LEARNING COMPLEX GEOMETRIC STRUCTURES FROM DATA WITH DEEP RIEMANNIAN MANIFOLDS

Anonymous authors
Paper under double-blind review

ABSTRACT

We present Deep Riemannian Manifolds, a new class of neural network parameterized Riemannian manifolds that can represent and learn complex geometric structures. To do this, we first construct a neural network which outputs symmetric positive definite matrices and show that the induced metric can universally approximate all geometries. We then develop differentiable solvers for core manifold operations like the Riemannian exponential and logarithmic map, allowing us to train the manifold parameters in an end-to-end machine learning system. We apply our method to learn 1) low-distortion manifold graph embeddings and 2) the underlying manifold of geodesic data. In addition to improving upon the baselines, our ability to directly optimize the Riemannian manifold brings to light new perspectives with which to view these tasks.

1 INTRODUCTION

Many complex structures in data are naturally captured by Riemannian geometry. For example, weighted graphs (Linial et al., 1995), latent spaces of generative models (Chen et al., 2018a), and natural science phenomena (Boyda et al., 2021) all naturally lie on non-Euclidean manifolds. Recent efforts in deep learning have focused on exploiting this geometric structure to build better models in the emerging field of Geometric Deep Learning (Bronstein et al., 2021).

Most successes in this area have focused on generalizing classic techniques to some fixed manifold like the Poincaré n-ball (Nickel & Kiela, 2017), hyperspheres (Cohen et al., 2018), or smooth surfaces (Masci et al., 2015). However, these methods are only capable of working with Riemannian manifolds that are selected *a priori*, meaning that the geometry is assumed–not recovered–from the data. Worse still, approaches which recover data geometry learn only with topological information. Since manifolds can have the same topology but vastly different geometries (e.g. hyperbolic and Euclidean space), this type of methodology is insufficient for capturing the true geometry.

In this paper, we overcome these limitations and introduce the first method for learning Riemannian manifolds from geometric information. We start by introducing Deep Riemannian Manifolds, the first neural-network parameterized Riemannian manifolds that are provably universal. Then, to work with these manifolds, we develop differentiable Riemannian manifold operations using neural ODEs (Chen et al., 2018b). In particular, for the manifold logarithmic map, we introduce Neural BVPs, a new type of neural differential equation solver. Finally, we apply our method to two existing tasks in the literature: learning graph embeddings and fitting geodesics. Our method improves over prior work and provides further insight into these problems. In particular, for graph embeddings we prove that our method is capable of solving the metric space embedding problem, and for the geodesic fitting tasks we show that our novel ability to train with geometry is critical for recovering the underlying manifold when data is more complicated.

2 DEEP RIEMANNIAN MANIFOLDS

In this section, we present Deep Riemannian Manifolds, our neural network parameterized Riemannian manifolds. Then, we derive several properties of Deep Riemannian Manifolds; in particular, they are able to universally approximate any geometric structure. Finally, we compare with other metrics on both theoretical and empirical grounds. Here, we find that our method use parameters efficiently, give stronger correctness guarantees, and exhibit traits that are amenable for geodesic-based training.

2.1 CONSTRUCTION

We construct our Deep Riemannian Manifolds as $(\mathbb{R}^n, g_{\text{nn}})$, where g_{nn} is a Riemannian metric parameterized by a neural network. We use $\mathbb{R}^n$ as our underlying manifold for computational purposes and show that this is sufficient for representing complex geometries. The key to good representation is how we construct g_{nn}.

Recall that a Riemannian metric is a smoothly varying inner product on the tangent spaces $T_x\mathcal{M}$. When $\mathcal{M} = \mathbb{R}^n$, it is instead a smoothly varying inner product on $\mathbb{R}^n$ because $T_x\mathbb{R}^n \cong \mathbb{R}^n$. Since inner products on $\mathbb{R}^n$ are given by $x^\top Ax$ where A is a symmetric positive definite (SPD) matrix, a Riemannian metric on $\mathbb{R}^n$ is a smooth function from $\mathbb{R}^n \to \mathcal{S}_n^+$, the space of SPD matrices.

With this natural equivalence, to design a Riemannian metric we simply need to construct a smooth map $g : \mathbb{R}^n \to \mathcal{S}_n^+$ using a neural network. The difficulty lies in representing outputs on the Lie Group $\mathcal{S}_n^+$ (Gallier & Quaintance, 2020). Naive attempts for constraining the neural network output (e.g. projection) are often parameter inefficient, computationally challenging, or difficult to differentiate (Goulart et al., 2019). Instead, we follow more recent geometry-based representations (Lezcano-Casado, 2019) and directly work with the Lie Group structure.

In particular, $\mathcal{S}_n^+$ is diffeomorphic to its Lie algebra $\mathcal{S}_n$, the set of symmetric $\mathbb{R}^{n\times n}$ matrices. To see this, we can decompose an SPD matrix into $PDP^\top$ for positive diagonal D and orthonormal P. With this structure, $\log(PDP^\top) = P\log(D)P^\top$, so the matrix exponential is the diffeomorphism between $\mathcal{S}_n$ and $\mathcal{S}_n^+$. Compared with the nonlinear structure of $\mathcal{S}_n^+$, $\mathcal{S}_n$ is a vector space of dimension $\frac{n(n+1)}{2}$ (determined by the upper triangular portion of the matrix). This means that, with a proper change of variables, a standard neural network can directly output values in $\mathcal{S}_n$.

Let $\text{sym} : \mathbb{R}^{\frac{n(n+1)}{2}} \to \mathcal{S}_n$ be the operator which takes a vector and outputs the matrix in $\mathcal{S}_n$ which has an upper triangular part equal to the vector (under some constant reordering). Using the exponential map, we can construct a Riemannian metric for a neural network $f_{\text{nn}} : \mathbb{R}^n \to \mathbb{R}^{\frac{n(n+1)}{2}}$ given by

$$g = \exp \circ \text{sym} \circ f_{\text{nn}} \tag{1}$$

2.2 THEORETICAL PROPERTIES

We verify that our Deep Riemannian Manifolds satisfies several desirable properties.

Prop 2.1 (Smoothness of Deep Metric)**.** *If f_{nn} has a smooth activation function such as* tanh*, then the metric given in equation 1 is smooth.*

Next, we show that Deep Riemannian Manifolds have a nontrivial representational capacity. In particular, they approximate all Riemannian manifolds with $\mathbb{R}^n$ as the underlying manifold.

Prop 2.2 (Universal Approximation of Riemannian Metrics on $\mathbb{R}^n$)**.** *On a compact set D of $\mathbb{R}^n$, for any metric h and $\epsilon > 0$ there exists a neural network metric g of the form in Equation 1 s.t. $\sup_{x\in D} \|g_x - h_x\| < \epsilon$, where $\|\cdot\|$ is the standard ℓ^2 norm.*

As a result, they are also able to universally approximate all Riemannian manifolds.

Theorem 2.3 (Universal Approximation of all Riemannian Manifolds)**.** *For a compact Riemannian manifold $(\mathcal{M}, g)$ and any $\epsilon > 0$, there exists a Deep Riemannian Manifold $(\mathbb{R}^n, g_{\text{nn}})$ with $n = 2\dim\mathcal{M}$ and an embedding $f : \mathcal{M} \to \mathbb{R}^n$ s.t. $|d_g(x, y) - d_{g_{\text{nn}}}(f(x), f(y))| < \epsilon$ for all $x, y \in \mathcal{M}$.*

The results are proven in Appendix A.

2.3 COMPARISON WITH OTHER METRICS

We compare against two other neural network based Riemannian metrics. The first, which we call the "$A^\top A$" metric, takes a neural network $f_{\text{nn}} : \mathbb{R}^n \to \mathbb{R}^{\frac{n(n+1)}{2}}$ and the upper triangular reshape ut to construct the metric $g(x) = A^\top A$ where $A = (\text{ut} \circ f_{\text{nn}})(x)$. The second, which we call the pullback metric, uses the differential of a neural network $f_{\text{nn}} : \mathbb{R}^n \to \mathbb{R}^N$ to construct a metric $g(x) = (D_x f_{\text{nn}})^\top (D_x f_{\text{nn}})$.

Both schemes have previously appeared when representing Riemannian metrics. The $A^\top A$ construction appears in Riemannian motion planning in robotics (Rana et al., 2019). The pullback metric is a

standard construction in Riemannian geometry (Do Carmo, 2016) and has been used when designing Riemannian metrics for generative models (we cover these in more detail in Section 5.2.1).

2.3.1 REPRESENTATION

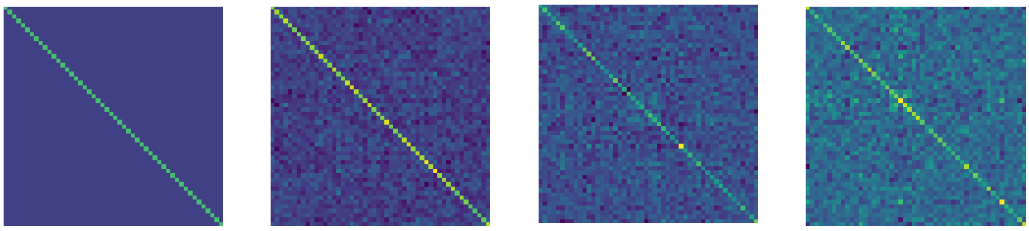
We first compare the representation ability of the metrics. In particular, we are concerned with 1) mathematical correctness and 2) the number of output dimensions required for good representation. The first concern is important to avoid singularities (even measure 0 sets may be traversed by our differential equation solver), and the second directly controls the size of our neural networks.

Correctness. Our metric correctly defines an SPD matrix for each input point x. The $A^\top A$ metric is not necessarily correct (e.g. if a column of the output A is 0) and adding a small positive noise to the diagonal restricts representation. The pullback metric is faithful if and only if f_{nn} is immersion, but this condition is difficult to maintain in a general f_{nn}. Importantly, the $A^\top A$ and pullback metrics can not be considered valid metrics since they may not be correct.

Dimensions. Both our metric and the $A^\top A$ metric use a fixed $\frac{n(n+1)}{2}$ dimensions. However, the pullback metric requires approximately $n^2 + 5n + 3$ dimensions by the Nash Embedding Theorem. Furthermore, it is unclear if the pullback metric provides universal approximation guarantees since this method differentiates through a neural network.

2.3.2 STABLE GEODESICS

Due to the natural limitations of solvers, incorrectly evaluated geodesics greatly hamper training. Therefore, we also consider the stability of the geodesics at initialization time, since incorrect initial evaluations would stifle any attempt to learn. Previous work has shown that random Riemannian metrics almost surely produce geodesics which are ill-conditioned (LaGatta & Wehr, 2014). Following this analysis, we evaluate the metrics at initialization time for a random data point (visualized in Figure 1) and analyze the noise in the matrices. We find that our proposed metric is the most noise-free, so we hypothesize that it would produce the most stable geodesics in training. This theory holds in practice: our proposed metric is the only metric that is capable of stable optimization for any of our test tasks. The other metrics' geodesics often fail to even evaluate at initialization.



(a) Euclidean (b) Our Metric (c) $A^\top A$ (d) Pullback

Figure 1: The matrix form of various neural network parameterized metrics at initialization time (the Euclidean metric is included as reference). Our metric is the most amenable for training since is has the least random noise. By contrast, the $A^\top A$ and pullback metrics are less amenable for training as the diagonals are less pronounced and the matrices begin looking more like random noise.

3 DIFFERENTIABLE NUMERICAL MANIFOLD OPERATIONS

In this section we introduce ways to compute and differentiate through the core operations on our Deep Riemannian Manifold, enabling gradient-based training. Specifically, we show how to implement the exponential map, logarithmic map, manifold distance, and geodesic interpolation functions in an autodifferentiation framework (Leal et al., 2018). These are all various solutions to the geodesic equation (given below) under varying problem settings.

$$\frac{d^2\gamma^i}{dt^2} + \left(\sum_l \frac{1}{2} g^{il} \left(\frac{\partial g_{jl}}{\partial x^k} + \frac{\partial g_{kl}}{\partial x^j} - \frac{\partial g_{jk}}{\partial x^l} \right) \right) \frac{d\gamma^j}{dt} \frac{d\gamma^k}{dt} = 0, \tag{2}$$

Exponential Map. The exponential map $\exp_x(v)$ is the time 1 solution to the differential equation defined by the geodesic equation with initial value $\gamma(0) = x, \gamma'(0) = v$. Under our Deep Riemannian Manifold, this is a standard initial value problem (IVP) in $\mathbb{R}^n$; we can implement computation and differentiation with a Neural ODE (Chen et al., 2018b) using a linearized version of Equation 2.

Logarithmic Map. The logarithmic map $\log_x(y)$ is the initial velocity of the curve satisfying Equation 2 with $\gamma(0) = x, \gamma(1) = y$. This is the inverse of the exponential map, so $\exp_x(\log_x(y)) = y$, $\log_x(\exp_x(v)) = v$. For our Deep Riemannian Manifold, the logarithmic map is a boundary valued problem (BVP); these differential equations have yet to be fully integrated into deep learning. Therefore, we develop a BVP analogue of Neural ODEs, which we call Neural BVPs.

To solve our BVP, we use the Gaussian process based solver of Arvanitidis et al. (2019) which was built for manifold geodesics. Compared to the methods such as bvp5c (Kierzenka & Shampine, 2008), Arvanitidis et al. (2019) produced more stable results with lower running time.

To differentiate through our BVP, we develop an adjoint sensitivity analysis based on implicit differentiation. The manifold specific case is given below in Proposition 3.1, and the full result for general BVPs with proof is given in Appendix A.

Prop 3.1 (Adjoint Sensitivity For Log Map)**.** *On the manifold $(\mathbb{R}^n, g)$ with $x, y \in \mathbb{R}^n$, suppose that $\log_x(y) = v$. Then*

$$D_x \log(x, y) = -(D_v \exp_x(v))^{-1} \circ D_x \exp_x(v) \tag{3}$$

and

$$D_y \log(x, y) = -(D_v \exp_x(v))^{-1} \tag{4}$$

For our manifold parameters θ, we also have that

$$D_\theta \log_x(y) = -(D_v \exp_x(v))^{-1} \circ D_\theta \exp_x(v) \tag{5}$$

The derivatives of $\exp_x$ can be computed through standard adjoint sensitivity analysis for IVPs.

Previous adjoint sensitivity analyses for BVPs often use the path from x to y for computation (Serban & Petzold, 2002). Our derivation does not need this information, although it does significantly speed up running time by allowing us to batch through function calls in $D \exp_x$.

Geodesic Interpolation. A classic result of Riemannian geometry states that $\exp_x(tv)$ is the time t solution to the IVP defining $\exp_x(v)$ (Lee, 1997). Therefore, interpolating between points x, y can be done with $\exp_x(t \log_x(y))$, which is fully differentiable with our above framework.

Manifold Distance. Given a minimal unit time geodesic γ from x to y, the distance function is the integral $\int_0^1 \left\|\gamma'(t)\right\|_g dt$. While this can be evaluated using a numerical integration method, this is memory intensive and adds more numerical error. Instead, we use another classical result from Riemannian geometry, namely $d_g(x, y) = \left\|\log_x(y)\right\|_g$, to compute our distance (Do Carmo, 2016).

Pseudocode. We give the pseudocode for operations in Module 1. We let θ denote the neural network parameters in our Deep Riemannian Manifold and GEOEQ_θ be the linearized version of Equation 2 with the metric determined by θ. The methods BVPSOLVE, ODEINT, ODEINTBACKWARDS are black-box numerical differential equation solvers

4 Manifold Graph Embeddings

Recent work has shown that non-Euclidean graph structures like trees or cycles naturally lie on manifolds like hyperbolic or spherical space (De Sa et al., 2018; Sarkar, 2011). Motivated by this approach, several methods have proposed using a variety of canonical manifolds–e.g. matrix or product manifolds–to capture other types of graph constructs (Cruceru et al., 2020; Gu et al., 2019). However, these methods are not capable of modelling all possible graph structures, and choosing the correct manifold for graph data requires a heuristically driven hyperparameter search over all possible (combinations of) manifolds (Gu et al., 2019).

We employ our Deep Riemannian Manifolds to learn graph embeddings and find that we are able to overcome both of the aforementioned problems. We prove that theoretically our manifolds can perfectly model all graphs using only three dimensions, and we empirically show that we can learn these during training. These results provide the first approach which can theoretically solve the metric space embedding problem and hint at a deeper connections between graphs and manifolds.

Module 1 Pseudocode for Deep Riemannian Manifold Operations.

function MANIFOLDLOG(x, y, θ)
  $v := \text{BVPSOLVE}(x, y, \text{GEOEQ}_\theta)$
  save x, y, θ and v for the backwards pass.
  return y
function LOGBACKWARDS(∇v)
  $\widehat{y} := \text{MANIFOLDEXP}(x, v, \text{GEOEQ}_\theta)$
  // Construct Jacobian Matrix J
  for $i := 1$ to n **do**
    $\_, j_i, \_ = \text{EXPBACKWARDS}(e_i)$
  $J := [j_1, \ldots, j_n]^\top$
  $dx, \_, d\theta := \text{EXPBACKWARDS}(\nabla v)$
  $\nabla x, \nabla \theta := -(J^{-1})^\top dx, -(J^{-1})^\top d\theta$
  $\nabla y = -(J^{-1})^\top \nabla v$
  returns $\nabla x, \nabla y, \nabla \theta$

function MANIFOLDEXP(x, v, θ)
  $y := \text{ODEINT}(x, v, \text{GEOEQ}_\theta)$
  save x, v, θ and y for the backwards pass.
  return y
function EXPBACKWARDS(∇y)
  $\nabla x, \nabla v, \nabla \theta := \text{ODEINTBACKWARDS}(\nabla y)$
  return $\nabla x, \nabla v, \nabla \theta$

function MANIFOLDINTERP(x, y, t, θ)
  $v := \text{MANIFOLDLOG}(x, y, \theta)$
  return $\text{MANIFOLDEXP}(x, tv, \theta)$

function MANIFOLDDISTANCE(x, y, θ)
  return $\left\|\text{MANIFOLDLOG}(x, y, \theta)\right\|_{g_\theta}$

4.1 THEORY

Given a connected finite graph $G = (V, E)$ with edge costs $c : E \to \mathbb{R}^+$, there exists a natural metric space structure over G with points V and $d_G(v_1, v_2) = \min_{\gamma:\text{path from } v_1 \to v_2} \sum_{e \in \gamma} c(e)$. For a smooth output metric space (X, d_X), we seek to construct an embedding $f : V \to X$ that minimizes the distortion, which takes an optimal minimum value of 1.

$$\text{distortion}(f) = \max_{i<j} \frac{d_G(v_i, v_j)}{d_X(f(v_i), f(v_j))} \cdot \max_{i<j} \frac{d_X(f(v_i), f(v_j))}{d_G(v_i, v_j)} \tag{6}$$

While classical methods find algorithmic ways to embed graph into Euclidean space (Linial et al., 1995; Indyk & Matousek, 2004), a seminal result by Sarkar (2011) showed that trees could be embedded into two-dimensional hyperbolic space with arbitrary low distortion, while the same does not hold for Euclidean space of any dimension.

Theorem 4.1 (Sarkar)**.** *For a finite tree $T = (V, E)$, there exists a sequence of embeddings $f_i : V \to \mathbb{H}^2_{K_i}$ into two-dimensional hyperbolic space with varying curvature K_i s.t. $\lim_{i\to\infty} \text{distortion}(f_i) \to 1$.*

Inspired by this result, we prove an analogous theorem for embedding general graphs into general manifolds. In particular, we show that our formulation of Deep Riemannian Manifolds

Theorem 4.2. *For a finite graph $G = (V, E)$, there exists a sequence of embeddings $f_i : V \to (\mathbb{R}^3, g_i)$ s.t. $\lim_{i\to\infty} \text{distortion}(f_i) = 1$. This can be done with $\mathbb{R}^2$ instead of $\mathbb{R}^3$ if the graph is planar.*

We present a stronger version of the above theorem with proof in Appendix A. These types of theorems are optimal since perfect distortion can not generally be achieved. For a perfect embedding, any vertex of degree ≥ 3 implies splitting geodesics, which can not exist on a Riemannian manifold.

4.2 EXPERIMENTAL RESULTS

Motivated by our theoretical derivations, we empirically test our Deep Riemannian Manifolds by embedding a variety of graphs and compare against pre-existing manifold graph embeddings. Specifically, we test against common manifolds like hyperbolic and spherical space, their products, and various matrix manifolds.

To train our embeddings, we optimize both the embeddings and manifold parameters using gradient descent using stress as our surrogate loss function:

$$\text{stress}(f) = \sum_{i<j} (d_G(v_i, v_j) - d_{\mathcal{M}}(f(v_i), f(v_j)))^2 \tag{7}$$

Further training details such as the network architecture are given in Appendix B.1.

4.2.1 SIMPLE GRAPHS

We begin by embedding simple graphs. We first consider trees and cycles from Cruceru et al. (2020) since these structures embed canonically in hyperbolic and spherical space. We also consider two non-traditional structures which resemble cubes. All graphs are visualized in Figure 2.

Our results are summarized in Table 1. As expected, hyperbolic and spherical spaces produce the best embeddings for trees and cycles. However, for these spaces our Deep Riemannian Manifold is still capable of learning reasonable embeddings. By contrast, our Deep Riemannian Manifold produces the best embeddings for the non-canonical graphs.

We also visualize several of our learned embeddings in Figure 3. We examine the geodesics arising from two dimensional embeddings of our datasets in Euclidean, hyperbolic space, and a Deep Riemannian Manifold. The geometry learned by our Deep Riemannian Manifold is complex and varies with the dataset.

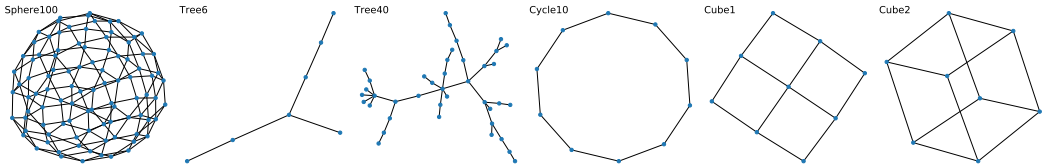


Figure 2: The simple graphs we consider. The first four can be represented by canonical manifolds, while the last two can not be captured by any known Riemannian manifold.

	Sphere100			**Tree6**			**Tree40**			**Cycle10**			**Cube1**			**Cube2**		
	2	5	10	2	5	10	2	5	10	2	5	10	2	5	10	2	5	10
Euclidean	42.64± 21.88	2.85± 0.14	2.82± 0.08	3.14± 2.80	1.43± 0.07	1.51± 0.10	44.64± 19.64	7.85± 1.26	5.62± 0.39	1.67± 0.06	1.91± 0.12	1.79± 0.04	6.21± 1.27	1.83± 0.09	1.74± 0.16	4.37± 0.54	2.05± 0.07	1.98± 0.09
Poincare Ball	12.11± 0.60	2.42± 0.04	2.64± 0.05	1.65± 0.04	1.63± 0.00	1.63± 0.00	8.57± 2.13	3.45± 0.12	2.38± 0.12	1.93± 0.00	1.93± 0.00	1.94± 0.00	1.80± 0.01	1.81± 0.01	1.81± 0.01	2.66± 0.31	2.17± 0.01	2.16± 0.01
Lorentz	8.91±2.25	2.35±0.05	2.41±0.19	1.63± 0.5	1.29± 0.05	1.29± 0.04	9.84± 2.56	2.95±0.62	1.88±0.21	1.72± 0.01	1.73± 0.00	1.72± 0.00	5.32±0.25	1.58±0.02	1.62±0.05	4.66±0.48	2.23±0.12	2.21±0.02
Sphere	2.49± 0.00	2.11± 0.00	1.75± 0.01	1.69± 0.02	1.71± 0.01	1.71± 0.00	11.61± 0.94	6.83± 0.90	5.51± 0.23	1.31± 0.00	1.24± 0.00	1.21± 0.00	1.78± 0.05	1.76± 0.01	1.76± 0.01	2.48± 0.04	2.15± 0.05	2.10± 0.01
Deep Manifold	102.74± 62.74	2.41± 0.01	2.51± 0.06	2.01± 0.80	1.13± 0.04	1.13± 0.05	71.57± 15.44	4.47± 0.65	2.51± 0.37	8.23± 5.69	1.77± 0.55	1.34± 0.07	5.99± 1.49	1.55± 0.15	1.53± 0.09	3.69± 1.02	1.65± 0.03	1.58± 0.05

Table 1: Test Distortion for 2, 5, and 10 dimensional embeddings of the simple graphs. We report the mean and standard deviation over 3 trials. We highlight the lowest distortion as well as any other values which are within one standard deviation.

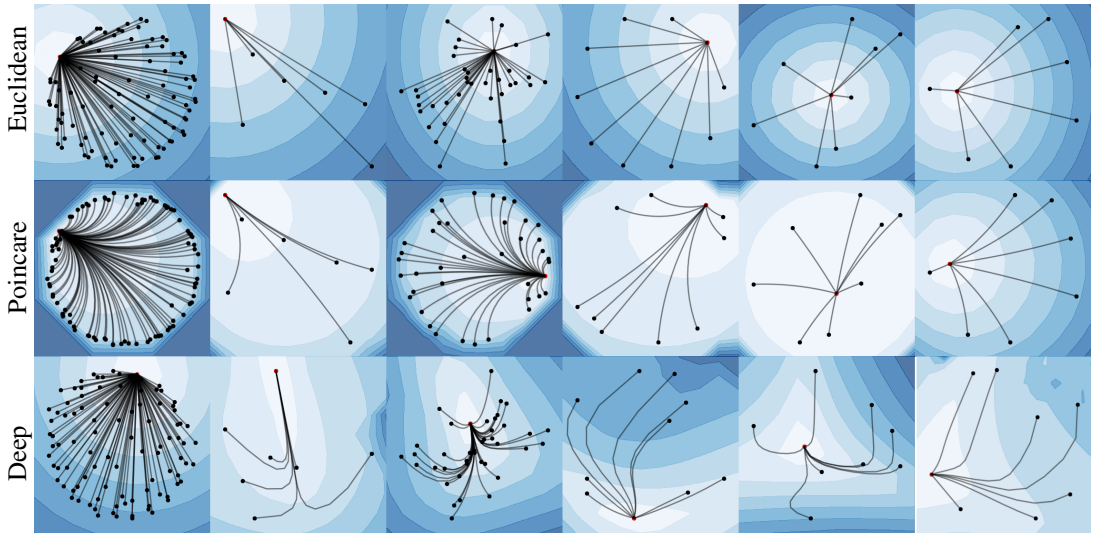


Figure 3: Geodesics and 2D embeddings on the graphs. Each column correspond to the geodesics connected to a point (highlighted in red). The contours show the geodesic distances. The deep metric accurately captures the node distances and induces non-trivial geodesics.

4.2.2 EMPIRICAL GRAPHS

We also test against several real world structures found in the Network Repository Rossi & Ahmed (2015). The three datasets are i) a protein graph, ii) an unweighted version of a Hamiltonian circuit simulation problem, and iii) a social network among southern women. We test on these graphs because they represent a wide variety of real world phenomena and do not have a canonical structure.

We train and report distortion in Table 2. In addition to the standard model spaces, we also test on product spaces and various matrix manifolds, namely the space of SPD matrices and the Grassmannian Gr. Our Deep Riemannian Manifold is able to learn better embeddings than the preexisting methods, which is expected due to the complexity of the graphs. We also visualize an embedding of the protein graph in our Deep Riemannian Manifold in Figure 4. Finally, we note that, for the baseline, finding the optimal manifold to embed into is challenging as the number of possible products increases exponentially with dimension (and there is no heuristic for how matrix manifolds behave when incorporated into a product manifold).

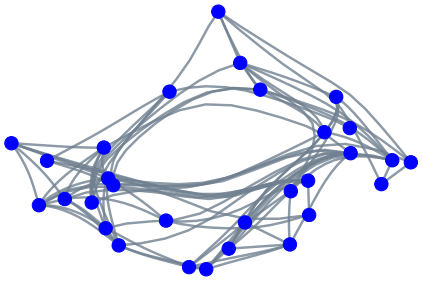


Figure 4: A visualization of our learned embedding for the protein graph. We draw the graph nodes (given in blue) and edges connecting neighbors. Our method is able to capture the nontrivial geometry of the graph.

dim	$\mathcal{M}$	**Protein**	**Hamiltonian**	**Social**
5	$\mathbb{E}^5$	4.259 ± 0.721	5.512 ± 0.494	2.963 ± 0.279
	$\mathbb{H}^5$	2.228 ± 0.033	3.969 ± 0.511	2.329 ± 0.083
	$\mathbb{S}^5$	2.940 ± 0.373	5.596 ± 0.677	4.195 ± 0.440
	$\mathbb{E}^2 \times \mathbb{H}^3$	6.629 ± 0.607	6.545 ± 1.687	4.241 ± 0.290
	$\mathbb{E}^2 \times \mathbb{S}^3$	8.684 ± 0.784	6.184 ± 0.389	5.484 ± 1.825
	$\mathbb{H}^2 \times \mathbb{S}^3$	6.851 ± 0.926	6.414 ± 0.785	5.643 ± 1.638
	Deep	2.008 ± 0.214	3.724 ± 0.315	2.212 ± 0.053
10	$\mathbb{E}^{10}$	3.380 ± 0.347	3.314 ± 0.238	2.359 ± 0.078
	$\mathbb{H}^{10}$	2.178 ± 0.020	2.718 ± 0.029	2.229 ± 0.046
	$\mathbb{S}^{10}$	2.219 ± 0.026	3.485 ± 0.218	2.834 ± 0.355
	$\mathbb{E}^5 \times \mathbb{H}^5$	3.339 ± 0.566	2.964 ± 0.565	2.523 ± 0.234
	$\mathbb{E}^5 \times \mathbb{S}^5$	3.621 ± 0.757	4.117 ± 0.251	3.423 ± 0.806
	$\mathbb{H}^5 \times \mathbb{S}^5$	2.709 ± 0.252	4.318 ± 0.391	3.506 ± 0.350
	$\mathcal{S}_4^+$	3.498 ± 0.239	3.824 ± 0.599	2.503 ± 0.163
	$\mathrm{Gr}(5, 2)$	5.088 ± 1.338	5.228 ± 0.366	2.885 ± 0.113
	Deep	1.958 ± 0.023	2.668 ± 0.218	2.152 ± 0.051

Table 2: Test distortions for 5 and 10 dimensional embeddings of our empirical graphs. We report the mean and standard deviation over 3 trials and highlight the lowest and all values within one SD. For all empirical graphs, our Deep Riemannian Manifold achieves optimal results.

5 GEODESIC FITTING

Learning the geometric structure for optimal paths has long been a core problem in robotics. A complete solution would allow agents to extrapolate shortest paths on new data points, allowing for a better generalization and understanding of the world. When given a direct cost function to minimize, classical techniques like trajectory optimization (Betts, 1998) and the calculus of variations (Morrey, 1966) suffice. But, one can not expect such a nice mathematical formulation in the real world. To overcome this limitation, recent advances invoke the machine learning paradigm to instead learn the geometry of the space from observations alone (Beik-Mohammadi et al., 2021; Rana et al., 2019; Ratliff et al., 2018).

We first formalize the geodesic fitting task and prove that, under suitable conditions, the problem is well-formulated. Then, we compare against the most recent baseline given in Beik-Mohammadi et al. (2021), which is the only prior work which learns a Riemannian manifold to model the environment. We find that their method is insufficient for more difficult problem settings since it only relies on topological information. We show that our Deep Riemannian Manifold overcomes these issues by providing the first geodesic fitting training method that relies on geometric information.

5.1 THEORY

We first formalize the problem and provide a guarantee. Given a finite collection of paths $\gamma_i : [0, 1] \to \mathbb{R}^d$ for $i \in [n]$, we wish to recover a manifold $(\mathbb{R}^d, g)$ s.t. γ_i all satisfy the geodesic equation for the metric g. The following theorem guarantees conditions for which a metric g can be found:

Theorem 5.1 (Geodesic Fitting Viability (Informal))**.** *If the curves all follow basic geodesic properties, ie smooth, injective, don't branch, and each intersection point is well-behaved, then there exists a Riemannian metric g s.t. all the curves are geodesics.*

The full theorem and proof can be found in Appendix A. In addition to several results in Whiting (2011) which characterize trajectory optimization as a geodesic on a Riemannian manifold, our theorem shows that Riemannian manifolds are the natural model for geodesic fitting.

5.2 Experimental Results

We employ our Deep Riemannian Manifolds to fit the geometric information directly. We propose to instead fit a discretized version of the geodesic using ℓ^2 loss. Full details found in Appendix B.2).

5.2.1 Comparison with Previous Methods

We first examine the baseline given in Beik-Mohammadi et al. (2021). The core approach, which was developed in papers such as Arvanitidis et al. (2017); Hauberg (2019); Arvanitidis et al. (2020), learns a stochastic-output VAE and uses the pullback Riemannian metric to determine geodesics. However, the training procedure for the VAE completely disregards the temporal nature of the geodesic. This is because it treats the geodesic as an unordered collection of points. Without the order, we do not expect this method to work well with complex geometric information.

Concretely, we show this by comparing our method to a Euclidean space version of (Beik-Mohammadi et al., 2021). We specifically test situations where a purely topological approach may not work, such as learning a set of geodesics from one point to many points or from many points to many points. To illustrate this, we construct simple test tasks of recovering the Euclidean metric on the plane. As shown in our results in Figure 5, the topological approach of Beik-Mohammadi et al. (2021) struggles while our geometric approach can easily match the geodesics.

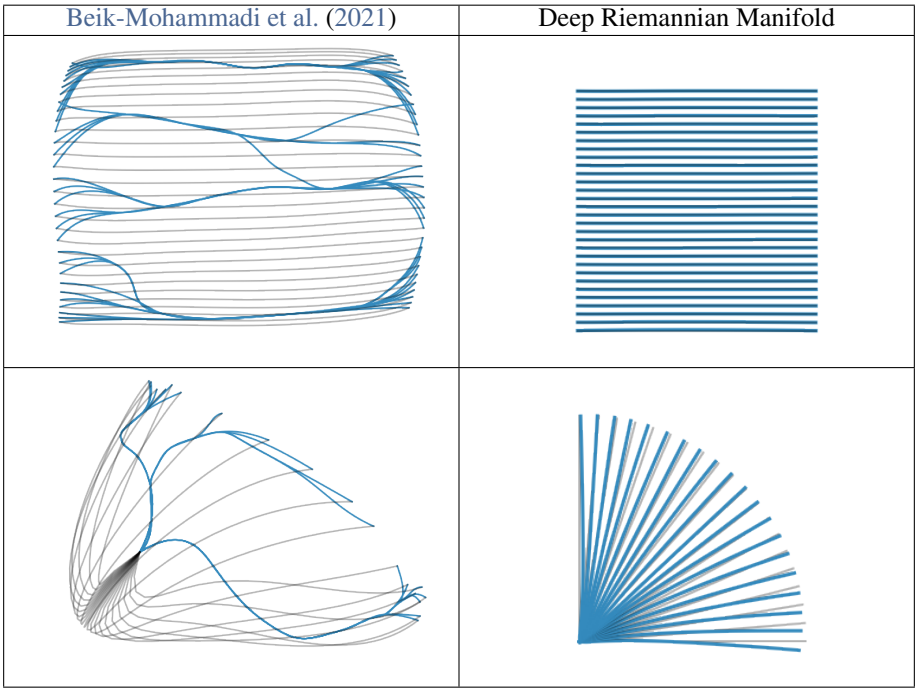


Figure 5: Fitting simple Euclidean geodesics. The blue curves are the learned geodesics, and the gray curves are the ground truth. On the left, the ground truth is warped by going through the VAE. Our method is able to recover the Euclidean geodesics, while Beik-Mohammadi et al. (2021) produces very inaccurate geodesics.

5.2.2 MODELLING ROBOTIC MOTION ON $SE(2)$

We consider a more challenging task of learning the motion of robot with an end effector which moves on the special Euclidean group $SE(2)$. This is the group of roto-rotations and has been extensively used in geometric control and robotics (Bullo & Lewis, 2019; Elbanhawi & Simic, 2014). We fit the geodesics with our Deep Riemannian Manifolds and visualize the fitted geodesics in Figure 6. As expected, our learned geodesics closely match the true geodesics.

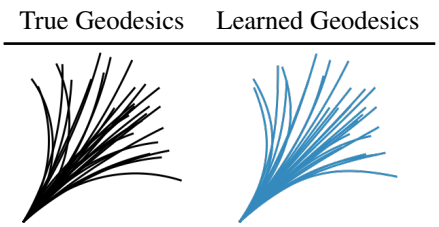


Figure 6: We learn the metric to recover geodesics in SE(2)

6 DISCUSSION

Nontrivial Topology. In this work, we only considered Riemannian manifolds on $\mathbb{R}^n$. While our theory and results show that we can capture nontrivial topological structure, extending our method to nontrivial topologies might be a useful inductive bias. The challenge that needs to be overcome is the construction of the logarithmic map, as one would have to solve a BVP across different charts.

Computational Cost. We tested the time it took for us to compute a forward and backwards pass of our logarithmic map (the core nontrivial operation). We found that it took between 2-3 seconds (on an Nvidia 1080ti) when the geodesic was relatively stable. However, unstable geodesics tended to degrade this performance rather substantially. The major bottlenecks are the solver and the Jacobian construction in the logarithmic map backwards.

Accuracy of Geodesics. While it is near impossible to rigorously verify that our geodesics are minimal, we examined the paths of several Deep Riemannian Manifolds by varying the solver's initial guess. We found that the solver would tend towards the same solution unless the geometry was unstable, or when many geodesics would fail to evaluate.

7 RELATED WORK

Manifold Learning. Despite being similarly named, our work is mostly orthogonal to the field of manifold learning. In particular, while manifold learning methods like t-SNE (van der Maaten & Hinton, 2008) or Isomap (Tenenbaum et al., 2000) reduce dimensions and may or may not preserve distances (Chari et al., 2021), our method is capable of learning a Riemannian manifold from distances and much more general geometric data. However, our work is applicable to recent deep learning based manifold learning approaches (Brehmer & Cranmer, 2020; Kalatzis et al., 2021) as an additional method of optimizing with geometric information.

Differentiable Programming. Our work also incorporates aspects of differentiable programming. In particular, the exponential map is an application of Neural ODEs (Chen et al., 2018b), and the logarithmic map is developed using techniques from both Neural ODEs and implicit differentiation (Domke, 2012; Gould et al., 2016; Agrawal et al., 2019; Amos & Kolter, 2017; Bai et al., 2020; Lou et al., 2020). We hope that our introduction of Neural BVPs may also be of interest beyond our logarithmic map use case.

8 CONCLUSION

We have presented Deep Riemannian Manifolds and have developed a methodology to optimize them through geometric quantities like distance and interpolation. We then tested our method on graph embeddings and geodesic fitting. In both cases, we showed that our method improves upon previous work and sheds light on new angles to approach these problems.

We hope that our paper is the first step in working with and representing more general Riemannian manifolds. In particular, many questions concerning topology and efficient, stable computation still remain unsolved. Potential future researchers may also find fruitful avenues in building other machine learning models on top of our Deep Riemannian Manifolds.

REFERENCES

Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J. Zico Kolter. Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems*, volume 32, pp. 9562–9574, 2019.

Brandon Amos and J. Z. Kolter. Optnet: Differentiable optimization as a layer in neural networks. In *ICML*, 2017.

Georgios Arvanitidis, Lars Kai Hansen, and Søren Hauberg. Latent space oddity: on the curvature of deep generative models. *arXiv preprint arXiv:1710.11379*, 2017.

Georgios Arvanitidis, Søren Hauberg, Philipp Hennig, and Michael Schober. Fast and robust shortest paths on manifolds learned from data. *arXiv preprint arXiv:1901.07229*, 2019.

Georgios Arvanitidis, Søren Hauberg, and Bernhard Schölkopf. Geometrically enriched latent spaces. *arXiv preprint arXiv:2008.00565*, 2020.

Shaojie Bai, V. Koltun, and J. Z. Kolter. Multiscale deep equilibrium models. *ArXiv*, abs/2006.08656, 2020.

H. Beik-Mohammadi, Søren Hauberg, Georgios Arvanitidis, G. Neumann, and L. Rozo. Learning riemannian manifolds for geodesic motion skills. *ArXiv*, abs/2106.04315, 2021.

John T. Betts. Survey of numerical methods for trajectory optimization. *Journal of Guidance Control and Dynamics*, 21:193–207, 1998.

S. Bonnabel. Stochastic gradient descent on riemannian manifolds. *IEEE Transactions on Automatic Control*, 58:2217–2229, 2013.

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. *Physical Review D*, 103(7), Apr 2021. ISSN 2470-0029. doi: 10.1103/physrevd.103.074504. URL http://dx.doi.org/10.1103/PhysRevD.103.074504.

Johann Brehmer and Kyle Cranmer. Flows for simultaneous manifold learning and density estimation. *arXiv preprint arXiv:2003.13913*, 2020.

Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges, 2021.

Francesco Bullo and Andrew D Lewis. *Geometric control of mechanical systems: modeling, analysis, and design for simple mechanical control systems*, volume 49. Springer, 2019.

Tara Chari, Joeyta Banerjee, and Lior Pachter. The specious art of single-cell genomics. *bioRxiv*, 2021.

Nutan Chen, Alexej Klushyn, Richard Kurle, Xueyan Jiang, Justin Bayer, and P. V. D. Smagt. Metrics for deep generative models. In *AISTATS*, 2018a.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 31*, pp. 6571–6583. Curran Associates, Inc., 2018b.

R. Cohen, P. Eades, Tao Lin, and F. Ruskey. Three-dimensional graph drawing. In *Graph Drawing*, 1994.

T. Cohen, M. Geiger, Jonas Köhler, and M. Welling. Spherical cnns. *ArXiv*, abs/1801.10130, 2018.

Calin Cruceru, Gary Bécigneul, and Octavian-Eugen Ganea. Computationally tractable riemannian manifolds for graph embeddings. *arXiv preprint arXiv:2002.08665*, 2020.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989. doi: 10.1007/BF02551274. URL https://doi.org/10.1007/BF02551274.

Christopher De Sa, Albert Gu, Christopher Ré, and Frederic Sala. Representation tradeoffs for hyperbolic embeddings. *Proceedings of machine learning research*, 80:4460, 2018.

Manfredo P Do Carmo. *Differential geometry of curves and surfaces: revised and updated second edition*. Courier Dover Publications, 2016.

Justin Domke. Generic methods for optimization-based modeling. In *AISTATS*, volume 22, pp. 318–326, 2012.

M. Elbanhawi and M. Simic. Sampling-based robot motion planning: A review. *IEEE Access*, 2: 56–77, 2014. doi: 10.1109/ACCESS.2014.2302442.

J. Gallier and J. Quaintance. Differential geometry and lie groups: A computational perspective. 2020.

P. Goulart, Y. Nakatsukasa, and Nikitas Rontsis. Accuracy of approximate projection to the semidefinite cone. *ArXiv*, abs/1908.01606, 2019.

Stephen Gould, Basura Fernando, Anoop Cherian, Peter Anderson, Rodrigo Santa Cruz, and Edison Guo. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. 2016.

Albert Gu, Frederic Sala, Beliz Gunel, and Christopher Ré. Learning mixed-curvature representations in product spaces. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HJxeWnCcF7.

Søren Hauberg. Only bayes should learn a manifold (on the estimation of differential geometric structure from data), 2019.

P. Indyk and J. Matousek. Low-distortion embeddings of finite metric spaces. In *Handbook of Discrete and Computational Geometry, 2nd Ed.*, 2004.

Dimitris Kalatzis, Johan Ziruo Ye, Jesper Wohlert, and Søren Hauberg. Multi-chart flows, 2021.

J Kierzenka and Lawrence Shampine. A bvp solver that controls residual and error. *European Society of Computational Methods in Sciences and Engineering (ESCMSE) Journal of Numerical Analysis, Industrial and Applied Mathematics*, 3:27–41, 01 2008.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

Tom LaGatta and Jan Wehr. Geodesics of random riemannian metrics. *Communications in Mathematical Physics*, 327(1):181–241, Feb 2014. ISSN 1432-0916. doi: 10.1007/s00220-014-1901-8. URL http://dx.doi.org/10.1007/s00220-014-1901-8.

Allan M. M. Leal et al. autodiff, a modern, fast and expressive C++ library for automatic differentiation. https://autodiff.github.io, 2018. URL https://autodiff.github.io.

J.M. Lee. *Riemannian Manifolds: An Introduction to Curvature*. Graduate Texts in Mathematics. Springer New York, 1997. ISBN 9780387982717. URL https://books.google.com/books?id=ZRQgH7FQafgC.

Mario Lezcano-Casado. Trivializations for gradient-based optimization on manifolds. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 9154–9164, 2019.

N. Linial, E. London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15:215–245, 1995.

Aaron Lou, Isay Katsman, Qingxuan Jiang, Serge Belongie, Ser-Nam Lim, and Christopher De Sa. Differentiating through the fréchet mean, 2020.

Jonathan Masci, D. Boscaini, Michael M. Bronstein, and Pierre Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, pp. 832–840, 2015.

Charles Bradfield Morrey. Multiple integrals in the calculus of variations. 1966.

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

Muhammad Asif Rana, Anqi Li, Harish Chaandar Ravichandar, Mustafa Mukadam, S. Chernova, Dieter Fox, Byron Boots, and Nathan D. Ratliff. Learning reactive motion policies in multiple task spaces from human demonstrations. In *CoRL*, 2019.

Nathan D. Ratliff, Jan Issac, and Daniel Kappler. Riemannian motion policies. *ArXiv*, abs/1801.02854, 2018.

Ryan A. Rossi and Nesreen K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015. URL http://networkrepository.com.

Rik Sarkar. Low distortion delaunay embedding of trees in hyperbolic plane. In *Proceedings of the 19th International Conference on Graph Drawing*, GD'11, pp. 355–366, Berlin, Heidelberg, 2011. Springer-Verlag.

R. Serban and L. Petzold. Efficient computation of sensitivities for ordinary differential equation boundary value problems. *SIAM J. Numer. Anal.*, 40:220–232, 2002.

Joshua B. Tenenbaum, Vin De Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290 5500:2319–23, 2000.

Laurens van der Maaten and Geoffrey E. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

J. Whiting. Path optimization using sub-riemannian manifolds with applications to astrodynamics. 2011.

A STATEMENTS WITH PROOFS

Prop A.1 (Smoothness of Deep Metric)**.** *If f_{nn} has a smooth activation function, then $g := (\exp \circ \text{sym} \circ f_{nn})$ is smooth as well.*

Proof. Since matrix multiplication is smooth and a composition of smooth functions is smooth, f_{nn} is smooth. sym is a smooth function since it is simple reshaping, and exp is smooth. Therefore g is smooth. □

Prop A.2 (Universal Approximation of Riemannian Metricson $\mathbb{R}^n$)**.** *On a compact set D of $\mathbb{R}^n$, for any metric h and $\epsilon > 0$ there exists a neural network metric g of the form in Equation 1 s.t. $\sup_{x \in D} \|g_x - h_x\| < \epsilon$, where $\|\cdot\|$ is the standard ℓ^2 norm.*

Proof. On a compact set, the exponential map is Lipschitz since it is continuous. Let K be the Lipschitz constant. Let $f : \mathbb{R}^n \to \mathbb{R}^{n(n+1)/2}$ be the function s.t. $h = \exp \circ \text{sym} \circ f$; f is well defined since exp and sym are bijective on our domains. Construct a neural network f_{nn} s.t. $\sup_{x \in D} \left\| f(x) - f_{\text{nn}}(x) \right\| < \frac{\epsilon}{K}$ (Cybenko, 1989). Then we see that $\sup_{x \in D} \|h_x - g_x\| \leq K \left\| f(x) - f_{\text{nn}}(x) \right\| < \epsilon$. □

Theorem A.3 (Universal Approximation of all Riemannian Manifolds)**.** *For a compact Riemannian manifold $(\mathcal{M}, g)$ and any $\epsilon > 0$, there exists a Deep Riemannian Manifold $(\mathbb{R}^n, g_{\text{nn}})$ with $n = 2 \dim \mathcal{M}$ and an embedding $f : \mathcal{M} \to \mathbb{R}^n$ s.t. $|d_g(x, y) - d_{g_{\text{nn}}}(f(x), f(y))| < \epsilon$ for all $x, y \in \mathcal{M}$.*

Proof. Let f be a Whitney embedding. For a point $x \in \mathcal{M}$ we define a metric on $T_{f(x)}\mathbb{R}^n$ as follows. For the $T_{f(x)}f(\mathcal{M})$ component, let the metric just be g. For the normal space $N_{f(x)}f(\mathcal{M})$, let the metric be $Ng_{\mathbb{E}}$, where $g_{\mathbb{E}}$ is the Euclidean metric and N is some constant. Note that g is a proper metric on $f(\mathcal{M})$. We can extend this metric locally to an open neighborhood U of $f(\mathcal{M})$. We also let the metric on the manifold $\mathbb{R}^n \setminus f(\mathcal{M})$ be $Ng_{\mathbb{E}}$. Using a smooth partition of unity, we can combine these metrics to get a metric on $\mathbb{R}^n$. As we take the limit as N goes to ∞, the geodesic equation will cause our geodesics between points on $f(\mathcal{M})$ to adhere closer and closer to $f(\mathcal{M})$. Note that f is an isometry, so the distance on $(\mathbb{R}^n, g_{\text{nn}})$ will converge to the distance on the submanifold. In addition to the previous proposition, this proves our result. □

Theorem A.4 (Adjoint Sensitivity For BVPs)**.** *Suppose we are given a BVP $B(x, y)$ with corresponding IVP $I(x, v)$ s.t. $I(x, B(x, y)) = y$. Suppose our solution to $B(x_0, y_0)$ is b. Then*

$$D_x B(x_0, y_0) = -D_v I(x_0, b)^{-1} \circ D_x I(x_0, b) \tag{8}$$

$$D_y B(x_0, y_0) = -D_v I(x_0, b)^{-1} \tag{9}$$

Note that the derivatives of I can be computed through the adjoint-sensitivity analysis. Furthermore, if our ODE problems are controlled by some parameter θ, then

$$D_\theta B(x_0, y_0) = -D_v I(x_0, b)^{-1} \circ D_\theta I(x_0, b)$$

Proof. The core of this proof comes from the fact that $I(x_0, B(x_0, y_0)) = y_0$. Differentiating this identity w.r.t y using the chain rule gives us

$$\begin{aligned} I &= D_y y_0 \\ &= (D_y I)(x_0, B(x_0, y_0)) \\ &= \begin{bmatrix} D_x I(x_0, B(x_0, y_0)) \\ D_v I(x_0, B(x_0, y_0)) \end{bmatrix} \begin{bmatrix} 0 & D_y B(x_0, y_0) \end{bmatrix} \\ &= D_v I(x_0, B(x_0, y_0)) D_y B(x_0, y_0) \end{aligned}$$

rearranging and setting $b = B(x_0, y_0)$ gives us that $D_y B(x_0, y_0) = -D_v I(x_0, b)^{-1}$, as desired.

For x, differentiating through the same identity gives us that

$$
\begin{aligned}
0 &= D_x y_0 \\
&= (D_x I)(x_0, B(x_0, y_0)) \\
&= \begin{bmatrix} D_x I(x_0, B(x_0, y_0)) \\ D_v I(x_0, B(x_0, y_0)) \end{bmatrix} \begin{bmatrix} I & D_x B(x_0, y_0) \end{bmatrix} \\
&= D_x I(x_0, B(x_0, y_0)) + D_v I(x_0, B(x_0, y_0)) D_x B(x_0, y_0)
\end{aligned}
$$

and rearranging gives us that $D_x B(x_0, y_0) = -D_v I(x_0, b)^{-1} \circ D_x I(x_0, b)$.

For θ, the proof is very similar to the proof for x. In particular, from the same differentiation we get that

$$
\begin{aligned}
0 &= D_\theta y_0 \\
&= (D_\theta I)(x_0, B(x_0, y_0), \theta) \\
&= \begin{bmatrix} D_x I(x_0, B(x_0, y_0)) \\ D_v I(x_0, B(x_0, y_0)) \\ D_\theta I(x_0, B(x_0, y_0)) \end{bmatrix} \begin{bmatrix} 0 & D_\theta B(x_0, y_0) & I \end{bmatrix} \\
&= D_v I(x_0, B(x_0, y_0)) D_\theta B(x_0, y_0) + D_\theta I(x_0, B(x_0, y_0))
\end{aligned}
$$

and rearranging gives us $D_\theta B(x_0, y_0) = -D_v I(x_0, b)^{-1} \circ D_\theta I(x_0, b)$. □

Theorem A.5 (Quasi-Isometric Finite Metric Space Embeddings into Riemannian Manifolds)**.** *Let $G = (V, E)$ be a finite connected graph. There exists a topological embedding $f : V \to \mathbb{R}^3$ and a sequence of Riemannian metrics $g_1, g_2, \ldots$ on $\mathbb{R}^3$ s.t. $\lim_{n\to\infty} d_{g_n}(f(x), f(y)) = d_G(x, y)$ for all $x, y \in V$.*

Proof. A quick note on notation. We assume that the absolute value $|\cdot|$ and the distance function $d(\cdot, \cdot)$ refers to the distance in Euclidean space. All other distances (whether it be the graph distance or distance induced by the Riemannian metric) will be marked as such.

Note that there always exists a topological embedding into $\mathbb{R}^3$ Cohen et al. (1994). So, it suffices to construct the metrics given any such topological embedding f. The trivial case of $|V| \leq 2$ is direct. For the case $|V| > 3$, we proceed by defining "bump" metrics on the graph which gives the desired length convergence.

To construct these bump metrics, we first consider a tubulur neighborhood of the graph that is sufficiently small to avoid "undesirable" self-intersection. Then, we construct the metric locally using bump functions on this neighborhood. Finally, we smooth this out.

We start by examining the topological embedding. For an edge $e \in E$ with endpoints $x, y \in V$, let $f(e)$ be the line segment joining $f(x)$ and $f(y)$ and define $|f(e)| = |f(x) - f(y)|$. Define θ to be the minimum angle formed by the intersection of any two $f(e)$.

We construct the metrics g_n as follows. Consider the conditions.

- $\frac{2 \cot \frac{\theta}{2}}{n} > \min_{x,y \in V, x \neq y, (x,y) \in E} |f(x) - f(y)|$
- $\frac{2}{n} > \min_{x,y \in V, x \neq y} |f(x) - f(y)|$
- $\frac{2}{n} > \min_{e_1, e_2 \in E, e_1 \cap e_2 = \emptyset} |f(e_1) - f(e_2)|$
- or $\max_{e \in E} \frac{|e|}{|f(e)|} \leq \sqrt{n}$

The first three condition happen when the tubular neighborhood we will construct around our graph contains a self-intersection, while the last condition (and in particular our usage of the $\sqrt{n}$ term) control whether we can smooth out the graph. If any of these conditions hold, let g_n be the Euclidean

metric $g_{\mathbb{E}}$ (we throw it away with this default metric). Otherwise proceed as follows. For all $e \in E$ define $h_e^n(x) = (\sqrt{n} - \frac{|e|}{|f(e)|}) e^{-\frac{1}{1/n^2 - d(x,f(e))^2}} \cdot \mathbb{1}_{\{d(x,f(e))<1/n\}}$. This is a bump function with support $D_e^n = \{x : d(x, f(e)) < 1/n\}$. Define a smooth partition of unity p_e^n for the supports D_e^n and let $h_n(x) = \sqrt{n} - \sum_{e \in E} h_e^n(x) p_e^n(x)$. Note that h_n extends smoothly to $\mathbb{R}^3$ since $h_e^n(x) = 0$ when p_e^n is not defined. Define $g_n(x) = h_n \cdot g_{\mathbb{E}}$.

By our conditions on n above, we note the following properties.

1. For each e, we there exists a set $\overline{D_e^n} \subset D_e^n \setminus \bigcup_{e' \in E, e \neq e'} D_{e'}^n$ s.t. $\overline{D_e^n} \cap f(V) = \emptyset$, $\overline{D_e^n}$ is connected, and $s_e^n = \overline{D_e^n} \cap f(e)$ is a connected sub-segment of $f(e)$ with $|s_e^n| \geq |f(e)| - \frac{2\cot\frac{\theta}{2}}{n}$. In particular $\overline{D_e^n}$ is a cylinder in between the two endpoinds of e, and we can denote the endpoints of s_e^n as x' and y' (which are closer to $f(x)$ and $f(y)$ respectively).
2. For each $x \in V$, $f(x) \in D_e^n$ iff $x \in e$.
3. For $e_1 \neq e_2$, $D_{e_1}^n \cap D_{e_2}^n \neq \emptyset$ iff $D_{e_1}^n \cap D_{e_2}^n$ is a connected neighborhood of $f(e_1 \cap e_2)$.
4. g_n is a smooth Riemannian metric. This is because h_n is smooth by construction and > 0.

First, we show that $\lim_{n\to\infty} d_{g_n}(f(x), f(y)) \leq d_G(x, y)$. Let $x = x_1, x_2, \ldots, x_m = y$ be the path of a geodesic between x and y on the graph (V, E). Consider the images of the points $f(x_1), \ldots, f(x_m)$ and the path γ created by following $f(x_1) \to \cdots \to f(x_m)$ through straight line segments. For all $i \in [m-1]$, we have that

$$\begin{aligned} L_{g_n}(f(x_i, x_{i+1})) &= L(s_e^n) + L((f(x), x')) + L((y', f(y)) \\ &\leq \frac{|(x_i, x_{i+1})|}{|f((x_i, x_{i+1}))|} |s_e^n| + \sqrt{n}(|(f(x), x')| + |(y', f(y)|) \\ &\leq |(x_i, x_{i+1})| + \frac{2\cot\frac{\theta}{2}}{\sqrt{n}} \end{aligned}$$

$$\begin{aligned} L_{g_n}(f(x_i, x_{i+1})) &\geq L(s_e^n) \\ &\geq \frac{|(x_i, x_{i+1})|}{|f((x_i, x_{i+1}))|} (|f((x_i, x_{i+1}))| - \frac{2\cot\frac{\theta}{2}}{n}) \end{aligned}$$

and so $\lim_{n\to\infty} L_{g_n}(\gamma) = d_G(x, y)$. By taking a limit on $d_{g_n}(f(x), f(y)) \geq L_{g_n}(\gamma)$, we get that $\lim_{n\to\infty} d_{g_n}(f(x), f(y)) \leq d_G(x, y)$.

Now, we show that $\lim_{n\to\infty} d_{g_n}(f(x), f(y)) \geq d_G(x, y)$. Since g_n is uniformly $\sqrt{n} \cdot g_{\mathbb{E}}$ on $(\bigcup_{e \in E} D_e^n)^c$, for a sufficiently large n no geodesic connecting points in $f(V)$ will ever enter $(\bigcup_{e \in E} D_e^n)^c$. By properties 2 and 3, this implies that geodesic passes through $D_{e_1}^n$ can only enter $D_{e_2}^n$ by passing through a neighborhood which uniquely contains $f(e_1 \cap e_2)$. For a geodesic under g_n from $f(x)$ to $f(y)$, it must pass through the neighorhoods $U_1, U_2, \ldots, U_m$ which are indexed by vertices $x_i \in V$, uniquely contain $f(x_i)$, and satisfy $x = x_1, y = x_m$. Since the geodesic passes through x_i and x_{i+1} for $i \in [m-1]$, we note that it must pass through $\overline{D_e^n}$. Since $d_{g_n}(U_i, U_{i+1}) \geq L_{g_n}(s_{(x_i, x_{i+1})}^n)$, this means $d_{g_n}(f(x), f(y)) \geq \sum_{i=1}^{m-1} d_{g_n}(U_i, U_{i+1}) \geq \sum_{i=1}^{m-1} L_{g_n}(s_{(x_i, x_{i+1})}^n)$. In particular, $\lim_{n\to\infty} d_{g_n}(f(x), f(y)) \geq \lim_{n\to\infty} \sum_{i=1}^{m-1} L_{g_n}(s_{(x_i, x_{i+1})}^n) = \sum_{i=1}^{m-1} |(x_i, x_{i+1})| \geq d_G(x, y)$.

Therefore, we have proven the desired: $\lim_{n\to\infty} d_{g_n}(f(x), f(y)) = d_G(x, y)$. □

Remark. *The above theorem holds for an embedding into $\mathbb{R}^2$ if (V, E) is a planar graph. This is because the construction of the bump metrics is agnostic to dimension as long as a topological embedding is provided.*

Theorem A.6 (Existence of Metric for Geodesics)**.** *Let n and d be positive integers and $\{\gamma_i : [0,1] \to \mathbb{R}^d\}_{i=1}^n$ be a set of smooth injective curves. Suppose that the intersection between any two curves is a finite set. Furthermore, suppose that for all intersection points x with intersecting geodesics $\gamma_{i_1}, \dots, \gamma_{i_m}$ that the set $\{\gamma_{i_j}'(x)\}_{j=1}^m$ is linearly independent. Then there exists a Riemannian metric g on $\mathbb{R}^d$ s.t. γ_i is a geodesic under g for all i.*

To prove this, we make use of the following lemma:

Lemma A.7 (Extension of Linearly Independent Vector Fields)**.** *Suppose that vector fields $X_1, \dots, X_n$ are defined on an open subset U of a manifold $\mathcal{M}$ and are linearly independent at a point p. Then there is a local neighborhood $V \subset U$ of p s.t. $X_1, \dots, X_n$ are linearly independent for all $p \in V$.*

Proof. First, assume $\dim \mathcal{M} = n$. Recall that the determinant is smooth, so the determinant of the vector fields $X_1, \dots, X_n$ (which we denote d) is a smooth function on U. Since $d(p) > 0$, we can use smoothness to find an open neighborhood V of p s.t. $0 \notin d(V)$. This is exactly the set where $X_1, \dots, X_n$ are linearly independent, as linear independence occurs exactly when $d \neq 0$.

When $m = \dim \mathcal{M} > n$, we can linearly extend X_i to a basis of $T_p\mathcal{M}$. We can smoothly extend the vector fields $X_{n+1}, \dots, X_m$ to a neighborhood U' around p. We would then apply the above process to $X_1, \dots, X_m$ on the subset $U' \cap U$ to get a set V. In this set V, $X_1, \dots, X_m$ are linearly independent, so $X_1, \dots, X_n$ are as well. □

Proof of Thm A.6. We construct vector fields X_i defined on $\gamma_i([0,1])$ by defining X_i to be $\gamma_i'(t)$. Our goal is to extend these vector fields to a local neighborhood of the curves to define a metric.

Let $D_i^\epsilon = \{x : d(\gamma_i([0,1]), x) < \epsilon\}$. Let $\epsilon > 0$ be sufficiently small s.t. $D_i^\epsilon \cap D_j^\epsilon$ consists only of neighborhoods of the intersection points of γ_i and γ_j. By the conditions on our curves, we can smoothly extend our vector fields X_i to all D_i^ϵ.

Let x be an intersection point with $\gamma_{i_1}(x), \dots, \gamma_{i_m}(x)$ the intersecting geodesics. Define U_x to be the neighborhood of x in $\bigcap_{j=1}^m D_{i_j}^\epsilon$. We apply Lemma A.7 to U_x, x, and the vector fields $X_{i_1}, \dots, X_{i_m}$ to find local neighborhoods V_x s.t. $X_{i_1}, \dots, X_{i_m}$ are linearly independent on each V_x.

Now, we shrink our radius ϵ to a suitable radius ϵ' s.t. $D_i^{\epsilon'} \cap D_j^{\epsilon'} \subset \bigcup_{x \in \gamma_i \cap \gamma_j} V_x$. This means that around each intersection point the vector fields are linearly independent.

Around each intersection point, we extend the vector fields $X_{i_1}, \dots, X_{i_m}$ of the intersecting curves to a local frame $X_{i_1}, \dots, X_{i_m}, Y_1, \dots, Y_{m-n}$. We can extend these local frames to the rest of $\bigcup_{i=1}^n D_i^{\epsilon'}$ s.t. the vector fields X_i remain basis vectors because the neighborhoods form a submanifold.

Now, we construct the metric on $\bigcup_{i=1}^n D_i^{\epsilon'}$ as $g(Y_i, Y_j) = \delta_{ij}$. This metric mimics the standard Euclidean metric (except for our new vector field instead of the standard directions), so geodesics have derivatives which are (a constant multiple of) Y_i. In particular this means that the curves γ_i are all geodesics.

Finally, we define the metric over all of $\mathbb{R}^d$. We can patch together a metric on $\mathbb{R}^d \setminus \overline{\bigcup_{i=1}^n D_i^{\epsilon'/2}}$ (say the Euclidean metric) and our above metric on $\bigcup_{i=1}^n D_i^{\epsilon'}$ using a partition of unity to define a metric on all of $\mathbb{R}^d$. Under this metric, all γ_i are geodesics, as desired. □

Remark. *The condition is not tight. In particular, all we need is that the geodesics behave nicely around intersection points, which our linear independence condition guarantees. This means that there can be $\geq d$ geodesics intersecting at a point as long as they are well behaved.*

B Training Details

B.1 Manifold Graph Embeddings

For our graph tasks, we randomly initialize embeddings in Euclidean space with a mean 0 unit variance normal distribution. We train for 500 epochs. For all methods, we use a batch size of 32

on larger graphs and 8 on smaller graphs in order to ensure proper mini-batching. We optimize the embeddings with Riemannian SGD Bonnabel (2013) with a learning rate of $1e-2$.

For learned curvature models, we parameterize the curvature (which is either a positive or negative value) with the exponential map. We update the curvature using the Adam Optimizer Kingma & Ba (2015) with step size of $1e-2$, betas $(0.9, 0.999)$, and cosine annealing.

For the Deep Riemannian Manifold, our metric neural network is a two-layer neural network with hidden size 32 and $\tanh$ activation. We also update the manifold parameters with the same optimizer as for the learned curvature models, although we have an initial learning rate of $5e-3$. In contrast to the other manifolds, we initialize our embeddings in an initial "burnin" phase in which we optimize for the Euclidean metric. This helps stabilize the initialization and prevents embeddings from overly interfering with other geodesics.

B.2 Geodesic Fitting

For our geodesics experiments, recall that we only learn the neural network parameters. We optimize our neural network parameters with an Adam optimizer with learning rate $5e{-}3$ and betas $(0.9, 0.999)$. Similar to the graph embedding example, we also bound the norm of the gradients by 5 to account for gradient explosions before updating. Furthermore, for both experiments our neural network has 2 layers, $\tanh$ activation, and a hidden size of 32. We train our method with 50 epochs with a batch size of 5.

To compare with Beik-Mohammadi et al. (2021), we generate two datasets of Euclidean geodesics. The first is a set of 30 geodesics from $(0, i)$ to $(1, i)$, where i ranges from 0 to 1 at equal intervals. We discretize the geodesic into 30 points. The first is a set of 30 geodesics from $(0, 0)$ to $(\cos\theta, \sin\theta)$, where θ ranges from 0 to $\pi/2$ at equal intervals. We again discretize the geodesic into 30 points.

The baseline is taken from Arvanitidis et al. (2020), which is the Euclidean version of Beik-Mohammadi et al. (2021). The VAE has a hidden layer size of 32 with 2 layers. The latent space has dimension 2. The method is trained with the Adam optimizer (Kingma & Ba, 2015) for 500 epochs (with 175 epochs of warmup) with a learning rate of $1e-3$ and standard β. Batch size is 128. The RBF is trained for 5000 epochs and we use no ambient space metric since they produce no notable change in quality.

For the $SE(2)$ example, we train with 800 training examples in batches of size 128 and test with 200 test examples. We train for 50 epochs and take $T = 10$.

C Additional Experiments

We include additional geodesic visualizations here. In particular, we visualize geodesics emanating from multiple points for one of the tree datasets. These are shown in Figure 7.

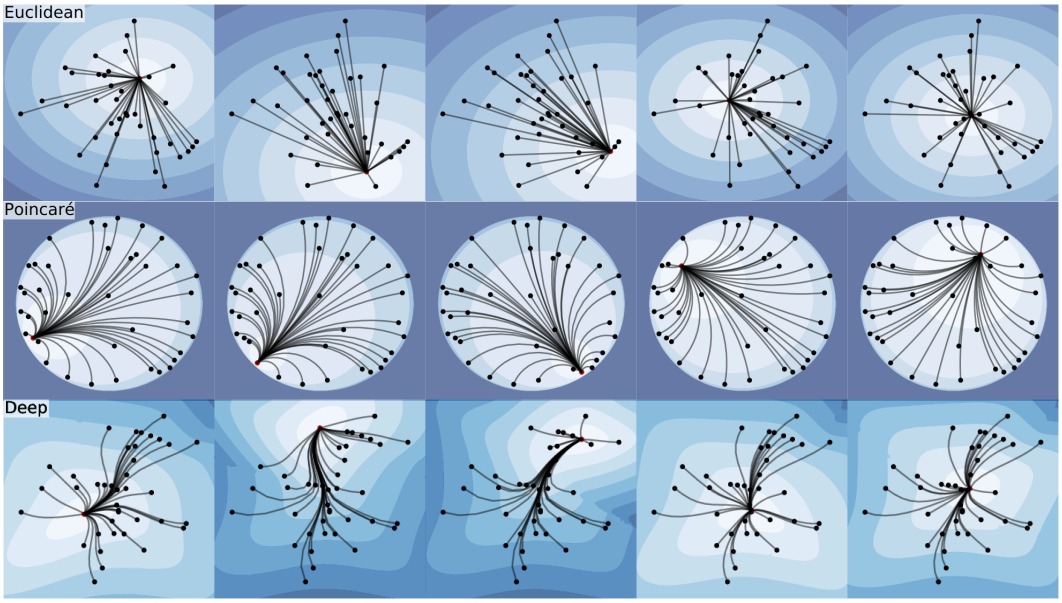


Figure 7: Geodesics and 2D embeddings on the Tree40 graph, where each column correspond to the geodesics connected to a point (highlighted in red). The contours show the geodesic distances. The deep metric accurately captures the node distances and induces non-trivial geodesics.