via a gyrovector [51] addition. We give the results in Table 10 and show that not only is this method less desirable geometrically, but also gives worse results on our chosen benchmarks. The Euclidean model is given as a baseline and the Riemannian ResNet here is a reference. All models are implemented with a comparable number of parameters and are two layer residual neural networks.

Dataset	Airport ($\delta = 1$)
Euclidean	$69.4_{\pm 1.8}$
Gyrovector	$60.8_{\pm 0.9}$
RResNet Horo	$\mathbf{75.9}_{\pm 2.5}$

Table 10: Node classification results for RResNet with three different residual connection designs (test F1 score is the metric given).

Experiments on the SPD Manifold

C.6 Datasets

We apply our SPD architecture on four different video recognition tasks. For all tasks, we generate covariance or correlation matrices sampled from each video's frames. Given frames $t \in \{1, \ldots, T\}$ and their corresponding feature vectors $x_t \in \mathbb{R}^n$, we generate a $n \times n$ covariance matrix by sampling the frames: $\mathrm{X} = \frac{1}{T-1} \sum_{t=1}^{T} (x_t - \mu)(x_t - \mu)^T$. Optionally, we can divide the matrices by the standard deviations to instead generate correlation matrices. For certain tasks, we find that these have better conditioning.

While covariance and correlation matrices are positive semi-definite, they are not necessarily SPD. In fact, they are only SPD if the set of sampled vectors, $\{x_1, \ldots, x_T\}$, consists of n linearly-independent vectors. If the sampled vectors x_t, x_{t+1}, are similar, which is the case for neighboring frames of a video, the matrices may be close to singular. This phenomenon poses issues in downstream tasks such as taking a matrix logarithm, which can create numerical instability. For all tasks, we preprocess our data by removing covariance matrices which fail a Cholesky decomposition.

AFEW. AFEW [13] is an emotion recognition dataset consisting of 1,345 videos and 7 classes. As done in Brooks et al. [5], Huang and Gool [26], we use covariance matrices created from 20×20 video frames, flattened into 400-dimensional x_t vectors.

FPHA. The First-Person Hand Action Benchmark (FPHA) [18] consists of 1,175 videos of humans performing 45 different tasks. The dataset includes the (x, y, z) coordinates of 21 joint locations from a human hand. Following the approach of Hussein et al. [27], for each frame, we flatten the coordinates into a 63-dimensional vector x_t. We then take the correlation matrices. We use subjects 1-3 for training and 4-6 for validation.

NTU RGB+D. NTU RGB+D [48] is an action recognition dataset which includes the 3D locations of 25 body joints. NTU RGB+D is a large scale dataset with 56,880 videos and 60 tasks. For our x_t vectors, we use the flattened versions of 3D joint coordinates as feature vectors. Our matrices have dimension 75.

HDM05. Mocap Database HDM05 [39] is another action recognition dataset which includes 3D locations of 31 joints. Following the task designed in Huang and Gool [26], the goal is to classify each video clip into one of 117 action classes. We use the covariance matrices provided in Brooks et al. [5].

C.7 Architectural Details

Given a dataset of covariance matrices, our goal is to classify a matrix into one of several classes. To illustrate, we give our architecture for the AFEW task as an example. Because of how costly it would be to parameterize vector fields at this dimension, we use a BiMap layer [26], $\mathrm{BiMap}_{d_{i+1}}^{d_i} : SPD(d_i) \to SPD(d_{i+1})$ as a base point remapping from 400×400 matrices to 50×50 matrices. We use vector field design 4 from Appendix B. In the context of this problem, we have:

$$\ell_1(X) = Q\mathrm{diag}(f_1(\mathrm{spec}(X)))Q^T \tag{22}$$

where $f_1 : \mathbb{R}^{50} \to \mathbb{R}^{50}$, spec : $SPD(50) \to \mathbb{R}^{50}$, $P \in O(50)$ (spec is defined above in Appendix B). In practice, we experiment with a variety of f_1 designs, such as sequences of linear layers or 1D convolutions. Note the vector field is a map $\ell_1 : SPD(50) \to T\,SPD(50)$. We express our forward pass as

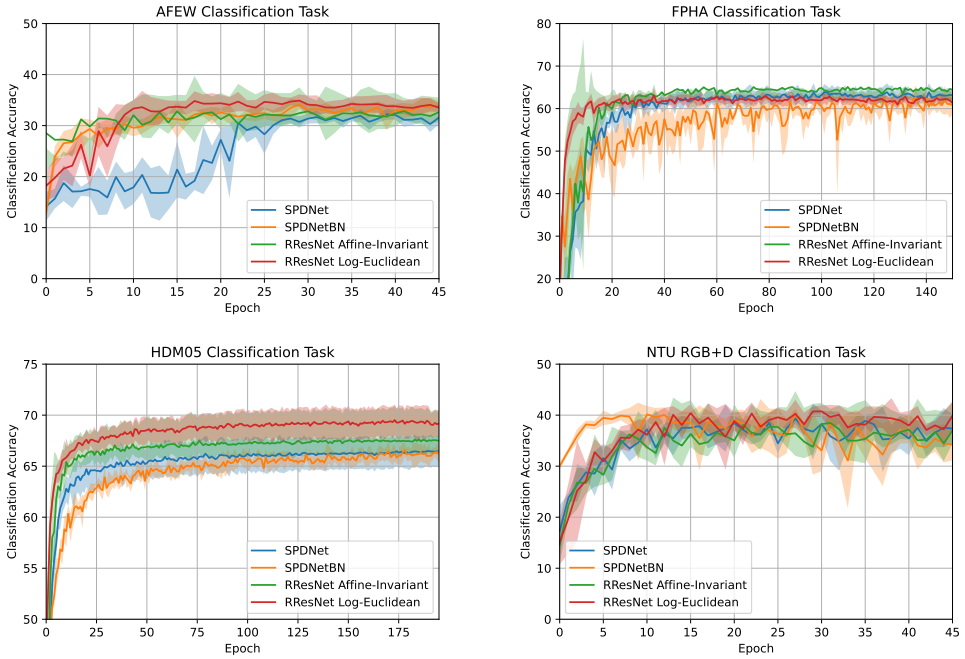

Figure 5: Validation accuracies for our RResNet compared to the SPDNet [26] and SPDNetBN [5] baselines. For each model, results are averaged over five trials. Error bars denote one standard deviation away from the mean accuracy. We observe that our model converges faster and achieves higher accuracies than SPDNet and SPDNetBN.

$$g(x) = \exp_{\mathrm{BiMap}_{50}^{400}(x)}(\ell_1(\mathrm{BiMap}_{50}^{400}(x))) \tag{23}$$

which is a map $g : SPD(400) \to SPD(50)$. Our exp map depends on the Riemannian metric we choose on the manifold. Thereafter we apply a logarithm to the eigenvalues of the 50×50 matrices (this helps linearize features [5]). 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 [19] to train the model.

C.8 Results

We compare our RResNet design above (Appendix C.7) to SPDNet [5, 26], a network architecture for SPD matrix learning. All models have a comparable number of parameters. To replicate the results of Brooks et al. [5], we use a learning rate of $5 \cdot 10^{-2}$ for the baseline. We find that any higher learning rate causes training instability. However, we observe that our model remains stable with a learning rate of $1 \cdot 10^{-1}$. Our model has faster convergence and achieves a higher accuracy than SPDNet and SPDNetBN (see Table 2 in the main paper and Figure 5 above). Moreover, our model's ability to switch out geometries (as given by the log-Euclidean and affine-invariant metrics) gives the ability to outperform prior work on all tasks.

C.9 Comparison Between Vector Field Designs

For the SPD manifold, we illustrate differences between our four different vector field designs outlined in Appendix B on the AFEW task. Results are given in Table 11. Note that the chosen spectral map-induced vector field is very efficient in terms of parameter count and performs best in terms of accuracy.

C.10 Ablation Study

Nonlinearity Ablation

	AFEW[13]	Number of Parameters
Naive	$34.90_{\pm 0.74}$	6,290,007
Structured	$34.76_{\pm 1.07}$	1,664,407
Parsimonious	$34.82_{\pm 1.82}$	38,782
Spectral Map	$\mathbf{36.38_{\pm 1.29}}$	45,057

Table 11: We compare the accuracy of our four vector field designs for the SPD manifold. We see that the spectral map provides the best balance of accuracy and parameter efficiency.

We ablate the nonlinearity in the spectral map design in Table 12, and find that the nonlinearity slightly improves performance. For AFEW, we use a one layer vector field, which is why the reported accuracies are the same.

	AFEW	FPHA	NTU RGB+D	HDM05
Aff-Inv w/o Nonlinearity	$35.17_{\pm 1.78}$	$65.03_{\pm 2.22}$	$41.27_{\pm 0.22}$	$65.92_{\pm 1.27}$
Aff-Inv	$35.17_{\pm 1.78}$	$\mathbf{66.53_{\pm 1.64}}$	$41.00_{\pm 0.50}$	$67.91_{\pm 0.66}$
Log-Euc w/o Nonlinearity	$36.38_{\pm 1.29}$	$65.25_{\pm 2.14}$	$42.87_{\pm 0.83}$	$68.73_{\pm 1.75}$
Log-Euc	$\mathbf{36.38_{\pm 1.29}}$	$64.58_{\pm 0.98}$	$\mathbf{42.99_{\pm 0.23}}$	$\mathbf{69.80_{\pm 1.51}}$

Table 12: We show that classification accuracy either improves or remains the same with the nonlinearity. For AFEW, we use one layer in the vector field, which is why the reported accuracies are the same.

Geometry Ablation

We study the geometry of the SPD manifold by comparing our Riemannian ResNet to a Euclidean ResNet. For the Euclidean network, we treat each matrix as a Euclidean vector by flattening it into a length $n \times n$ vector. We then pass it through a Euclidean ResNet. Our results in Table 13 show that the Riemannian ResNets (Aff-Inv and Log-Euc) perform significantly better across all datasets.

	AFEW	FPHA	NTU RGB+D	HDM05
Euclidean	$30.08_{\pm 1.36}$	$30.72_{\pm 1.03}$	$34.63_{\pm 3.10}$	$0.80_{\pm 0.10}$
Aff-Inv	$35.17_{\pm 1.78}$	$\mathbf{66.53_{\pm 1.64}}$	$41.00_{\pm 0.50}$	$67.91_{\pm 1.27}$
Log-Euc	$\mathbf{36.38_{\pm 0.24}}$	$64.58_{\pm 0.98}$	$\mathbf{42.99_{\pm 0.23}}$	$\mathbf{69.80_{\pm 1.51}}$

Table 13: We show that the Euclidean ResNet performs worse across all datasets, and fails for HDM05.

Residual Connection Ablation

Similar to the gyrocalculus used in Ganea et al. [17], López et al. [34] have extended gyrovector operations to the manifold of SPD matrices. In particular, the authors define Möbius addition as $X \oplus Y = \sqrt{X}Y\sqrt{X}$ for SPD matrices X, Y. It is reasonable to ask how this purely algebraic, non-geometric construct performs when used to implement a residual connection. With this choice of addition, the residual connection for a ResNet specific to the SPD manifold would have the form $\ell_i(X) + X = \sqrt{\ell_i(X)}X\sqrt{\ell_i(X)}$. In Table 14, we show that this choice of addition struggles to reach the accuracy of our Riemannian ResNet design.

Experiments on Spherical Space

C.11 Dataset

We wish to explore the generality of our method: in particular, our ability to vary geometry without constructing entirely new operations for each manifold. We repeat one of the experiments tested on our hyperbolic RResNet, swapping out the hyperbolic manifold for the spherical manifold.

	AFEW	FPHA	NTU RGB+D	HDM05
Gyrovector	$23.23_{\pm 0.98}$	$61.33_{\pm 4.74}$	$40.77_{\pm 3.10}$	$5.69_{\pm 2.15}$
Aff-Inv	$35.17_{\pm 1.78}$	$\mathbf{66.53_{\pm 1.64}}$	$41.00_{\pm 0.50}$	$67.91_{\pm 1.27}$
Log-Euc	$\mathbf{36.38_{\pm 1.29}}$	$64.58_{\pm 0.98}$	$\mathbf{42.99_{\pm 0.23}}$	$\mathbf{69.80_{\pm 1.51}}$

Table 14: We show that the Riemannian ResNet model with Möbius addition struggles to reach the classification accuracies of our exponential map design. The difference is most pronounced on HDM05, where the gyrovector model struggles to learn meaningful representations.

CoRA. This dataset is described above in Appendix C.1. With $\delta = 11$, CoRA is the least hyperbolic of the datasets tested with our hyperbolic RResNet. As such, we wanted to try swapping the RResNet geometry to better match the data geometry.

C.12 Architectural Details

The design of our spherical RResNet is identical to that of our hyperbolic RResNet (described in Appendix C.2), aside from switching the geometric representation from hyperbolic to spherical. As before, we first have a linear layer to move from the input dimension to a lower dimension. Then we use our RResNet as an encoder. Here we only test link prediction, so we use a Fermi-Dirac decoder.

We train for 2000 epochs using the Adam optimizer [30], and we again found all hyperparameters via random search.

C.13 Results

We give results for link prediction on CoRA, displayed in Table 15. Mean and standard deviation across 5 separate trials are reported.

	Dataset **Hyperbolicity**	CoRA $\delta = 11$
Hyp	RResNet	$88.9_{\pm 0.2}$
	RResNet Graph	$87.6_{\pm 0.9}$
Sphere	RResNet	$90.7_{\pm 1.0}$
	RResNet Graph	$\mathbf{91.7}_{\pm 0.4}$

Table 15: Test accuracy of various models, in terms of ROC AUC.

We find that even the most basic spherical RResNet design, which does not use a feature map, outperforms both hyperbolic RResNets. This indicates that our model improves when endowed with geometry more suitable for given data. Additionally, our model's flexibility allows us to easily obtain such results without altering the architecture.

D Theoretical Results

In this section we give a variety of theoretical results that demonstrate the principled nature of our Riemannian ResNet construction.

D.1 Reduction to Standard ResNet in Euclidean Case

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

Proposition 1. *When $\mathcal{M}^{(i)} \cong \mathbb{R}^{d_i}$, our RResNet with the embedded vector field design is a standard residual network.*

Proof. Note that the embedded vector fields take the form:

$$\ell_i(x) = \text{proj}_{T_x\mathbb{R}^n}(n_i(x)) = n_i(x) \tag{24}$$

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

$$f(x) = x^{(m)} \tag{25}$$

$$x^{(0)} = x \tag{26}$$

$$x^{(i)} = \exp_{h_i(x^{(i-1)})}(\ell_i(h_i(x^{(i-1)}))) \tag{27}$$

$$= h_i(x^{(i-1)}) + \ell_i(h_i(x^{(i-1)})) \tag{28}$$

$$= h_i(x^{(i-1)}) + n_i(h_i(x^{(i-1)})) \tag{29}$$

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

$$x^{(i)} = x^{(i-1)} + n_i(x^{(i-1)}) \; \forall i \in [m] \tag{30}$$

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

D.2 Hyperbolic Neural Networks (HNNs) [17] Learn via a Hyperbolic Bias

We make note of the fact that although the gyrovector generalization of Euclidean networks offered by Ganea et al. [17] is algebraic and generalizable to many manifolds such as hyperbolic space and the manifold of SPD matrices [34], the linear layer of the construction is Euclidean, except for the hyperbolic bias addition. We illustrate this in what follows.

Proposition 2. *For $x \in \mathbb{H}^n$ and hyperbolic matrix-vector multiplication [17] defined by*

$$M^{\otimes}(x) = \tanh\left(\frac{||Mx||}{||x||}\tanh^{-1}(||x||)\right)\frac{Mx}{||Mx||} \tag{31}$$

where $M : \mathbb{R}^n \to \mathbb{R}^n$ is a linear map, we have

$$M_2^{\otimes}(M_1^{\otimes}(x)) = \tanh\left(\frac{||M_2||||M_1x||}{||x||}\tanh^{-1}(||x||)\right)\frac{M_2M_1x}{||M_2||||M_1x||} = (M_2M_1)^{\otimes}(x) \tag{32}$$

Proof. For two linear maps of the same size M_1, M_2 we have:

$$M_2^{\otimes}(M_1^{\otimes}(x)) = \tanh\left(\frac{||M_2M_1^{\otimes}(x)||}{||M_1^{\otimes}(x)||}\tanh^{-1}(||M_1^{\otimes}(x)||)\right)\frac{M_2M_1^{\otimes}(x)}{||M_2M_1^{\otimes}(x)||} \tag{33}$$

$$= \tanh\left(\frac{||M_2M_1^{\otimes}(x)||}{\tanh\left(\frac{||M_1x||}{||x||}\tanh^{-1}(||x||)\right)}\left(\frac{||M_1x||}{||x||}\tanh^{-1}(||x||)\right)\right)\frac{M_2M_1^{\otimes}(x)}{||M_2M_1^{\otimes}(x)||} \tag{34}$$

$$= \tanh\left(\frac{||M_2||||M_1x||}{||x||}\tanh^{-1}(||x||)\right)\frac{M_2M_1x}{||M_2||||M_1x||} = (M_2M_1)^{\otimes}(x) \tag{35}$$

□

We see that we have cancellation that de facto reduces the learning of two hyperbolic linear layers with no hyperbolic bias to the learning of a single hyperbolic layer. Inductively, this precise argument applies to any number of layers. This reduction is characteristic to what one sees in the case of Euclidean networks, and more importantly, from the above equation we see that learning hyperbolic linear layers de facto reduces to learning Euclidean linear maps (M_1 and M_2 above) that are placed in between an initial Riemannian $\log$ map (taken at the origin) and a trailing Riemannian $\exp$ map (taken at the origin).

Thus, the main non-Euclidean, hyperbolic construct in Ganea et al. [17] is the hyperbolic bias, introduced in Section 3.2 of Ganea et al. [17]. Our method is distinctly different in that even simple residual linear layers make use of geodesic information; hence, learning does not reduce to the Euclidean case.

E Comparison with Other Constructions

Here we elaborate on how our method compares with other constructions, elucidating a claim made in the main paper. We note that compared to other methods, our construction is fully general (in the sense that it extends to all Riemannian manifolds) and better conforms with geometry. For example, general methods like HNN [17], HGCN [8], and SPDNetBN [5] use the fact that hyperbolic space and the SPD manifold are spaces with everywhere non-negative curvature, meaning that geodesics are unique. As such, core building blocks of these models globally project to a Euclidean space via a map known as the Riemannian $\log$ map, which can be thought of as an inverse to the exponential map. This global projection relies on the choice of a particular base point (equivalent to the base point in the $\exp$ map definition), which is arbitrarily selected. Once this projection has taken place, prior methods usually simply perform a Euclidean operation, and project back to the manifold via the usual Riemannian $\exp$. This system does not generalize to manifolds which are not globally diffeomorphic to Euclidean space (note this is quite restrictive and different from being locally diffeomorphic to Euclidean space), and furthermore, the reliance on fundamentally Euclidean operations and arbitrary base point destroys the geodesic geometry: the log map can be thought of as linearizing manifold geometry with respect to a certain base point—the further away points are from this base point, the more distorted the projected geometry.

More specialized methods like Chen et al. [10], Shimizu et al. [49] instead work with algebraic operations that exist only for hyperbolic space. These do not generalize to arbitrary manifolds, which limits potential applications. By comparison, our method simultaneously generalizes to arbitrary manifolds and directly conforms with underlying geometry.